

# Ordinal Machines

Eric Steinhart

## 1. Introduction

I aim to extend the concepts of algorithms and machines arbitrarily far into the set-theoretic hierarchy of mathematical objects. I work within the universe  $V$  of *sets* defined by Von Neumann - Bernays - Godel *class* theory (VBG) and the von Neumann theory of *ordinal numbers* developed in VBG (Hamilton, 1982: chs. 4 & 6). Since the theory of ordinals is essential to the theory of these machines, I refer to them as *ordinal machines*.

## 2. The Theory of Ordinal Numbers

I begin with the von Neumann theory of ordinals. According to von Neumann, 0 is an ordinal; the ordinal  $n+1$  is the collection of all ordinals from 0 to  $n$ . So: 1 is  $\{0\}$ ; 2 is  $\{0, 1\}$ ; 3 is  $\{0, 1, 2\}$ , and  $n+1$  is  $\{0, 1, \dots, n\}$ . The finite ordinals are the familiar whole numbers 0, 1, 2, and so on. I extend the theory of finite ordinals to the infinite by using the von Neumann identification of ordinals with sets. According to von Neumann, the set  $\{\}$  is the integer 0; if  $n$  is an integer, then the set  $(n \cup \{n\})$  is the integer  $(n+1)$ . 0 is the *initial* ordinal. If  $n$  is an ordinal, then  $(n+1)$  is the *successor* of  $n$ .

The von Neumann theory of ordinals is easily extended to the infinite by defining *limit* ordinals. The first limit ordinal  $\omega$  is defined recursively:  $\{\}$  is in  $\omega$ ; if  $n$  is in  $\omega$ , then  $(n \cup \{n\})$  is in  $\omega$ .  $\omega$  is the set of all the finite ordinals. It is the set of the whole numbers. The first limit ordinal  $\omega$  is the set  $\{0, 1, 2, \dots\}$ .  $\omega$  has a successor  $(\omega+1)$ . The ordinal  $\omega+1$  is the set  $\{0, 1, 2, \dots, \omega\}$ .  $\omega+1$  contains *one more* member than  $\omega$ .  $\omega$  and  $\omega+1$  are very different. While every member of  $\omega$  is finite, one member of  $\omega+1$  is not finite. While  $\omega$  contains no greatest (i.e. final) member,  $\omega+1$  does contain a greatest member. An ordinal is *complete* if and only if it has some greatest member; it is *incomplete* otherwise.

The *order type* of any well-ordered set  $S$  is the least ordinal  $\kappa$  such that  $\kappa$  is ordered by magnitude and there is an order-preserving 1-1 correspondence from  $\kappa$  to  $S$ . For example: the order-type of  $\{0, 1/2, 3/4, 7/8, \dots\}$  is  $\omega$ ; the order-type of  $\{0, 1/2, 3/4, 7/8, \dots, 1\}$  is  $\omega+1$ . There is no order-preserving 1-1 map from the ordinal  $\omega = \{0, 1, 2, \dots\}$  to the set  $S = \{0, 1/2, 3/4, 7/8, \dots, 1\}$  so long as  $\omega$  is ordered by magnitude.<sup>1</sup> The least ordinal that is "order isomorphic" (when ordered by magnitude) to  $S$  is  $\omega+1$ . While  $\omega$  and  $\omega+1$  have the same size (cardinality  $\aleph_0$ ), they do not have the same order type.  $\omega$  and  $\omega+1$  are countable ordinals. The ordinals extend indefinitely far beyond the countable ordinals. The Axiom of Choice in VBG ensures that every set is well-ordered (Hamilton, 1982, ch. 5). For every set  $S$  there is some ordinal  $\kappa$  such that  $\kappa$  is the order type of  $S$ . The ordinals in traditional set theory extend out to an inaccessible limit. It is possible to add that limit — the first inaccessible ordinal — to VBG and to define ever greater *large cardinals* that serve as increasingly powerful upper bounds on transfinite series (Drake, 1974).

---

<sup>1</sup>The set  $\{0, 1/2, 3/4, 7/8, \dots, 1\}$  is easily put into 1-1 correspondence with the well-ordered set  $\{1, 2, 3, \dots, 0\}$ ; but  $\{1, 2, 3, \dots, 0\}$  is not well-ordered by magnitude.

### 3. Acts and Situations

An *act* is simple: it is not decomposable into any combination of simpler acts. An act *produces* some *object*. Every act produces one and only one object (it is well-defined). So any well-ordered set of acts determines some unique well-ordered set of objects. A *task* is some well-ordered set of acts (hence of objects). Some tasks are machines.

Every task occurs in some universe  $(V, F)$  where  $V$  is some set of objects and  $F$  is some set of functions from  $V$  to  $V$ .  $V$  is the universe of objects and  $F$  is the universe of acts. For example: I refer to the classical computing systems defined by Turing as *Turing systems*. Some but not all Turing systems are *Turing machines* in the sense of "machine" defined here. For every Turing system, the universe  $V$  consists of triples of the form  $(t, h, s)$  where  $t$  is its tape,  $h$  is the position of its read-write head, and  $s$  is its internal state. For a Turing system, the universe  $F$  of acts consists of the functions read, write, left, right, change-state. Each function in  $F$  changes the state of the Turing system.

An act produces an object. *Situations* are containers for objects. If a machine produces object  $x$  then it produces the situation  $\{x\}$ . I add the set-theoretic wrapper to handle the degenerate cases in which machines fail to produce any object in their universe. If  $V$  is some set of objects, then the set of situations determined by  $V$  is the set of all  $\{v\}$  such that  $v$  is in  $V$ . I add the empty set to the set of situations determined by  $V$ . The empty set is an *empty situation*. If the action of a machine is not defined, it produces the empty situation  $\{\}$ . If the empty set itself is one of the objects that a machine produces, then the situation that contains that product is  $\{\{\}\}$ . If there is no danger of confusion, I let the object  $x$  stand for its situation  $\{x\}$  and I also let  $V$  stand for the set of situations it determines.

### 4. Tasks of Finite Order Type

A *machine* of finite order type is a task that satisfies (implements) an algorithm of finite order type. An *algorithm* of finite order type is a definition that involves three conditions: (1) an initial condition; (2) a successor condition; and (3) a final condition.

The *initial condition* of a finite algorithm specifies the initial situation for its machine. The initial situation has no predecessors. The machine  $S$  that satisfies the initial condition associates the initial ordinal 0 with the initial situation. I use the initial ordinal 0 to index the initial situation: the initial situation is  $S_0$ . For example: an algorithm for playing chess specifies that  $S_0$  is the standard starting arrangement of pieces on the chess board.

The *successor condition* of a finite algorithm specifies the successors of the initial situations. For any ordinal  $n$ , the successor condition defines a successor situation for the successor ordinal  $n+1$ . If  $S_n$  is some situation produced by a machine  $S$ , then the successor situation is  $S_{n+1}$ . If an algorithm defines situations  $S_0$  through  $S_n$ , and if the successor situation  $S_{n+1}$  does not depend in any way on the previous situations defined by the algorithm, then  $S_{n+1}$  is an entirely novel situation. It is the initial situation of a new algorithm. If we want to extend a task from an ordinal  $n$  to an ordinal  $n+1$  rather than start a new task, it is necessary to define the successor situation  $S_{n+1}$  in terms of previous situations. It is necessary to define an act  $A_n$  that *changes*  $S_n$  into  $S_{n+1}$ . The act  $A_n$  does not produce a new situation; it transforms an existing situation into another situation. Since  $S_{n+1}$  is defined in terms of  $S_n$ , or in terms of  $S_i$  for  $i < n+1$ ,  $S_{n+1}$  is defined *recursively*. The successor condition of any finite algorithm recursively defines  $S_{n+1}$  for all  $n$ .

The *final condition* of a finite algorithm specifies some condition in which a successor situation is final situation. The machine that implements the algorithm *halts* as it produces that final situation. Since every finite algorithm halts after some finite number of acts, the final condition specifies a final finite ordinal  $z$  such that  $S_z$  is the final situation of the machine. The task is *complete* with the production of the final situation  $S_z$ .

## 5. Machines Indexed by Finite Ordinals

For any well-ordered set  $S$ , there is some ordinal  $\kappa$  such that there is a function  $f$  from  $\kappa$  to  $S$ . The function *indexes*  $S$  with the members of  $\kappa$ . For example: the ordinal 5 is the set  $\{0, 1, 2, 3, 4\}$ ; the set  $S$  is  $\{a, b, c, d, e\}$ ; the function  $f$  pairs each member of 5 with some member of  $S$  like this:  $(0, a), (1, b), (2, c), (3, d), (4, e)$ . That function makes the series  $S_0 = a, S_1 = b, S_2 = c, S_3 = d, S_4 = e$ . That series is indexed by the ordinal 5.

Since every task is some well-ordered set of acts, and since every well-ordered set of acts uniquely determines some well-ordered set of situations, every task is equivalent to the well-ordered set of situations determined by its acts. For any well-ordered set  $S$ , there is some ordinal  $\kappa$  such that there is a function  $f$  from  $\kappa$  to  $S$ . So, for every task  $S$ , there is some ordinal  $\kappa$  such that there is a function  $f$  from  $\kappa$  to  $S$ . The set  $S$  indexed by  $\kappa$  is a series. Every machine is a task; it is a well-ordered set of situations. So, if machine  $M$  of finite order type operates in universe  $(V, F)$ , then  $M$  is a function from some finite ordinal  $\kappa$  to  $V$ . If a machine  $M$  is a function from an ordinal  $\kappa$  to  $V$ , then the order type of  $M$  is  $\kappa$ .

Every finite machine produces  $\kappa$  situations for some finite  $\kappa$ . The function  $M$  associates the initial ordinal 0 with the initial situation specified by  $M$ 's algorithm. If the function  $M$  associates  $n < \kappa - 1$  with some situation in  $V$ , then  $M$  recursively associates  $n + 1 \leq \kappa - 1$  with some successor situation in  $V$  according to  $M$ 's algorithm. The function  $M$  associates the final (greatest) ordinal  $\kappa - 1$  in  $\kappa$  with the final situation specified by  $M$ 's algorithm.

If some Turing system runs through  $n$  acts (with  $n$  finite) then that system is a function  $M$  from ordinal  $n$  to the universe  $V$  of Turing machine configurations (tape  $t$ , head position  $h$ , internal state  $s$ ). That function associates each ordinal  $k$  in  $\{0, 1, \dots, n\}$  with some Turing machine situation  $S_k$ . The initial configuration of the Turing machine  $S_0$  consists of the tape inscribed with an input. The successor configurations  $S_k$  for  $k < n$  are recursively defined by the acts of the system on its tape. The final configuration  $S_n$  consists of the tape inscribed with the system output, the final head position, and the halt state. Any Turing system that halts is a Turing machine, since it has a final configuration. Any Turing system that does not halt is not a Turing *machine* – it is just an incomplete series of acts.

## 6. Ill-Defined Supertasks on Incomplete Ordinals

Tasks of order type  $\omega$  ( $\omega$ -tasks) are sometimes called *supertasks*. Koetsier & Allis (1997: 295) define supertasks like this: "A supertask consists of a well-defined initial situation  $S_0$  and an infinite sequence of well-defined acts  $A_1, A_2, A_3$ , etc. that are such that for all natural numbers  $j$  the execution of  $A_j$  in the situation  $S_{j-1}$  is possible and leads to a well-defined situation  $S_j$ . During the execution an infinite sequence of situations  $S_1, S_2, S_3$ , etc. is

generated." There is an *enormous* difference between  $\omega$ -tasks and  $(\omega+1)$ -tasks. Benacerraf (1962: 772) refers to  $(\omega+1)$ -tasks as *super-duper-tasks*: "If a super-task is a task sequence of order type  $\omega$ , then a super-duper-task is the result of tacking an extra ( $\omega$ -th) task at the end of a super-task". The typical error of the supertask literature is to treat  $\omega$ -tasks as if they were  $(\omega+1)$ -tasks.<sup>2</sup> I distinguish between *well-defined* and *ill-defined* supertasks of any order type. I argue that while there are well-defined supertasks order type  $\omega+1$ , there are no well-defined supertasks of order type  $\omega$ . Supertasks of order type  $\omega$  are incomplete; their algorithms do not specify any final situation.

For example: consider the Thomson lamp (Thomson, 1954). Zeus has one of these lamps. Neither Zeus nor his lamp are bothered by the physical limitations that trouble mortals. The lamp is off at some time  $t=0$ . The lamp does not change by itself: if the lamp is off it stays off unless and until it is turned on; if the lamp is on it stays on unless and until it is turned off. Zeus turns the lamp on slowly. At first it takes Zeus  $1/2$  second to turn the lamp on. So for any time  $t$ , if  $0 \leq t < 1/2$ , the lamp is off; at time  $t=1/2$ , the lamp is on. Suppose the continuous interval  $[s, e)$  contains every point  $t$  such that  $s \leq t < e$ . Note that  $[s, e)$  does *not* contain the point  $e$ . So: during the time interval  $[0, 1/2)$  the lamp is off; at time  $t=1/2$ , the lamp is on. Zeus now turns the lamp off twice as fast as he turned it on before. During the interval  $[1/2, 3/4)$ , the lamp is on; at time  $t=3/4$ , the lamp is off. Zeus repeatedly switches the lamp on and off. Zeus *accelerates* his actions: if it took him  $x$  seconds to change the state of the lamp, then it takes him  $x/2$  seconds to change it again. If  $n$  is even, then at time  $t=(2^n-1)/2^n$ , the lamp is off; if  $n$  is odd, then at time  $t=(2^n-1)/2^n$ , the lamp is on. Suppose we say that the  $n$ -th *Z-point* is  $Z_n = (2^n-1)/2^n$ . So:  $Z_0$  is 0,  $Z_1$  is  $1/2$ ,  $Z_2$  is  $3/4$ , and so on. So the interval  $[0, 1/2)$  is the 0-th *Z-interval*  $[Z_0, Z_1)$ ; the  $n$ -th *Z-interval* is  $[Z_n, Z_{n+1})$ . If  $n$  is even, then the lamp is off during the  $n$ -th *Z-interval*; if  $n$  is odd, then the lamp is on during the  $n$ -th *Z-interval*. At time  $t=1$ , Zeus has performed the supertask that consists of changing the state of the lamp  $\omega$  times. What is the lamp state at time  $t=1$ ?

Benacerraf (1962) has shown that the lamp state at  $t=1$  is not determined by the switching supertask.<sup>3</sup> It is perfectly consistent with the performance (I do not say completion) of the supertask for the lamp to be either off, on, smashed, or whatever. There is no *continuity* between the lamp states for  $t<1$  and the lamp state for  $t=1$ . The lamp state at  $t=1$  is not related to any of its states before that time. The preceding series of acts has no effect on either Zeus's act at  $t=1$  or the state of the lamp at  $t=1$ . Whatever happened to the lamp before  $t=1$ , it makes no difference to the lamp at  $t=1$ . If 1 is the limit time of the supertask, then the supertask is *discontinuous at limits*. Whether or not the series of situations

---

<sup>2</sup>The typical error of the supertask literature is that the author defines the state of some system for all and only the finite ordinals but then poses a problem concerning the state of that system at the limit ordinal  $\omega$ . For example: the author defines the state of the system for all and only moments of time of the form  $(2^n-1)/2^n$  for all finite  $n$ ; then the author poses a problem concerning the state of the system at time 1. There is no problem: the state of the system at time 1 is not defined.

<sup>3</sup>Benacerraf describes a series of fair and foul numbers. Let  $f((2^n-1)/2^n)$  be fair if  $n$  is even and foul if  $n$  is odd. Function  $f$  is defined for every finite ordinal  $n$ . Function  $f$  associates every number in the set  $\{0, 1/2, 3/4, 7/8, \dots\}$  with the value fair or foul. Function  $f$  is not defined for the number 1; it does not tell us whether 1 is fair or foul. Function  $f$  does not associate every number in the set  $\{0, 1/2, 3/4, 7/8, \dots, 1\}$  with the value fair or foul. If Zeus computes  $f(n)$  at time  $t=(2^n-1)/2^n$ , then at time  $t=1$  Zeus is not computing.

defined by a  $\omega$ -supertask converges to a limit (as the Thomson lamp task does not), the  $\omega$ -supertask algorithm does not associate its limit time with that limit situation. Supertasks of order type  $\omega$  are incomplete by definition. Every  $\omega$ -supertask is discontinuous at limits. Since the  $\omega$ -supertask does not define any situation at  $t=1$ , it has not been completed at  $t=1$ . While a supertask compressed into Z-points is defined on the open interval  $[s, e)$ , it is *not* defined on the closed interval  $[s, e]$ . No matter what Zeus did during  $[s, e)$ , the result at  $t=1$  is that Zeus performed exactly one act that changed the lamp state from its initial off state to some final state. Since that act is not defined by the supertask, the supertask defines no action at all on  $[s, e]$ . Since the supertask has no outcome on  $[s, e]$ , and since some other act not defined by the supertask is performed during  $[s, e]$ , I conclude that Zeus in fact does not perform the supertask during  $[s, e]$ ; consequently, he does not perform it even during  $[s, e)$ . The definition of the  $\omega$ -supertask leads to the contradiction that it both has and has not been done in  $[s, e]$ . Since the argument applies to *any* task of order type  $\omega$  compressed by acceleration into Z-points, I say there are *no* well-defined  $\omega$ -supertasks.

Some series of situations  $\langle s_n \rangle$  is a  $\omega$ -series if and only if, for every finite  $n$ ,  $s_n = s(n)$  and it is a *finite* task to produce  $s(n)$ . Every  $\omega$ -series  $\langle s_n \rangle$  of situations lacks a final situation and so is incomplete. No  $\omega$ -series of situations (or acts) is well-defined. Since  $\omega$ -supertasks are not completable; they are ill-defined. Although there are  $\omega$ -series of acts (endless series of acts), there are no machines that produce  $\omega$ -series of situations. Every machine produces some final situation. No supertask of order type  $\omega$  is any machine.

## 7. Well-Defined Supertasks on Complete Ordinals

One way to avoid the difficulties associated with  $\omega$ -supertasks is to change the ill-defined  $\omega$ -supertask into a well-defined  $(\omega+1)$ -supertask by adding a *final condition* that associates some limit ordinal with some limit situation. The distinction between tasks of order type  $\omega$  and  $\omega+1$  is crucial for defining machines that *complete* infinitely many acts. Any machine that produces the limit of a series of situations is a machine that *completes* infinitely many acts. Such a machine is indexed by a complete ordinal (e.g.  $\omega+1$ ).

If  $\langle s_n \rangle$  is any  $\omega$ -series of situations, then the production of the whole series  $\langle s_n \rangle$  *and its limit* is a task of order type  $\omega+1$ , since it requires an additional act to make an additional situation. The production of the whole series  $\langle s_n \rangle$  is a production on the ordinal  $\{0, 1, 2, \dots\}$ , since it associates each  $n$  in that ordinal with some  $s_n$ . The production of the whole series  $\langle s_n \rangle$  *and its limit* is a production on the ordinal  $\{0, 1, 2, \dots, \omega\}$ , since it associates each  $n$  in that ordinal with some  $s_n$  and the limit ordinal  $\omega$  with the limit of the series  $\langle s_n \rangle$ . The act that defines the limit  $s_\omega$  in terms of the whole series  $\langle s_n \rangle$  is a limit act. It is a final act that leads to a final situation. *Infinite machines are continuous at limits.* They perform limit acts at limit ordinals and associate limit ordinals with limit situations.

Zeus loves to compute. He has an infinitely long tape divided into equal finite intervals (squares). The tape has a leftmost beginning but no finite rightmost end: if  $x$  is any square of the tape, then there is a square of equal finite size to the right of  $x$ . At first every square of the tape is inscribed with the number 0. Zeus associates the leftmost square with the

number 0; if Zeus associates some square with the number  $n$ , then he associates the next square to the right with the number  $n+1$ . While the total number of squares on Zeus's tape is  $\omega$ , there is no  $\omega$ -th square on the tape. Zeus uses his tape to determine the locations of the primes in the natural numbers. For any  $n$ , and for any finite interval of time, he is able to determine whether  $n$  is prime or not. Let  $\pi(n)$  be 1 if  $n$  is prime and 0 if  $n$  is not prime. At time  $t=0$ , Zeus places the initial square of the tape under his pen. At  $t=1/2$ , Zeus has written  $\pi(0) = 0$  in that square and pulled the tape leftwards one square. At  $t=(2^n-1)/2^n$ , he has written  $\pi(n)$  in square  $n$  and advanced the tape. Let the  $n$ -th state of Zeus's tape be  $p_n$ . The situation  $p_n$  is defined like this:  $p_n(i)$  is  $\pi(i)$  for  $0 \leq i \leq n$  and  $p_n(i)$  is 0 for  $n < i < \omega$ . For every  $Z$ -point in the interval  $[0, 1]$ , the tape associated with  $Z_n$  is  $p_n$ . Suppose the tape  $p_\omega$  is defined like this: for every  $i$ ,  $p_\omega(i)$  is  $\pi(i)$ . Suppose further that we treat these tapes as real numbers. Since there is no greatest prime, every  $p_n$  is less than every  $p_{n+1}$ , and every  $p_n$  is less than  $p_\omega$ . The series of tapes  $\langle p_n \rangle$  converges to  $p_\omega$ . For every real  $\epsilon > 0$ , there is an integer  $m$  such that  $|p_n - p_\omega| < \epsilon$  for all  $n \geq m$ . Tape  $p_\omega$  is the *limit* of the series  $\langle p_n \rangle$ . The production of  $p_\omega$  is a distinct act from the production of each  $p_n$ . For every  $n$ , there is some square on  $p_n$  beyond which the tape contains all 0s. Since there is no greatest prime, there is no such square on  $p_\omega$ . The tape  $p_\omega$  has at least one property that no  $p_n$  has. The property "tape  $x$  has 1 beyond every 1" is not continuous at limits; that property appears discontinuously at  $t=1$  for  $p_\omega$ . So the production of the tape  $p_\omega$  with that novel property is a distinct act. If Zeus's action is *continuous at limits*, then he produces the successor tape  $p_n$  at every time  $t=Z_n$ , and he *additionally* produces the limit tape  $p_\omega$  at time  $t=1$ . If Zeus's action is continuous at limits, then he performs a  $(\omega+1)$ -supertask.

## 8. Machines Continuous At Limit Ordinals

Some supertasks of order type  $\omega+1$  are well-defined. Those supertasks involve producing limits. Such supertasks are *continuous at limit ordinals*. Such supertasks associate every item in  $\{0, 1, 2, \dots, \omega\}$  with some situation. One way to make such an association is specify conditions for the three kinds of ordinals: (1) the initial ordinal 0; (2) the successor ordinals; and (3) the limit ordinal  $\omega$ . The limit ordinal  $\omega$  is the final ordinal of  $\omega+1$ .

A *machine* of order type  $\omega+1$  is a task that satisfies (implements) an algorithm of order type  $\omega+1$ . An *algorithm* of order type  $\omega+1$  is a definition that involves (1) some series of  $Z$ -points; (2) some function  $f$ ; (3) some limit operation  $L$ ; and (4) some condition for each type of ordinal. The series of  $Z$ -points occurs within some interval  $[s, e]$ . The function  $f$  is defined for every finite  $n$  and associates each finite  $n$  with some situation  $f(n)$ . For every finite  $n$ , the production of  $f(n)$  is a task of finite order type. The limit operation  $L$  defines a limit situation in terms of all preceding finite situations. There are three kinds of ordinal conditions: (1) an initial condition; (2) a successor condition; and (3) a final condition. The final condition is a limit condition: it associates  $\omega$  with the limit situation.

The *initial condition* of an algorithm of order type  $\omega+1$  on some  $Z$ -points associates  $Z_0$  is associated with the situation that contains  $f(0)$ . The *successor condition* for a supertask on some  $Z$ -points has this form: for any finite  $n$ , if the  $Z$ -point  $Z_n$  is associated with the situation that contains  $f(n)$ , then the  $Z$ -point  $Z_{n+1}$  is associated with the situation that contains  $f(n+1)$ . The *final condition* for a supertask on some  $Z$ -points depends on whether

or not the limit of the  $f(n)$  exists. Say  $\langle f_n \rangle$  is the series of  $f(n)$ . If the limit of the series  $\langle f_n \rangle$  as  $n \rightarrow \omega$  exists (if the series converges to some object  $f_\omega$ ), then the limit Z-point  $Z_\omega$  is associated with the situation that contain  $f_\omega$ ; if the limit of the series does not exist (if the series diverges), then  $Z_\omega$  is associated with the empty set  $\{\}$ . The empty set indicates that no situation is produced at  $Z_\omega$ . The limit  $Z_\omega$  in  $[s, e]$  is the end point  $e$ .

If the limit of  $\langle f_n \rangle$  exists, then it is continuous with the series  $\langle f_n \rangle$ . I treat continuity as increasing resemblance. The idea is that as  $n \rightarrow \omega$ , the *difference* between the  $f_n$  and the limit situation becomes arbitrarily small. Two familiar species of limits are *numerical limits* and *set-theoretic limits*. Numerical limits are defined using the familiar Weierstrass  $\epsilon$ - $\delta$  theory of limits:  $\lim_{n \rightarrow \omega} f_n$ . The numerical limit of  $\{0, 1/2, 3/4, 7/8, \dots\}$  is 1. Set-theoretic limits are defined in terms of infinite unions:  $\bigcup_{n < \omega} S_n$ . For example: the set-theoretic limit of the von Neumann ordinals  $\{\}$ ,  $\{\{\}\}$ ,  $\{\{\{\}\}\}$ , and so on is the von Neumann  $\omega$ . The finite von Neumann ordinals are continuous with the von Neumann  $\omega$ . Figure 1 shows numerical and set-theoretic supertasks.

Since machines that perform supertasks of order type  $\omega+1$  are able to perform countably infinitely many tasks, I refer to them as  $\aleph_0$ -machines. Pitowsky (1990) calls them "Platonist computers". If  $\aleph_0$ -machine  $M$  operates in universe  $(V, F)$ , then  $M$  is a function from some countable ordinal  $\kappa$  to  $V$ . I do not doubt that  $\aleph_0$ -machines exist in the Platonic world of mathematical objects. I agree with Earman & Norton (1993) and Hogarth (1994) that the existence of  $\aleph_0$ -machines in any physical universe is a contingent proposition. If the system of possible physical universes is logically complete, some of them contain  $\aleph_0$ -machines. If there is no finite upper bound on the speed with which tasks are performed in some physical universe, and if there are no difficulties from spatial or causal features of that universe, then  $(\omega+1)$ -supertasks can be performed in finite temporal intervals in that universe. I doubt that  $\aleph_0$ -machines exist in our physical universe.<sup>4</sup>

#### Numerical Algorithm

$$S_0 = 0;$$

$$S_n = \sum_{k=1}^n \frac{1}{2^k}; \text{ for successor } n$$

$$S_\omega = \lim_{n \rightarrow \omega} S_n; \text{ for limit } \omega$$

#### Set-Theoretic Algorithm

$$V_0 = \{\};$$

$$V_n = \text{pow}(V_{n-1}); \text{ for successor } n$$

$$V_\omega = \bigcup_{n < \omega} V_n; \text{ for limit } \omega$$

**Figure 1.** Numerical and set-theoretic algorithms.

## 9. Examples of Machines Continuous At Limit Ordinals

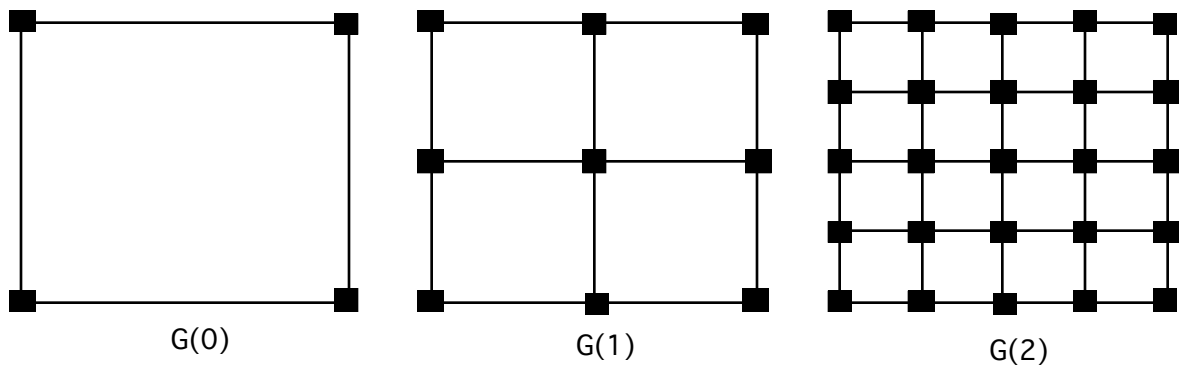
---

<sup>4</sup>It seems that all machines in our universe are of finite order type. I am not talking about artifacts: any sequence of natural changes is a machine. It seems that the speed of light is a finite upper bound on change; space and time seem to have finite lower bounds (the Planck length and the Planck time); physical quantities and causal laws seem quantized.

There is a long history of *accelerating systems* that perform an initial act in  $1/2$  time-unit (a second, hour, day), the next act in  $1/4$  time-unit, and so on (Weyl, 1963: 42; Grunbaum, 1969; Boolos & Jeffrey, 1980; Copeland, 1998a). The limit situations of those systems are typically not precisely defined but are merely assumed to exist in accordance with vague continuity principles. The many paradoxes associated with these systems in the supertask literature bear witness to the failure to take mathematical care with the infinite.

An endless series of acts produces a series of situations of order type  $\omega$ , but does not produce any final situation. If an *accelerating Turing system* only associates times  $0, 1/2, 3/4, 7/8$  and so on with situations, but does not associate the limit  $1$  of that series with any situation, then it is not a machine at all — it is merely an endless series of acts; it is an incomplete and therefore ill-defined series of acts. Every Turing *machine* halts at some final ordinal, either a successor or a limit ordinal; if some Turing *system* does not halt, it is not a Turing machine. An *accelerating Turing machine* compresses infinitely many operations into the  $Z$ -points of some finite temporal interval  $[s, e]$  by (1) associating  $Z_0 = s$  with some initial situation; (2) associating the  $Z_n$  for any finite  $n$  with some successor situation; and (3) associating  $Z_\omega = e$  with the some precisely defined final limit situation. McCarthy & Shapiro (1987), Copeland (1998b), and Hamkins & Lewis (forthcoming) precisely define limit situations for accelerating Turing machines. Accelerated Turing machines are more powerful than finite Turing machines. They can compute the halting function for finite Turing machines; they can therefore compute Rado numbers.

I describe a set-theoretic  $\aleph_0$ -machine that produces an infinitely subdivided 2D grid for a cellular automaton. The initial situation  $G(0)$  is produced at  $Z$ -point  $Z_0$ . For any square of cells in  $G(n)$  at time  $Z_n$  for any finite  $n$ , the insertion operation does two acts: (1) it puts 4 cells between the corners of that square and in the center of that square; and (2) it adds links from all the newly inserted cells to their previously or newly inserted neighbors. The machine therefore produces the successor situation  $G(n+1)$  at time  $Z_{n+1}$ . Figure 2 shows the changes from  $G(0)$  to  $G(1)$  to  $G(2)$ . Note that diagonal links are not shown in Figure 2. When it is inserted, each cell gains 8 neighbors. The limit situation  $G(\omega)$  is the union of all the  $G(n)$  for finite  $n$ .<sup>5</sup> The machine produces  $G(\omega)$  at the limit time  $Z_\omega$ . The situation  $G(\omega)$  is an infinitely subdivided 2D cellular space.



**Figure 2.** Insertions of cells in a 2D space.

<sup>5</sup>The union of all the  $G(n)$  is precisely defined set-theoretically. Each  $G(n)$  for finite  $n$  is a graph: it is a pair  $(V(n), R(n))$  where  $V(n)$  is a set of cells and  $R(n)$  is the neighbor relation.  $G(\omega)$  is  $(V(\omega), R(\omega))$  where  $V(\omega)$  and  $R(\omega)$  are the respective infinite unions.

## 10. Supertask Repetitions of Supertasks

Supertasks performed in some finite time divide the finite interval  $[s, e]$  into Z-points. For instance: the Z-points of  $[0, 1]$  are the familiar  $0, 1/2, 3/4, 7/8$  and so on. Every adjacent pair of Z-points on  $[s, e]$  determines a subinterval  $(Z_n, Z_{n+1}]$  of  $[s, e]$ . The subinterval does not contain the point  $Z_n$  since it was the final point of the previous subinterval. So:  $(0, 1/2], (1/2, 3/4]$ , and so on are subintervals of  $[0, 1]$ . Any subinterval  $(Z_n, Z_{n+1}]$  is divisible into its own series of internal Z-points. The Z-points in  $(1/2, 3/4]$  are  $5/8, 11/16, 23/32, 47/64$ , and so on. If it is possible to perform a  $(\omega+1)$ -supertask on some  $[s, e]$ , then it is possible to perform such a  $(\omega+1)$ -supertask on any  $(Z_n, Z_{n+1}]$  in  $[s, e]$ .

For instance: if A and B are supertasks, perform A on the interval  $(0, 1/2]$  and perform B on the interval  $(3/4, 1]$ . We can't perform the second supertask on  $[1/2, 1]$  because the Z-point  $1/2$  is already associated with the final situation of the first supertask A. So we have to skip to the first Z-point of the next subinterval. It is straightforward to repeat this pattern for any finite number of supertasks. The machine that performs the 2 supertasks A and B is a function from the ordinal  $\{0, 1, 2, \dots, \omega, \omega+1, \omega+2, \dots, \omega+\omega\}$ . The ordinal  $\omega+\omega$  is  $\omega*2$ . Likewise: n supertasks are performed on  $\omega*n$ . The machine M that performs some finite number n of supertasks is a function from  $\omega*n$  to some set V of situations.

There are  $\aleph_0$  pairs of adjacent Z-points in any subdivided  $[s, e]$ ; so, if  $\kappa$  is any countable ordinal, then is possible to perform  $\kappa$  supertasks by nesting them in the subintervals of  $[s, e]$ . If it is (physically) possible to perform any supertask on the Z-points of some finite interval  $[s, e]$ , and if  $\langle A_n \mid n < \omega \rangle$  is any series of well-defined supertasks, then it is (physically) possible to perform every supertask in  $\langle A_n \rangle$  on the nested subintervals of  $[s, e]$ . Importantly, since a  $\omega$ -series of supertasks is ill-defined, we need to associate the final point e of  $[s, e]$  with the limit of all the supertasks. Suppose it is a supertask to produce the situation  $S_n$  for any finite n. Here is an algorithm for taking the limit of infinitely many supertasks: (1) associate 0 with  $S_0$ ; (2) for any n, associate  $Z_n$  with  $S_n$ ; (3) associate the final ordinal  $\omega*\omega$  with the limit of  $S_n$  as  $n \rightarrow \omega$  if it exists or with  $\{\}$  if it does not exist.

If M is a machine that performs many supertasks, then its algorithm has 4 conditions: (1) an initial condition; (2) a successor condition; (3) a limit condition; (4) a final condition. For  $(\omega+1)$ -supertasks, the limit and final conditions are identical (since the limit ordinal is the final ordinal of the  $(\omega+1)$ -series). However, if M performs n supertasks, then M iterates past n limit ordinals. The limit condition of M's algorithm defines the situation produced by M for limit ordinals generally. For example: if  $A_{i,n}$  is the i-th task in the n-th supertask, then associate  $S_{\omega*n}$  with  $\lim_{i \rightarrow \omega} A_{i,n}$  for all finite n;  $S_{\omega*\omega} = \lim_{n \rightarrow \omega} S_{\omega*n}$ . If every set is well-ordered, then is no reason to stop with the countable ordinals. There are machines on all complete countably infinite ordinals. Since there are well-orderings of uncountably infinite ordinals, there are machines on all complete ordinals (e.g.  $\omega_1+1$ ). These machines are classified by cardinals:  $\aleph_1$ -machines,  $\aleph_2$ -machines,  $\dots$   $\aleph_\omega$ -machines.

## 11. Generalized Supertasks on Complete Ordinals

A *machine*  $M$  is a function from some ordinal  $\kappa+1$  to the universe  $V$  of sets such that  $M$  satisfies an algorithm. An *algorithm* is a rule that associates every  $\alpha \leq \kappa$  with some set  $V_\alpha$  in accordance with certain conditions. The conditions are spelled out for (1) the *initial* ordinal 0; (2) *successor* ordinals  $< \kappa$ ; (3) *limit* ordinals  $< \kappa$ ; and (4) the *final* ordinal  $\kappa$ .

A machine performs some task according to some algorithm. A *task* is some well-ordered set (some series) of *acts*. An act *produces* some *situation*. The machine starts with an initial situation (its input). The *initial condition* of the machine's algorithm associates the initial ordinal 0 with the initial situation  $S_0$ . The machine performs act  $A_0$ . Act  $A_0$  changes situation  $S_0$  into situation  $S_1$ . I say the machine *produces* situation  $S_1$ . If the machine has performed  $n$  acts for some ordinal  $n$ , so that it has produced situation  $S_n$ , then it performs the next act and thereby produces the situation  $S_{n+1}$ . For every ordinal  $n$ , the *successor condition* of the machine defines an act that changes the situation  $S_n$  into the situation  $S_{n+1}$ . So the machine associates every successor ordinal  $n$  with some situation  $S_n$ . If the machine has performed every act  $A_n$  for every ordinal  $n$  less than some limit ordinal  $\lambda$ , then the machine performs an act  $A_\lambda$  for the limit ordinal  $\lambda$ . The *limit condition* of the machine's algorithm associates the limit ordinal  $\lambda$  with situation  $S_\lambda$ . The machine at last performs a final act  $A_\kappa$ . The final act  $A_\kappa$  associates the final ordinal  $\kappa$  with some situation  $S_\kappa$ . The *final condition* of the machine's algorithm specifies the final act.

For example: if  $\kappa$  is any complete ordinal, then there is some machine that produces the iterative hierarchy of sets on  $\kappa$ . If  $\kappa$  is an inaccessible cardinal, then the machine that produces  $V_\kappa$  produces a model of ZFC set-theory. Since the class of ordinals  $On$  is not complete (it contains no greatest ordinal), there is no machine that produces the iterative hierarchy of sets. Figure 2 shows the algorithms for the ZFC and the whole hierarchy.

## 12. Conclusion

I have defined algorithms and machines generally in terms taken from pure mathematics. I defined an algorithm as a conjunction of rules for (1) the initial ordinal 0; (2) the successor ordinals; (3) the limit ordinals; and (4) some final ordinal. I defined a machine as a function from some ordinal  $\kappa$  to some set of situations  $V$  that satisfies an algorithm. The machine is a model of the algorithm in the sense that the algorithm's condition on any  $\alpha$  in  $\kappa$  is true of the situation that the machine associates with  $\alpha$ . I discussed machines that perform finite tasks and machines that perform transfinite supertasks. Every machine is effective. While the physical limits on effectiveness are set by the spatio-temporal-causal structure of each universe, I see no mathematical limit to the concept of effectiveness.

## References

- Benacerraf, P. (1962) Tasks, super-tasks, and the modern Eleatics. *Journal of Philosophy* 59 (24), 765 - 784.
- Boolos, G. & Jeffrey, R. (1980) *Computability and Logic*. 2nd Ed. New York: Cambridge University Press.
- Copeland, B. J. (1998a) Super Turing-machines. *Complexity* 4 (1), 30 - 32.
- Copeland, B. J. (1998b) Even Turing machines can compute uncomputable functions. In C. Calude, J. Casti, and M. Dinneen (Eds.), *Unconventional models of Computation*. New York: Springer-Verlag.
- Drake, F. (1974) *Set Theory: An Introduction to Large Cardinals*. New York: American Elsevier.
- Earman, J. & Norton, J. (1993) Forever is a day: Supertasks in Pitowsky and Malament-Hogarth spacetimes. *Philosophy of Science* 60, 22 - 42.
- Grunbaum, A. (1969) Can an infinitude of operations be performed in a finite time? *British Journal of the Philosophy of Science* 20, 203 - 218.
- Koetsier, T. & Allis, V. (1997) Assaying supertasks. *Logique et Analyse* 159, 291 - 313.
- Hamilton, A. (1982) *Numbers, Sets, and Axioms: The Apparatus of Mathematics*. New York: Cambridge University Press.
- Hamkins, J. & Lewis, A. (Forthcoming) Infinite time Turing machines. *Journal of Symbolic Logic*.
- Hogarth, M. (1994) Non-Turing computers and non-Turing computability. *Philosophy of Science Association 1994* (i), 126 - 138.
- McCarthy, T. & Shapiro, S. (1987) Turing projectability. *Notre Dame Journal of Formal Logic* 28 (4), 520 - 535.
- Pitowsky, I. (1990) The physical Church Thesis and physical computational complexity. *Iyyun* 39, 81 - 99.
- Thomson, J. (1954) Tasks and supertasks. *Analysis* 15, 1 - 13.
- Weyl, H. (1963) *Philosophy of Mathematics and Natural Science*. New York: Atheneum. (Original work in German 1927).