

Chapter 2

Every physical system in the universe, from wheeling galaxies to bumping proteins, is a special purpose computer in the sense that every physical system in the universe is implementing some computation or other.

E.Dietrich, *Thinking Computers and Virtual Persons*.

Computationalism

2.1. Overview

In this chapter the concept of computation is examined. This is necessary in order to make explicit the assumptions underlying computationalism, the metaphysical basis of the unified framework of emergent artificiality described in chapter 5. The idea of a formal system provides the starting point for the investigation given the initial identification of computation with symbol manipulation. This leads to the notion of an 'effective procedure' which is formalized in the mathematical concept of a Turing machine. The concept of a Universal Turing machine (UTM) and the Church-Turing Thesis are introduced as the basis of computationalism. Various concepts of computation are examined and computationalism is defined. The issue of connectionism is briefly considered and arguments are presented in favour of realizing UTMs in discrete, massively-parallel, locally-interacting media known as cellular automata. This leads to two related concepts, viz. Universal Cellular Automata and Digital Mechanics. The relation between computationalism and the physical world is examined in preparation for the presentation of the unifying framework of computationally emergent artificiality described in chapter 5. Finally, in preparation for the critique of computationalism presented in Part III of this study, the computer metaphor is shown to be reducible to a set of root metaphors and the link between computationalism and process philosophy is briefly examined.

2.2. What is *Computation* ?

The *McGraw-Hill Dictionary of Scientific and Technical Terms (Fifth Edition)* defines computation as "the act or process of calculating; the result so obtained." However, Feigenbaum et al. (1983) maintain that "the computer [is] badly misnamed. 'Computer' implies only counting and calculating, whereas this unpromising hunk of wires, tubes, switches, and lights [is], in principle, capable of manipulating any sort of symbol." (p.37)

Boden (1977) argues similarly: "computers do not crunch numbers [that is, calculate]; they manipulate symbols", whereby symbol is meant "a meaningless cipher that becomes meaningful by having meaning assigned to it by a user." (p.15) This identification of computation with meaningless symbol manipulation leads directly to the notion of a formal system.

2.3. Formal Systems

In this section the definition, interpretation and properties of a formal system are briefly described.

2.3.1. Definition

A formal system is a rigorously defined, that is, unambiguous, system of symbols and rules for forming and manipulating symbol structures in which only the form or 'shape' of the symbols and their combination in symbol structures is considered; possible content or meaning associated with the symbols - whether intrinsic or extrinsic - is ignored. For this reason, formal systems are often referred to as *token* systems (Moody,93), emphasizing the purely *syntactical* (formal or structural) and *semantically*-independent (meaning-less) nature of the primitives in such systems.

A *formal system* comprises

- an *alphabet* composed of a set of distinct (discrete) primitive symbols. A finite sequence of symbols is called a symbol structure or *string*
- a *grammar* or set of formation rules defining how symbols in the alphabet may be combined to generate well-formed formulae (wffs), that is, legal strings
- a set of *axioms* or legal strings (wffs) which are given *a priori*
- a set of *inference* or transformation *rules* defining how further legal strings (wffs) may be generated from existing strings

A *theorem* is a legal string (wff) capable of being produced by a finite sequence of applications of the inference rules to the axioms. The sequence of application of the rules constitutes the *proof* or derivation of the theorem.

For example, consider the formal system with

alphabet	{A,B}
grammar	all strings <i>s</i> are wffs
axiom	A

inference rules	$A \rightarrow AB$
	$sB \rightarrow sBB$
	$As \rightarrow AAs$

The operator ' \rightarrow ' denotes 'is replaced by' and indicates the application of an inference rule to an axiom or theorem. On this scheme, AAABB is *both* a legal string *and* a theorem. However, ABAB is a legal string but *not* a theorem since it is not provable from *within* ('inside') the system, that is, there is no proof sequence that leads to the production of this string. Thus, inference rules introduce a level of constraint in addition to that provided by grammar.

2.3.2. Interpretation of Formal Systems

According to Garnham (1988), "rules [in a formal computational system] make *direct* reference only to formal properties of symbols, but the choice of rules depends on what the symbols they manipulate stand for." (p.229) Consequently, a formal system requires an *interpretation* which is achieved by assigning meaning to the tokens of the system. On this basis, tokens are rendered symbolic in a Peircean semiotic or significative sense, viz. a symbol (or sign) is something which stands as something (else) in some respect or other for somebody (Fetzer,90). There is a connection between the symbol and that to which it refers, the latter providing the symbol with its meaning; hence, the implication that meaning (semantics) is *imparted to* formal systems (syntax) exosystemically, that is, from *without* ('outside') the system. For this reason, meaning is held to be *extrinsic* to formal systems. However, this position has been contested: For example, Pylyshyn (1980) maintains that semantics can effectively be reduced to syntax, viz. "all relevant semantic distinctions [must] be mirrored by syntactic distinctions - i.e. by features intrinsic to the representation itself." (p.113). On this view, semantics is *intrinsic* to a formal system because semantics is reflected in or *reducible to* syntax and formal systems are syntactical entities. As will be seen in chapter 3, the debate over intrinsic vs. extrinsic *semantics* has implications for the debate over intrinsic vs. extrinsic *emergence* as a consequence of the link between semantics and observation on the one hand, and observation and emergence on the other (Cariani,89) (Cariani,91).

An interpretation that satisfies (makes true) the axioms and theorems of the system is called a *model*. Formal systems are of value because, under suitable interpretation, isomorphisms, that is, one-one mappings or correspondences, may be established between primitives in a formal system (axioms, rules of inference) and primitives in a natural system (states, natural laws). This leads to a variant of the modelling relation due to Newton¹ (Fig 2.1) and briefly described as follows:

¹ According to Casti (1989), this modelling relation is *implicit* in Newtonian mechanics. However, Cariani (1989) traces the *explicit* introduction of this model to Helmholtz, Hertz and Mach in the 19th Century and its recent formulation to Rosen.

1. Using the interpretation, encode states of the world and causal laws into axioms and rules of inference in the formal system.
2. Derive theorems in the formal system.
3. Using the interpretation, decode theorems back into states of the world to which they correspond.

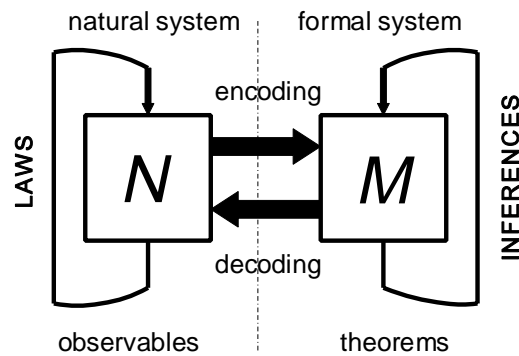


Fig 2.1 Newtonian modelling relation.

The problem is to discover rich and true isomorphisms between parts (or the whole) of reality and formal systems². However, it must be recognized from the outset that in order for this to be possible it is necessary to adopt a specific epistemology and ontology (metaphysics) with respect to the world which entails viewing the latter in terms of a set of context-free, determinate facts or 'atoms' in specific relations to one another (Dreyfus,93). The precise organization (structure or pattern) of these atoms at any instant constitutes the global system state at that instant (Elstob,84). Rosen (1988) maintains that the encoding-decoding operations must be non-formal since they involve mappings from continuous (or analog) to discrete (or digital) domains and visa-versa. This is significant because in a formal system, the alphabet is fixed; although possibly infinitely many strings can be produced by applying the inference rules to previously generated strings (theorems), the set of system primitives is itself 'closed', that is, finite and static. However, robotic-functionalist systems are able to augment (extend) or alternatively substitute (replace) one alphabet for another using measurement processes to generate new alphabetical primitives or symbols (Cariani,89). Such systems are *hybrid* analog-digital devices, that is, systems which include non-discrete components that produce symbols via a grounding in the external physical world (Harnad,90). It is crucial to appreciate that a formal system - *as a formal system* - exists (or rather, *subsists*) in the abstract, disembodied, and static sense of a Platonic form (section 2.7.2). Furthermore,

² *Richness* and *truth* are here viewed epistemologically: The former is associated with *prediction* (or anticipation) and the latter with *explanation* (or understanding).

all possible theorems for a formal system exist (or subsist) - as *potentialities* requiring *actualization* or embodiment in a suitable medium or substrate³ - once its alphabet, grammar, axioms and inference rules have been specified since the latter are all fixed (static). In this connection, it is interesting to note with Simon (1981) that

all mathematics exhibits in its conclusions only what is already implicit in its premises .. Hence all mathematical derivation can be viewed simply as change in representation, making evident what was previously true but obscure. (p.153)

This point is extremely significant to the debate on computational emergence which is discussed in chapters 5, 6 and 7. Since hybrid devices are not digital computers - although they belong to the class of machines with discrete computational elements (Cariani,89) - they will not be considered further in this study⁴.

2.3.3. Properties of Formal Systems

There are two properties of a formal system which arise in relation to interpretation:

□ *Consistency*

every axiom and theorem of a formal system upon interpretation in some possible world is a truth of that world, that is, there exists at least one non-contradictory interpretation in which a string and its negation are not both provable, that is, theorems.

□ *Completeness*

there exists a possible world for which all its truths are, under a suitable interpretation, axioms or theorems of the formal system. This may also be stated as the requirement that all 'admissible' strings be provable. Thus, the formal system described in section 2.3.1 is incomplete under the interpretation that all strings are wffs; however, if another interpretation whereby this is not the case (that is, ABAB is not a wff) can be found, then the system will be complete with respect to that interpretation.

However, Gödel has shown that for any formal system F that is finitely describable,

³ In chapter 7, it will be shown that formal and computational systems, as instances of "hard" artificiality (as artifactuality) are embodied - as *idealizations* - in the mind of the artificer-interpreter.

⁴ As will be seen in what follows, the relevance of hybrid devices to the debate on computationalism is questionable since their capacity to provide a framework within which to address some of the problems associated with the latter rests on the assumption that the physical world is *ontologically* continuous, the position *ultimately* contested by proponents of the computationalist thesis (section 2.6.6).

consistent and powerful enough to prove the basic facts about elementary arithmetic,

- I. F is incomplete, and
- II. F cannot prove its own consistency.

As Tipler (1995) states

Any formal system powerful enough to express the axioms of arithmetic contains a self-referential statement equivalent to 'This statement is unprovable'. If it is true, then the statement itself is unprovable, and arithmetic is incomplete .. On the other hand, if the statement is false, then, since it is equivalent to a statement of arithmetic, arithmetic would be logically inconsistent. (p.25)

Hence, any such system F will necessarily be either (i) incomplete and consistent, yet unable to prove its consistency, or (ii) inconsistent. Gödel's theorems, which constrain the axiomatization of the world, are limitations only on the extent to which a descriptive law for *every* phenomenon may be constructed (Casti,89). However, they become extremely relevant to the debate on computationalism since the latter in its "strong" version is a formistic *ontological* thesis (section 2.7.2) and hence, must account for everything in terms of mechanical (law-like) transformations.

2.4. Computation

The *Microsoft Press Computer Dictionary (Second Edition)* defines a computer as

any machine that does three things: accepts structured input [data], processes it according to prescribed rules [a program], and produces the result as output.

Computers are essentially of two kinds: analog and digital. Analog computers are continuous devices while digital computers are discrete devices. The metaphysical position defined and examined in this thesis, viz. computationalism, is grounded in the latter, viz. discrete devices; hence, the grounding relation between the *digital* computer metaphor and computationalism (section 2.7). Casti (1989) defines a computer as

a machine for transforming one set of meaningless symbols into another; in short, a device for physically executing the operations called for by the rules of a *formal logical system*. (p.268)

However, the concept of a computer as a symbol-manipulating machine remains problematic since it is unclear what is meant by a *machine*. Kelly (1993) identifies the following characteristics as fundamental to the notion of a machine: function, design, determinism, explicitness, automaticity and mediacy. He further argues that "any dilution of these properties would constitute a departure from a historical understanding of the nature of machines and invite a charge of redefinition." (p.4) Yet, according to Toulmin (1993), this redefinition has already taken place as a result of

the evolution of the modern scientific world view [which] transformed the concepts of 'machine' and

`mechanism' to a point at which their originators would scarcely recognize them. (p.140)

Toulmin maintains that in seventeenth century thought, a machine was regarded as "an instrument for transmitting outside action". The notion of a `living' or `thinking machine' was ruled out not because of anything in the empirical content of the science of the period, but as a consequence of the *canonical* definitions of matter and machine respectively. For this reason, he holds that

if Descartes, Newton or Leibniz had been shown a late 20th century computer, they could only have reacted by declaring "That's not a `machine' at all!" (p.146)

However, certain key ideas have been retained in the modern concept of a machine. For example, Newell (1980) defines a machine as

a system that has a specific determined behaviour as a function of its input. By definition, therefore, it is not possible for a single machine to obtain even *two* different behaviours, much less any behaviour." (p.148)

According to this view, function and determinism are held to be characteristic of modern machines including computers which leads directly to the notion of an effective procedure.

2.4.1. Effective Procedures

Garnham (1988) defines an *effective procedure* as

one that could be carried out *automatically* by a machine, with no human intervention - one whose result, when computed by a person, does not depend on `intuition' or any other process not open to objective inspection [emphasis added]. (p.225)

However, the notion of `machine' in the above definition remains somewhat vague, thereby engendering arguments over what is to be taken as constituting the *abstract essence* (or universal form) of an effective procedure. In order to resolve this problem, it is necessary to redefine the notion of an effective (or mechanical) procedure in terms of a machine designed specifically for this purpose. This leads directly to the concept of a Turing machine.

2.4.2. Turing Machines

A *Turing machine* (Turing,36) is a formal model of an effective procedure. Turing machines (TMs) possess the two essential properties of any model of an effective procedure, viz. (i) each procedure is *finitely describable*, and (ii) each procedure consists of *discrete steps*, each of which could be carried out *mechanically*, that is, without what Minsky has referred to as `innovation' or `intelligence' (Minsky,67). The following description of TMs is taken from Hopcroft and Ullman (1979):

A TM consists of a finite control, an input tape that is divided into cells, and a tape head that scans and prints (that is, reads from and writes to) one cell of the tape at a time (Fig 2.2). The tape has a leftmost cell but is *infinite* to the right. Each cell of the tape can hold exactly one of a finite number of tape symbols. Initially, the n left-most cells, for some finite $n \geq 0$, hold the input, which is a string of symbols chosen from a subset of the tape symbols called the input symbols. The remaining infinity of cells each hold a blank, which is a special tape symbol that is not an input symbol.

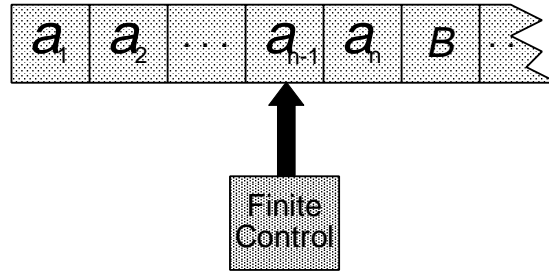


Fig 2.2 A Turing machine.

On each move and depending on the symbol in the cell scanned by (read from) the tape head and the state of the finite control, the TM performs the following operations, viz. changes state, prints (writes) a symbol into the cell, replacing what was printed there, moves its head left or right one cell position or halts (that is, remains in the current state).

A Turing machine (TM) is formally defined as follows:

$$TM = (Q, \Sigma, \Gamma, \delta, q_0, B, F),$$

where

- Q is a finite set of *states*,
 - Γ is a finite set of allowable *tape symbols*,
 - B a symbol of Γ , is the *blank*,
 - Σ a subset of Γ not including B , is the set of *input symbols*,
 - δ is a next-move function, a mapping from $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
(δ may, however, be undefined for some arguments),
 - q_0 in Q is a *start state*,
 - $F \subseteq Q$ is a set of *final states*.
-

2.4.3. Extensions to Turing Machines

Various extensions to TMs such as

- a bidirectional infinite tape
- multiple tapes
- nondeterministic operation
- multidimensionality of tape
- multiple tape heads

result in a machine which can be shown to be equivalent (in computational power) to the standard TM (Minsky,67). TMs are equivalent in computing power to digital computers and to the most powerful mathematical notions of computation (Hopcroft,79); consequently, the TM has become the accepted formalization of an effective procedure.

2.4.4. Computability and Decidability

Computability is usually defined with respect to TMs. For example, Minsky (1967) states that

a function $f(x)$ will be said to be Turing-computable if its values can be computed by some Turing machine T_f whose tape is initially blank except for some standard representation of the argument x . The value of $f(x)$ is what remains on the tape when the machine stops. (p.135)

Informally, a computable function is one which can be generated by a finitely defined mechanical procedure; conversely, an uncomputable function is one which cannot be so generated, even by executing an infinite number of steps. For example, the number pi is computable, even though its expansion is infinite and seemingly random, since the procedure or algorithm for determining its expansion (to any desired number of decimal places) is finite. Thus, the notion of an effective procedure leads to the concept of a TM and the latter defines what is meant by computability.

The notion of decidability is related to computability via a question: Is there a *general* procedure (that is, algorithm or TM) for determining in advance (deciding) whether or not a *particular* program (TM) will halt after a finite number of steps ? Turing showed that no such procedure exists (or *can* exist), that is, given a TM T_f and a tape with an input data set I , there is no way in general to say if T_f will ever finish processing I . This 'Halting Problem' is essentially Gödel's Incompleteness Theorem (section 2.3.3) as applied to TMs. What is significant in the context of this study is the question of which processes are Turing-computable and which (if any) are not; clearly, the possibility of *conceiving* the existence of uncomputables *as such* does not entail the non-existence of computational analogues of natural phenomena such as matter, life and mind (chapters

4 and 5)⁵. Furthermore, according to Cariani (1989), the question of computability only arises if the tapes of a TM are allowed to be *infinite* in length; TMs with finite tapes are equivalent to finite state machines (FSMs). On this basis and given the Finite Automaton (section 2.5.3) and 'finite nature' (section 2.6.6) theses, it follows that the universe must be a FSM and natural phenomena must all be computable.

2.4.5. Universality and The Universal Turing Machine (UTM)

Newell (1980) provides the following informal definition of the notion of universality:

For any class of machines, defined by some way of describing its operational structure, a machine of that class is defined to be universal if it can behave like any machine of the class .. The notion of universality thus arrived at is *relative*, referring only to a given class of machines [emphasis added]. (pp.149,150)

The latter position is endorsed by McMullin (1993a) who maintains the distinction between *computation* universality and *construction* universality, whereby the latter is understood the ability of a machine to build or construct any other machine in the same class given the appropriate program and materials (chapter 5). (Importantly, universal construction *necessitates* support for *self-reproduction*. This is because a universal constructor can construct *all* machines including the machine that is itself.) Minsky (1967) defines a Universal Turing machine (UTM) as follows:

A universal Turing machine (UTM) is a [Turing] machine U with the property that for each and every Turing machine T , there is a string of symbols d_T such that if the number x is written in standard notation on a blank tape, followed by the string d_T , and U is started in q_0 on the leftmost symbol of d_T , then when the machine stops the number $f(x)$ will appear on the tape, where $f(x)$ is the number that would have been computed if the machine T had been started with only x on its tape. (p.136)

McMullin (1993a) maintains that a UTM is *doubly* universal since

it is firstly universal with respect to all [Turing machine] computations (which give it its original title); but this then turns out (at least if the Church-Turing Thesis is accepted) to mean that it is universal with respect to the computations of *any* effective computing system whatsoever, not 'just' those of the [Turing machine] system. (p.4)

According to Newell (1980), "the class of all Turing machines is very large - by using enough states (and it may take a very large number) the input-output behaviour of *any* physical mechanism can be approximated as closely as required." (p.152) It must be remembered that digital computers can *only* approximate UTMs since no physical machine has an infinitely large memory (or 'tape'). TMs therefore provide an *idealistic* model of computation (Garnham,88) since physical computers are, in reality, finite state

⁵ In fact, on the "strong" computationalist position, natural phenomena such as matter, life and mind are themselves computational in nature.

machines (FSMs) (Cariani,89). This limitation certainly applies to *artifactual* computers, that is, to computational devices; however, it is an open issue whether the universe itself is finite or infinite⁶. Consequently, it is an open issue whether the universe is a UTM or FSM (assuming it *is* ontologically computational). Hence, TMs (as originally conceived) are *ontologically*-idealistic as opposed to existentially-constructable entities.

The concept of universality described above is intrinsically Platonic or formistic (section 2.7.2) in nature: Functionality is defined purely in abstract behavioural terms, the *situatedness* of functional activity, that is, the context in which behaviour is regarded *as* functional, being ignored. While it may indeed be the case that two systems are universal with respect to the Platonic (or space-time independent) aspect of their functionality, this functionality may necessitate specific spatio-temporal conditions in order to be realized. If this is the case, then can it really be said that the functionality of two systems operating in two different spatio-temporal contexts is identical ? Pattee (1995a), in the context of a discussion of the evolution of biological organisms, makes a similar point regarding the necessity of including spatio-temporal *responsivity* requirements in the definition of functionality: Two systems must generate identical behaviour within identical spatio-temporal observation frames in order to be considered functionally equivalent. Only by assuming a Platonic position is it possible to reduce functionality to context-free behavioural activity. According to Heidegger, however, the capacity for conceiving function in this way is restricted to a particular class of systems with the capacity for abstract, thematic reflection (chapters 1 and 6), viz. human beings (or *Dasein*); on his view, the *abstract* Platonic conception of functionality is incomplete since human beings are *concretely* 'thrown' into the *historical* situation that is 'world' (chapter 6). Thus, there is at least one kind of Being that is not adequately conceived in terms of ahistorical functionality.

2.4.6. The Church-Turing Thesis (CTT)

The Church-Turing Thesis (or CTT) states that any process which could naturally be called an effective procedure can be implemented by running a suitable program (TM specification) on a UTM. An alternative formulation of the CTT (Hopcroft,79) is that

the intuitive notion of 'computable function' can be identified with the class of partial recursive functions, i.e. the class of integer functions computable by Turing machines, assuming that the intuitive notion of 'computable' places no bounds on (i) the number of steps or (ii) the amount of storage (i.e. space-time requirements) involved in computation.

Although one cannot prove that the TM model is equivalent to the intuitive notion of an effective procedure or computer, there seem to be compelling arguments for this position, specifically the fact that universal recursive functions, Post canonical systems, Markov algorithms etc are all equivalent to UTMs. Minsky (1967) asserts that "any

⁶ Additional support for this position is provided by evolutionary cosmology (chapter 6).

procedure which could 'naturally' be called effective, can in fact be realised by a (simple) machine." (p.105) However, he recognizes the problems associated with this position, which is an informal statement of the CTT, emphasizing that it is "a subjective matter, for which only argument and persuasion are appropriate; there is nothing here we can expect to *prove*." (p.108) As Newell (1980) states,

Church's statement is called a *thesis* because it is not susceptible to formal proof, only to the accumulation of evidence. For the claim is about ways to formalize something about the real world, i.e. the notion of machine or determinate physical mechanism. (p.150)

The CTT raises a number of interesting issues. For example, Minsky (1967) states that

[the] most obvious application [of the notion of an effective procedure] is to computation and computers, but I believe it is equally valuable for clear thinking about biological, psychological, mathematical, and (especially) philosophical questions. (p.viii)

This generic application of the concept of an effective procedure (or computation) throughout the phenomenal hierarchy leads to the notion of computationalism.

2.5. Computationalism

The origins of *computationalism*, the metaphysical view that the world at its most fundamental level is computational in nature, can be traced to two related disciplines, viz. computational psychology and the philosophy of mind. The essence of the original notion is captured in the following statement of McMullin (1993b), viz.

all mental states and events, can, in principle, be completely reduced, without residue, to states and events of some universal computer (p.1)

and is clarified by the dictum "mind is to brain as program is to hardware" (Johnson-Laird,88), a computational variant of Cartesian mind-matter dualism referred to in the literature as the "strong" AI thesis (Searle,80) or, more explicitly, computationalism (Dietrich,90) (Shapiro,95). Sharples et al. (1989) define computationalism as

the notion that the operation of the mind can be explained entirely in terms of the formal, or functional, properties of a computational system.

Thus, computationalism is a type of functionalism (chapter 1), the latter of which is associated with notions such as *multiple-realizability* and medium independence (chapter 4). The functionalist position can be formally stated as follows:

a computational state s_1 is identical to a computational state s_2 if the causal relations holding between s_1 , its inputs, outputs and other states are identical to the causal relations holding between s_2 , its inputs, outputs and other states.

Significantly, the identity relation holding between two computational states makes no reference to the properties of the substrate or media in which the computations are realized beyond the capacity of these media to support computation. Hence, in computationalism, form is separable from matter and the mind can be understood without doing neurophysiology since it is independent of the brain in the same way that programs are independent of hardware (Searle,80). It is important to appreciate that multiple realizability is a general statement about the functional or causal equivalence of phenomena rather than a specific statement about the nature of phenomena such as mind. This makes possible the extension of the original mentalistic concept of computationalism to other phenomenal domains such as biology leading to the vitalistic dualism, 'life is to body as program is to hardware', known as the "strong" A-Life thesis (Sober,91). However, its application to physics leads to an interesting situation necessitating an inversion of the hardware-software relation: 'matter is to computation as program is to hardware'. What was previously identifiable as hardware (matter) in the mind-brain and life-body relations is now defined as software and what was previously associated with software (computation) is now identified with hardware. However, are 'soft-hardware' and 'hard-software' coherent concepts? In Kantian terms, moving from phenomena (the physical) to noumena (the metaphysical) has necessitated inverting the hardware-software dualism (Fig 2.3):

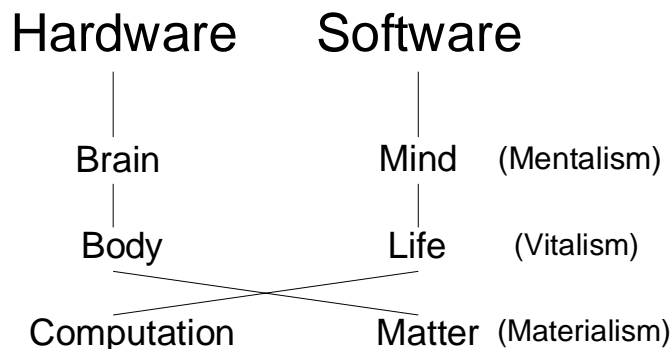


Fig 2.3 Computational Phenomenal Dualisms.

The dissimilarity between 'computational-materialism' and other computational phenomenal dualisms is described by Rosen (1987) as follows:

Although .. the mechanical theories of physics (and the field theories, too, for that matter) are closely related in their *form* to the ideas of Turing, and to the formalizations his machines execute, there is one obvious difference. Namely, a system of particles is, in a sense, *all hardware*. The dualism between state (phase) and dynamical law is at root not the same as the dualism between hardware and software, which is inherent in the 'mechanization' of formal processes by Turing machines. For instance, in our little embodiment of mechanics by a two-tape machine, *the tape processor (the 'hardware' of the machine) has no counterpart in the physics of the system*; the mechanical system itself has, in turn, become *all software*. (p.9)

Assuming the above inversion is valid, it is worthwhile assessing the status of the computationalist thesis. There are at least three perspectives on this issue: First, there are those such as Bringsjord (1996) who maintain that computationalism, as a thesis about mentality, *can* be and already *has* been refuted using variants of Searle's Chinese Room argument; second, there are others such as McMullin (1993b), who are sympathetic to this position, yet maintain that the arguments used to support it are weak; and finally, there are those such as Dennett (1995) who continue to uphold variants of the computationalist position on the basis of the following statement made by Pylyshyn (1980), viz.

computation is the only worked-out view of process that is both compatible with a materialist view of how a process is realized and that attributes the behaviour of the process to the operation of rules upon representations (p.113)

While there may be grounds for refuting computationalism within AI, cognitive science and the philosophy of mind, it is unclear at the outset whether (and if so, how) such arguments extend to computational life (Keeley,93) and/or computational matter (Fredkin,90). Statements such as the following due to Newell et al. (1976) support this latter cautionary view, viz.

the phenomena surrounding computers are deep and obscure, requiring much experimentation to assess their nature. (p.114)

General formulations of computationalism, such as that *any* physical structure in which states and state transitions can be interpreted as representing some other system is a computer (Churchland,92), support the application of the concept throughout the phenomenal hierarchy. For example, the idea that the world is a formal system of the Turing machine kind is expressed in the following speculation by Hofstadter (1979):

One could suggest .. that reality is nothing but one very complicated formal system. Its symbols do not move around on paper, but rather in a three-dimensional vacuum (space); they are the elementary particles of which everything is composed .. The [formation and transformation rules] are the laws of physics, which tell how, given the position and velocity of all particles at a given instant, to modify them, resulting in a new set of positions and velocities belonging to the 'next' instant. So the theorems of this grand formal system are the possible configurations of particles at different times in the history of the universe. The sole axiom is (or perhaps, was) the original configuration of all the particles at the 'beginning of time'. (p.53)

This position is endorsed by Tipler (1995) who *defines* the human being as

nothing but a particular type of machine, the human brain as *nothing but* an information processing device, the human soul as *nothing but* a program being run on a computer called the brain. Further, all possible types of living beings, intelligent or not, are of the same nature, and subject to the same laws of physics as constrain all information processing devices [emphasis added]. (p.xi)

However, perhaps the boldest statement of the computationalist thesis is that appearing

in (Tallis,94):

Physical systems are .. computational systems, processing information, just as computers do, and scientific laws may be considered as algorithms. This extraordinary view is enhanced by the observation that in post-classical (quantum) physics many physical quantities normally regarded as continuous are in fact discrete: nature is thus more amenable to digitization. In other words, the universe is not merely a huge computer: it is a huge *digital* computer. (p.31)

In order to further clarify the notion of computationalism, three related theses will now briefly be examined: (i) the Physical Symbol System Hypothesis, (ii) the Physical Church-Turing Thesis, and (iii) the Finite Automaton Thesis.

2.5.1. The Physical Symbol System Hypothesis (PSSH)

The *Physical Symbol System Hypothesis (PSSH)* (Newell,76) states that

a physical symbol system has the necessary and sufficient means for general intelligent action

whereby 'general intelligent action' is meant "behaviour appropriate to the ends of the system." (p.116) On this definition, a physical symbol system is necessarily a teleological (goal-directed) entity. While this *might* be acceptable for mind and life, the notion of teleological matter is incoherent with respect to Newtonian and post-Newtonian physics and the PSSH assumes the latter. Hence, in order to facilitate generalization of the PSSH to life and matter, it is necessary to redefine it as follows:

a physical symbol system has the necessary and sufficient means for generating phenomena such as matter, life and mind.

Newell et al. (1976) define *a physical symbol system (PSS)* as

- a set of physical patterns called *symbols*
- a set of physical structures called *expressions* composed from a number of instances (or *tokens*) of symbols related in some physical way
- a set of *processes* for producing new expressions (symbol structures) from existing expressions

The crucial issue for Newell (1980) is realizability:

a physical symbol system .. is realizable in our universe [and] its notion of symbol is a priori distinct from the notion of symbol that has arisen in describing directly human linguistic, artistic and social activities. (p.141)

Simon (1981) expands on the latter point by asserting the ontological nature of the PSS concept:

The computer is a member of an important family of artifacts called symbol systems, or more

explicitly, *physical symbol systems*. Another important member of the family (some of us think, anthropomorphically, it is the *most* important) is the human mind and brain. (pp.26-27)

Two notions are associated with a PSS:

- *Designation*: An expression designates an object if, given the expression, the system can either (i) affect the object itself or (ii) behave in ways depending on the object.

Designation therefore necessitates establishing the equivalent of encoding (sensor, measurement) and decoding (effector, control) relations between objects and a PSS.

- *Interpretation*: The system can interpret an expression if the expression designates a process and if, given the expression, the system can carry out the process.

According to Pylyshyn (1980), "[syntactic or symbolic] expressions are 'interpreted' by the built-in functional properties of the physical device." (p.113) Thus, a PSS is argued to be capable of self-interpretation. However, the most important feature about PSSs as regards the bearing of the PSSH on the issue of computationalism is that

symbol systems form a class - it is a class that is characterised by the property of *universality* .. Central to universality is flexibility of behaviour .. a universal machine is one that can produce an arbitrary input-output function; that is, that can produce an independence of output on input. (Newell,80;p.147)

As Newell (1980) states, "our situation is one of defining a symbol system to be a universal machine, and then taking as a hypothesis that this notion of symbol system will prove adequate to all of the symbolic activity this physical universe of ours can exhibit, and in particular all the symbolic activities of the human mind." (p.155)

From the above, it is observed that PSSs are distinguishable from formal systems (section 2.3) in two ways: (i) the primitives and operations in a PSS are physical and hence, subject to the laws of physics; (ii) symbols in a PSS exist ontologically, that is, independent of human interpretation. If the laws of physics are themselves recast in formal (syntactic) terms and a "strong" formalist or computationalist view of reality is assumed (chapters 4 and 5), PSSs become identical with formal systems. Newell et al. (1976) maintain that a PSS is part of a larger world which includes objects, that is, entities which are not PSSs. This might appear to entail the view that the reduction of PSSs to formal systems cannot be complete (that is, totalistic). However, under formalism, the ontology of objects is also ultimately formal (section 2.5.2); hence, the dualism of objects and physical symbol systems reduces to an ontologically monistic formal system. The two kinds of entities *can* be functionally distinguished for certain kinds of PSS: a mental or vital PSS is goal-directed whereas an object (material PSS) is not; however, whether such distinctions are intrinsic or extrinsic is an open issue.

2.5.2. The Physical Church-Turing Thesis (PCTT)

Randall (1996) defines four increasingly "strong" versions of the Church-Turing thesis:

CTT(1) All *computable* things correspond to a lambda-form or Turing Machine. (*Mathematical*)

CTT(2) All *physical* things correspond to a Turing Machine. (*Physical*)

CTT(3) All *thinkable* things correspond to a Turing Machine. (*Mental/Epistemological*)

CTT(4) All *things* correspond to a Turing Machine. (*Ontological*)

Acceptance of the validity all four versions of the CTT leads Randall to argue for the existence of immaterial (non-physical) computables only:

	<i>computable</i>	<i>uncomputable</i>
<i>physical</i> (<i>material</i>)	dissolution to void (forms = reality)	rejected by CTT(2) & dissolution to void
<i>non-physical</i> (<i>immaterial</i>)	accepted by all CTTs	rejected by CTT(4)

Table 2.1

On Randall's view, CTT(2) or the *Physical* Church-Turing Thesis (PCTT) must ultimately be replaced by CTT(4), that is, the *Ontological* CTT (OCTT). This is a consequence of the dissolution to void suffered when physical entities are analysed in order to determine what constitutes their 'similarity in difference': The reality of physical entities is defined by the extent to which they participate in Platonic forms (or ideas) to which the mind has access (section 2.7.2); hence, entities can have no real existence apart from the forms. On Randall's view, physics must be reduced to form since "any attempt to define physical things as things-in-themselves apart from the forms suffers dissolution into nothingness, as [happens] for static sets." (p.10) However, the metaphysical nature of the commitment to identify existence or Being with form must be recognized (chapter 1)⁷: Randall's reduction of the PCTT to the OCTT only holds if Platonism is assumed *a priori*; crucially, an existential-computationalist position is consistent with the PCTT. Moreover, Randall's claim that physical uncomputables are meaningless is problematic since examples of physical uncomputables have been documented in the literature (Calude,95). However, the OCTT is the basis of computationalism; hence, although the PCTT provides a sufficient framework within which to investigate the emergence of

7

Furthermore, and as will be seen in chapter 6, the conception of Being (or existence) in *static* (that is, Platonic) terms is highly problematic.

artificial analogues of natural *physical* phenomena from a computational substrate, the PCTT must be viewed as subsumable by the OCTT⁸.

Rosen (1991) describes the PCTT as follows:

through equivocation on the word 'machine', Church's Thesis can be painlessly transmuted into an assertion about the material world itself; an assertion about what can be entailed in the causal world of material phenomena. That is, Church's Thesis can be interpreted as an assertion about the structure of the category of all models of any material systems. (p.9)

According to Rasmussen (1991b), the PCTT entails holding that "a universal computer at the Turing machine level can *simulate* any physical process." (p.768) The problem with this position is that the word 'simulate' is ambiguous: Does it imply *appearance* (*as-if*, simulation proper) or *reality* (instantiation, realization) ? On the "strong" (ontological) interpretation of computationalism it must be the latter. (The difference between simulation, realization and *emulation* is discussed in chapter 4).

Svozil (1993) maintains that the CTT includes a physical as well as a syntactic claim by specifying which types of computations are physically realizable: "As physical statements may change with time, so may our concept of effective computation." A corollary of this is that digital computers are physical systems which are universal up to finite complexities. Thus, according to Svozil, physics constrains computation. However, if the PCTT is afforded ontological status the position is reversed. As Rosen (1991) argues, the implications of the CTT for the mathematical and physical worlds are quite different:

In mathematics, Church's Thesis does no .. damage, because whatever is not simulable is thereby relegated to the category of 'ineffective' .. Gödel's Theorem already tells us that, in these terms, almost all of Number Theory is thereby rendered 'ineffective'. But the material world is different; whatever happens, or can happen in it, must thereby be 'effective'; at least, it must be so in any normal usage of that term. Hence the equation 'effective' = 'simulable', which is the essence of Church's Thesis boxes us in from the outset to a world of simple systems; a world of mechanisms. (p.11)

Thus, while physics may constrain computation, computation also constrains physics. (This issue is examined further in chapter 4 in the context of a computational theory of matter.)

2.5.3. The Finite Automaton Thesis (FAT)

This "strong" (or ontological) version of the finite automaton thesis (FAT) states that the universe *is* a finite automaton, whereby *finite automaton* is meant a computational device

⁸ To the extent that mind (more specifically, consciousness) is a natural, yet *non-physical* phenomenon, this subsumption becomes necessary.

which is finite in all its features, for example, a TM with a finite number of internal states, finite number of input and output symbols, finite tape etc. This version of the thesis involves the following two claims (Svozil,93), viz.

- the 'laws of nature' are mechanistic, that is, computable in the Church-Turing sense
- under certain 'mild' assumptions, the computational capacities of physical systems are finite

The 'weak' (epistemological) version of the FAT merely states that there are phenomena in automaton universes which translate into physics and which are only *sensibly* analyzed using algorithmic techniques. The FAT is closely connected to the finite nature thesis (section 2.6.6); however, it is an open issue whether the universe it itself finite or infinite. Hence, the status of the FAT is *a priori* undecided and, perhaps, *undecidable*, given the endosystemicity of human observers⁹. However, computationally, the universe *is* either a finite automaton or a UTM.

2.5.4. Concepts of Computation

Emmeche (1993) (1994) proposes to extend the definition of computation by distinguishing four concepts (as contrasted with kinds) of computation:

- | | | |
|---|---|---------|
| □ | COC1: Formal or algorithmic (symbolic) | Reality |
| □ | COC2: Informal, intuitive, or 'mathematical' | Mind |
| □ | COC3: Biological | Life |
| □ | COC4: Physical or non-representational (non-symbolic) | Matter |

Emmeche (1994) maintains that the problem of the different concepts of computation may be resolved by adopting a pluralistic stance, viz. "that we simply face different kinds of computations." (p.19) However, this view is incoherent: If there are many kinds of computations, what is it that allows them to be identified *as* computations ? It must be that they are members of the *universal class* of computations. Hence, computational-pluralism must, somehow, reduce to computational-monism.

Following Fetzer (1990), Emmeche (1994) further argues for a semiotic (significative or sign-based) understanding of the concept of computation, viz.

it is by no means clear how to speak rationally about computations without presupposing the existence of a complex system including (a) a conceptual structure of symbols, rules of manipulations, and well-formed strings as axioms to be manipulated; plus (b) an organism or a well-designed physical device that in some way can do the manipulations; and thirdly, (c) an interpreter, that makes sense of (a) and (b). (p.19)

However, assuming the ontological form of the PCTT (section 2.5.2), (i) these triadic

⁹ That is, the situatedness of human observers within the physical universe.

functions must be realizable by a PSS that is itself a formal system and (ii) *COC2-COC4* must be reducible to *COC1*.

2.5.5. Definition: Computationalism

This metaphysical position may be stated in Kantian terms as follows: The noumenal reality underlying phenomenal appearance is computational; hence, all natural and artificial phenomena are, *ultimately*, computational phenomena. This statement is to be interpreted in the "strong" *ontological* sense implied by the maxim, "Being is computation" (chapter 6), and not in the "weak" *epistemological* sense that computation merely provides a means for understanding (explaining and predicting) the various manifestations of Being.

2.5.6. Analog Devices and Connectionism

There may be a problem with the Church-Turing thesis given that it was originally formulated in the context of discrete computational devices, that is, digital computers. Siegelmann (1995) has argued that analog connectionist devices such as analog recurrent neural networks (ARNN) have *super-Turing* capabilities; consequently, there may be a need to complement the Church-Turing thesis with an analog computation thesis. However, others have argued that this view is incorrect since it assumes, for example, that neural network coefficients can have arbitrarily large accuracy, that is, an infinite amount of information may be packed into each coefficient (Knight,96). Such devices, it is argued, are not physically realizable and since they clearly violate the finite automata thesis (FAT) (section 2.5.3) and Fredkin's finite nature thesis (section 2.6.5) will not be considered further herein¹⁰.

Regarding connectionist systems (that is, networks of elements with weighted links), Kelly (1993) observes that "there is some dispute about the real nature of connectionism, whether it is truly a new paradigm or is merely another mode of implementation of essentially classical systems." (p.202) This observation is supported by (1) the computational equivalence of discrete-state artificial neural networks and Turing machines (Bringsjord,90), (2) the ontological relation between declarative (or functional) programs and artificial neural networks (Salt,96), and (3) the formal equivalence of discrete-state artificial neural networks and symbolic computational systems (such as classifier systems and semantic networks) which can be shown by reduction to a generic

¹⁰ However, as stated in section 2.5.2, the existence of physical uncomputables *has* been documented in the literature. According to Cariani (1989), hybrid analog-digital devices constitute examples of *finite*, physical, uncomputable systems capable of open emergence (chapters 3 and 6). On this *hylomorphic* view, the physical universe is held to be finite in extent yet offering infinite capacity for classification via measurement. It is hard to see how this eliminates the role of infinity completely and, if Knight's criticisms of infinities in ARNNs extend to Cariani's evolutionary robotic devices, it must be the case that the equivalent of a TM tape of infinite length has been tacitly reintroduced into the picture.

mathematical formalism based on graph-theoretic concepts (Farmer,90). Since discrete *nature* is assumed in this study (section 2.6, chapters 4 and 5), analog devices and neural networks must be reducible to discrete devices.

2.6. Implementing Computationalism

In this section the cellular automaton formalism, which provides the means for realizing the computationalist world view described in section 2.5, is introduced. Only deterministic, that is, non-probabilistic, cellular automata (CAs) are considered. However, this does not present any problems since nondeterminism adds nothing to the power of TMs (section 2.4.3) and the class of CAs which are of ultimate interest in the context of this study, viz. Universal CAs (or UCAs), are computationally equivalent to UTMs.

2.6.1. From Turing Machines to Cellular Automata

Putnam (1988) maintains that

functionalists have abandoned the Turing machine formalism, and so far we have only the vaguest descriptions of what the computational formalism is supposed to be. Without a computational formalism, the notion of a 'computational state' is meaningless. (p.84)

Although Putnam's comments were made in the context of an evaluation of the functionalist programme within cognitive science and the philosophy of mind, they have much broader implications for an ontological computationalism grounding artificial analogues of natural phenomena such as matter, life and mind. Specifically, there is a need to redefine computationalism in terms of a new formalism, one which will support functional isomorphisms with natural phenomena. Assuming an emergentist perspective (chapters 3 and 5), it becomes necessary to examine the way in which a computational ontology must be realized in order to provide the ground for an artificial analogue of the lowest level in the natural phenomenal hierarchy, viz. the material or physical level.

Hillis (1985) describes the two senses in which 'computational model' may be understood: (i) "a model of all possible computational worlds", that is, a "metacomputational theory" or (ii) "a model of a particular computational system" and "physics may be such a model" (p.142). Analogously, Pattee (1995a) differentiates between two interpretations of computation: (i) "computation as a universal, abstract dynamics to which even the laws of physics must conform" and (ii) "computation as a locally programmable, concrete, material process strictly limited by the laws of physics." (p.29) Given the nature of the study described in this thesis, it could be argued that 'computational model' must be understood in the first sense since computationalism is a metaphysical position and hence, encompassing of the physical (chapter 1). However, given the specific form of computationalism presented in this study, viz. a computationalism supporting the emergence of artificiality (or artificial analogues of

natural phenomena), the computational model must support computational analogues of physical concepts since the physical level constitutes the lowest level in the phenomenal hierarchy.

In the context of a discussion of 'why computer science is no good', Hillis (1985) asks whether or not there will ever be "a model of computation that is as powerful and beautiful as our models of physics", maintaining that "computer science is missing many of the qualities that make the laws of physics so powerful: locality, symmetry, invariance of scale." (p.137) *Locality* means that objects must be in contact in order to interact with each other; action at a distance is not allowed. As Hillis states, "our old models of computation impose no locality of connection, even though the real world does." (p.139) (This Newtonian view is problematic since it ignores non-local physical interaction between particles at the quantum level. However, as discussed in chapter 5, attempts have been made to overcome such problems within classical or Newtonian systems.) *Symmetry* implies reversibility, viz. physical laws apply irrespective of the 'direction' of time. Again, as Hillis states, "in physics .. many fundamental quantities are conserved, whereas in our old models of computation, data can be created or destroyed at no cost." (p.138) (This is an idealization since in a closed system - and the universe is assumed closed under this argument - entropy, which measures the degree of disorder in a system, increases as a consequence of the dissipation of energy in the form of heat during collisions between bodies within the system.) *Invariance of scale* simply means that physical laws apply irrespective of the size of the objects concerned. (Again, this is an idealization: As stated in chapter 4, when objects are very small, quantum effects start to become significant, and when moving close to the speed of light, relativistic effects become significant). Assuming the problems described above - which arise under idealized conditions - can be overcome, a computational substrate supporting analogues of physical properties such as locality, symmetry, and invariance of scale provides a suitable candidate for defining what is meant by computation and realizing (or implementing) ontological computationalism. As will be seen in what follows and in chapter 5, the cellular automaton (CA) is such a substrate. Informal introductions to CAs based on an examination of the Game of Life are provided in (Berlekamp,82), (Gardner,83) and (Poundstone,85); more detailed investigations of elementary CAs are described in (Wolfram,83b,84b,86). In the following sections the CA formalism and some of its basic properties are described; CA models of specific natural phenomena such as matter, life and mind are discussed in chapter 5.

2.6.2. Cellular Automata and Computationalism

A cellular automaton (CA) provides a suitable mathematical framework (*formalism*) for modelling natural systems with large numbers of discrete degrees of freedom; in this respect, they are the discrete equivalent of models based on systems of differential equations. CAs represent universes of pure information (Stonier,92) and may be regarded as "stylised, synthetic universes" (Toffoli,87) of the Newtonian kind: Space is represented as a uniform Cartesian grid or lattice of cells with each cell containing a

finite amount of information ('matter'); absolute time advances in discrete steps and 'universal laws' are expressed by an update rule which is locally defined and globally applied. Wolfram (1984a) identifies the following five characteristics as defining standard CAs:

1. They consist of a discrete lattice of sites.
2. They evolve in discrete time steps.
3. Each site takes on a finite set of possible values.
4. The value of each site evolves according to the same deterministic rules.
5. The rules for the evolution of a site depend only on the local neighbourhood of sites around it.

Standard CAs provide general discrete models of *homogeneous* systems whose global behaviour is determined by the long-term time evolution of local interactions. It must be recognized from the outset that standard CAs are 'closed' systems (chapter 3) since the dynamics of such systems over time is completely determined by the initial state of the lattice cells and the local interaction rule (Aleksic,92).

A cellular automaton (CA) can be viewed either (i) as a computer, or (ii) as a *logical universe*, a structure into which may be embedded higher order computational structures including 'virtual' computers (Langton,90). This is a consequence of the fact that in a CA, objects interpretable as passive data and objects interpretable as computational devices are both assembled out of the same kind of structural elements and are subject to the same fine-grained laws. Hence, CAs can be viewed either as information *transducers* or informationally *autonomous systems* (Toffoli,87). Langton clarifies the distinction as follows:

On the first view, an initial configuration constitutes the data that the physical computer is working on, and the [state] transition function implements the algorithm that is to be applied to the data.

On the second view, the initial configuration itself constitutes a computer, and the [state] transition function is seen as the 'physics' obeyed by the parts of this embedded computer. The algorithm being run and the data being manipulated are functions of the precise state of the initial configuration of the embedded computer. In the most general case, the initial configuration will constitute a universal computer. (pp.15-16)

Clearly, for CAs to provide a means of realizing computationalism, the latter of the two positions must be adopted. According to Wolfram (1985),

one *expects* the fact that computers are as powerful in their computational capacities as any physically realizable system can be, so that they can simulate any physical system [emphasis added]. (p.735)

2.6.3. Definition: Cellular Automaton (CA)

Cellular automata (CAs) are discrete dynamical systems consisting of d -dimensional ($d \geq 1$) lattices of cells in which each cell is a finite state machine (FSM) - or finite state *automaton* (FSA) - defined by the following triplet:

$$\langle S, N, R \rangle \quad (2.1)$$

S is the set of states that each FSA may assume. S^N is the set of input neighbourhood states each of which is defined as the cross product of the states of those cells covered by a template of size $|N| \geq 1$. (By convention a cell is included in its neighbourhood). Examples of neighbourhood templates when $d=2$ for lattices with different geometries (square and hexagonal respectively) are shown in Fig 2.4. CAs with $d=3$ (Bays,87a) are particularly interesting since their geometries correspond to the perceived three-dimensions of the natural world and therefore provide a suitable basis for modelling the universe (chapter 5). $R: S^N \rightarrow S$ is the state-transition rule defined by associating a unique next state in S with each possible neighbourhood state in S^N . Since there are $|S|$ possible states which can be assigned for each of the $Q=|S^N|$ possible input neighbourhood states, there are $|S^Q|$ possible state-transition rules R which can be defined. For example, in a binary CA with a neighbourhood of three cells ($S=2, N=3$), there are 256 possible state-transition rules.

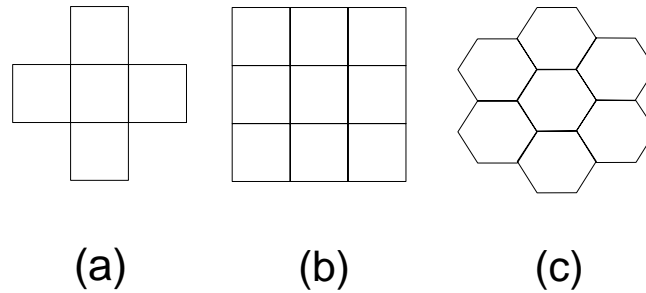


Fig 2.4 Various 2-D neighbourhood templates.

A CA is defined by the 5-tuple:

$$\langle k, S, L_0, N, R \rangle \quad (2.2)$$

where k with $k \geq 1$ is the size of the lattice (number of cells); L_0 is the initial configuration of cells in the lattice at time $t=0$ ($L_0 \in L$ and $|L|=|S|^k$); and S, N and R are as defined in (2.1). The lattice is either spatially (i) infinite ($k=\infty$), (ii) finite and periodic or (iii) finite and non-periodic. In standard CA, the local update rule R is applied *globally* (each FSA

computes the same function) and *synchronously* (all cells are updated simultaneously). For example, consider the 1-D periodic CA defined as follows:

$$\begin{aligned}
 S &= \{0,1\} \\
 |N| &= 3 \\
 k &= 10 \\
 L_0 &= 0010110100 \\
 R &= 000 \rightarrow 0, 001 \rightarrow 1, 010 \rightarrow 1, 011 \rightarrow 0, \\
 &\quad 100 \rightarrow 1, 101 \rightarrow 0, 110 \rightarrow 0, 111 \rightarrow 1
 \end{aligned}
 \quad (\text{XOR function})$$

The space-time evolution of this CA is as follows:

$$\begin{aligned}
 &0010110100 & (t=0) \\
 &0110000110 & (t=1) \\
 &1001001001 & (t=2) \\
 &0111111110 & (t=3) \\
 &1011111101 & (t=4)
 \end{aligned}$$

The space-time evolution of this CA (rule 150 following the Wolfram coding scheme) for 100 iterations ($t = 0..99$) with an initial configuration density $\rho_{INIT} = 0.01$ (lattice cells are initially assigned to be in the '1' or 'on' state with probability 0.01) is shown in Fig 2.5:

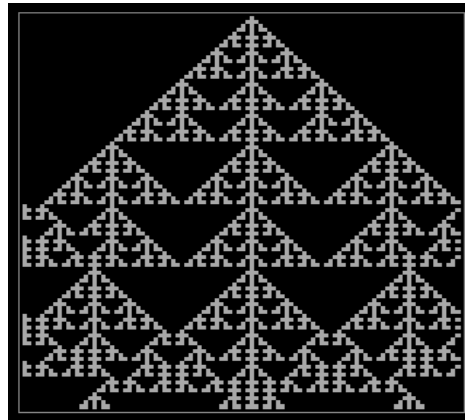


Fig 2.5 1-D CA space-time evolution (rule 150).

Conway's Game of Life rule (Berlekamp,82) is particularly interesting as an example of a 2-D CA that is capable of supporting universal computation. The rule is extremely simple: (i) if a dead cell is surrounded by exactly three live cells at time t then it will be live at time $t+1$; (ii) if a cell is in the live state and surrounded by either two or three live neighbouring cells at time t then it will remain live at time $t+1$ else it will die. Given a precise initial configuration of the lattice, this CA can generate an extremely diverse array of structures including blinkers (periodic oscillators), gliders (translating oscillators), puffer trains (dynamic structures which generate 'debris', that is, static

structures) and most importantly with respect to the construction of universal computers, glider guns (dynamic structures which emit gliders at regular intervals). An example of a glider is shown in Fig 2.6:

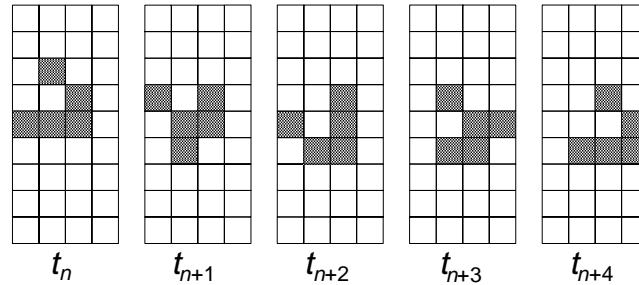


Fig 2.6 Space-time evolution of a glider.

After four iterations, the glider has translated diagonally by one cell. Gliders and other structures will be examined further in chapter 5 when various CA models of the universe are presented.

2.6.4. Describing the Structure of CAs

The following table describes a set of isomorphisms between natural systems, continuous dynamical systems, formal systems and TMs based on the Newtonian modelling scheme introduced in section 2.3.2:

<i>Natural</i>	<i>Dynamical</i>	<i>Formal</i>	<i>Computational</i>
primitives	number field	alphabet	tape symbols
state-space	state manifold	All symbol strings	All tape patterns
state	state	symbol string	tape pattern
initial state	initial conditions	axioms	input tape pattern
laws	vector field	inferences	program instructions
state-sequence	trajectory	derivation sequence	sequence of tape patterns
observables	attractors	theorems	output

Table 2.2 Isomorphisms between various systems.

Since CAs are discrete dynamical systems, they are readily described in terms which are analogous to those used in the description of continuous dynamical systems:

<i>Dynamical System</i>	<i>CA</i>
number field	FSM states
state manifold	state-space
state	state
initial conditions	initial state
vector field	state-transition function
trajectory	trajectory
attractors	attractors

Table 2.3 Isomorphism between continuous dynamical systems and CA.

In continuous dynamical systems described by systems of differential equations, a *manifold* is an n -dimensional space analogous to a surface. In a discrete dynamical system such as a CA, a *topological* manifold corresponds to the space defined by the geometry of the cell lattice. For example, in a finite two-dimensional CA with periodic boundary conditions, the topological manifold is a torus; in a three-dimensional CA, it is a toroid. The *state* manifold, on the other hand, defines the space of possible *global* states that the dynamical system can assume; however, it is more usual to refer to the state manifold of a CA as its state-space. In continuous dynamical systems, each point in the state manifold is described by a set of coordinates based on some number field: For example, each body in a gravitational system is described by three position coordinates and three velocity coordinates. In a CA, each point in the state-space is described by the state of the FSA at the corresponding lattice site. A CA state is defined as the global pattern formed by the states of the FSAs at each cell in the lattice; the state-space is the set of all such states. A state-transition function or rule is the equivalent of a vector field in a continuous dynamical system and determines how the CA evolves in space-time. In both discrete and continuous dynamical systems, a trajectory is a sequence of states, a basin of attraction is the set of all trajectories converging on an attractor, and the basin of attraction field is the set of all basins of attraction for a particular system. (In deterministic systems such as CA the basins are discontinuous, that is, unconnected). *Attractors* mark the end points in dynamical systems. In continuous dynamical systems, there are three kinds of attractors: (i) fixed point, (ii) limit cycle, and (iii) chaotic or 'strange' attractors. However, in discrete dynamical systems such as CA, four kinds of attractor have been identified corresponding to four classes of behaviour (Wolfram,83b), the first three of which are directly analogous to those identified in continuous dynamical systems:

<i>Class I</i>	evolution leads to a homogeneous state
<i>Class II</i>	evolution leads to a set of stable or periodic structures
<i>Class III</i>	evolution leads to a chaotic pattern
<i>Class IV</i>	evolution leads to complex structures, sometimes long-lived

A detailed study of the basin of attraction fields for a range of CAs and a related formalism, the random-boolean network (Kauffman,93), is presented in (Wuensche,93).

An example of a basin of attraction field is shown in the following graphical state-space portrait (Fig 2.7). The topology of the portrait is that of transient branching trees rooted on attractors; the directionality of the graph is from the outermost branches to the innermost attractor sites. Nodes in the network represent global system states and arcs represent the transitions between states. States without precursor states are referred to as 'Garden of Eden' states.

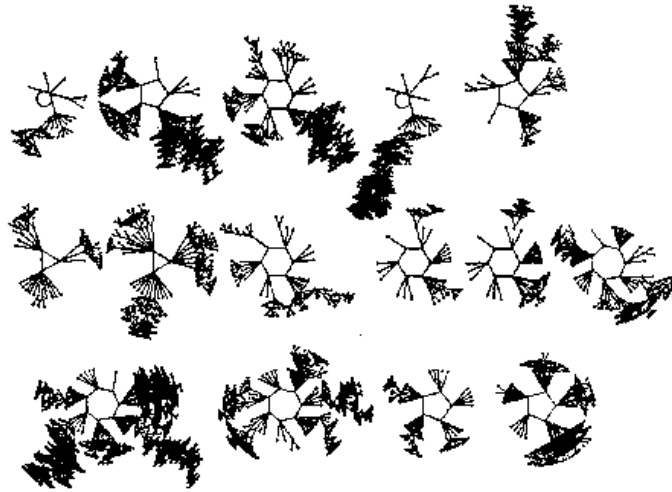


Fig 2.7 Example of a basin of attraction field.
[Source: (Wuensche,93)].

2.6.5. Universal Cellular Automata (UCAs)

CAs supporting behaviour in class IV, that is, generation of complex structures such as translating periodic oscillators or 'gliders', are of interest since the structures they produce can be used to construct universal computers (CA equivalents of UTMs). A Universal CA (UCA) is a computation-universal or UTM-equivalent CA capable of simulating the behaviour of any other CA (Fredkin,90). The empirical or constructive proof of computational universality for the two-dimensional CA known as the Game of Life is described in (Berlekamp,82) and outlined for the three-dimensional version of the same CA in (Bays,87a). However, computation-universal one-dimensional CAs also exist provided $|N-1| \text{ DIV } 2 > 1$ or/and $|S| > 2$ (Wolfram,83b).

2.6.6. "Digital Mechanics"

Laplace maintained that given the position and momentum of every particle in the universe at any given time, it is possible to predict the past or future of each particle. This formulation of the mechanistic idea of determinism (section 2.7.4) introduces the notion of *reversibility*, viz. that the spatio-temporal evolution of a system may be run

backward as well as forward. The Laplacian thesis was formulated in the context of a consideration of *linear* systems, which do not suffer the unpredictability effects associated with the *n*-body problem. (The latter arises from interactions involving *n* bodies, where $n \geq 3$; classical or Newtonian physics is unable to predict the outcome of such events because analytical solutions of *n*-body systems of interaction cannot be generated.) It can be shown that the Laplacian position is epistemologically untenable given the chaotic behaviour of non-linear dynamical systems; however, it remains intact as an ontological thesis finding its modern expression in computationalism. The idea that the future state of the universe can be completely determined by applying the laws of physics to its current state provides the foundation for digital mechanics (DM) (Fredkin,90), a computationalist theory of nature based on the cellular automaton (CA) formalism. Briefly stated, Fredkin argues that some CA model, a reversible universal CA, may be programmed to act like physics. In order for DM to model *both* classical (Newtonian) *and* post-classical (quantum) physics, it is necessary to assume the 'finite nature' thesis which is closely related to the finite automaton thesis (section 2.5.3):

- I the universe is finite in the amount of information contained in a finite *volume* of space-time
- II the universe is finite in the *total* volume of space-time

On this point, Fredkin maintains that

finite nature may or may not be true, but surely the assumption is not known to be false. If finite nature turns out to be false then DM is irrelevant as an exact model, but it might be useful as an approximate model (the way computers are presently used to model physics). (p.256)

Additionally, Fredkin maintains that DM must be computation-universal (section 2.4.5): If microscopic physics (assuming finite nature) was not universal, it would not be possible to macroscopically construct ordinary computers; however, since this *is* possible, it might be inferred that nature *is* an instantiation of a UTM. As Fredkin states,

finite nature does not just hint that the informational aspects of physics are important, it insists that the informational [that is, syntactic] aspects are all there is to physics *at the most microscopic level* [emphasis added]. (p.259)

However, simply because nature is computation universal does not necessarily imply that it is a UTM since it is quite possible that nature supports a more powerful form of information processing (Rasmussen,91b). Additionally, the notion of information and information-processing is itself problematic since information implies the existence of an entity which can be informed. A possible solution to this problem is provided in the ambiguity of the closing words in the above statement, viz. "at the most microscopic level": It is conceivable that entities with the capacity for being informed may *emerge* from the syntactic DM substrate; thus, syntax can give rise to semantics with the latter as an *emergent property* (chapter 3) of the substrate. This idea is explored in chapter 5 where an emergentist scheme based on DM is presented as a unifying framework for

computational matter, life and mind.

2.6.7. Computationalism and The Physical World

Barrow (1991) maintains that it is an unresolved issue whether the symmetry of physical laws or the notion of computation is fundamental:

Is the Universe a cosmic kaleidoscope or a cosmic computer, a pattern or a program? Or neither? The choice requires us to know whether the laws of physics constrain the ultimate capability of abstract computation. Do they limit its speed and scope? Or do the rules governing the process of computation control what laws of Nature are possible? (pp.203-204).

The possible ways in which physics constrain the possibilities for computation are discussed in chapter 5. As a preamble to that discussion, it is worth considering the following: In order for reality to be computational at its most basic level, that is, in order for computationalism to be an ontological thesis, it is necessary that the universe do only computable things. However, it is recognized that uncomputable mathematical operations exist, at least in some abstract sense¹¹ (section 2.4.4). In order to reconcile this fact with the computationalist thesis, it is necessary to maintain that computationalism is a thesis about *existence* and since "Being is computation" (section 2.5.5 and chapter 6), that is, to exist is to be computable, uncomputables do not *exist*; rather they *subsist* in a separate Platonic realm (section 2.7.2) or in the imagination. However, there are (at least) three problems with this position: First, the claimed physical *existence* of uncomputables (section 2.5.2); second, the fact that mind (more precisely, imagination) is capable of conceiving the possibility of uncomputables and yet is, in some sense¹², a physical phenomenon (Penrose,89) (Penrose,94); third, on this view, subsistence is equivalent to non-existence (since uncomputable) and yet subsistence is surely a way (or mode) of Being implying, thereby, that Being somehow *transcends* both computationalism and existence (chapters 6 and 7).

Barrow maintains that the concept of computation may need to be redefined given the quantum theoretical picture of reality (since quantum computation is more powerful than TM computation). However, under computationalism, physical theories based on mathematics involving non-TM-computable entities must be reducible *in principle* to theories which are describable in purely TM-computational terms. In chapter 5, arguments for subsuming quantum theory into computation theory based on a CA realization of digital mechanics (section 2.6.6.) will be presented.

¹¹ For example, as Platonic objects in the mind.

¹² To the extent that mind is phenomenon within the natural universe, mind can be viewed as physical; however, this should not be taken as supporting conventional materialism.

2.7. Deconstructing Computationalism

The previous sections were concerned with constructing a view of reality based on the concept of the computer. In this section, the computer metaphor is shown to be reducible to a synthesis of the form and machine metaphors. This reduction allows for the grounding of computationalism in at least three antecedent metaphysical positions, viz. determinism, mechanism and formism. Additionally, and as a consequence of the commitment to realize computationalism in a cellular automaton (CA) substrate, there is a need to examine a fourth metaphysical position related to mechanism, viz. atomism. The importance of reducing computationalism to antecedent metaphysical positions is that it provides a point of entry for a critique of the former: *Potential* shortcomings of computationalism may be identified by examining the *actual* (that is, existing) shortcomings associated with its philosophical precedents.

2.7.1. Computers, Forms and Machines

Putnam (1988) claims that computer science heralded the birth of a new paradigm (chapter 1) in the sense that it created a new metaphor for thinking about reality. However, is the computer metaphor new in the sense that it defines a 'root' metaphor (chapter 1) ? As stated previously (section 2.4), Toulmin (1993) maintains that the notion of the machine has evolved through history. Are computers simply a new kind of machine ? In order to answer this question, it is necessary to examine the possible connections between the computer metaphor and other existing root metaphors, specifically the metaphors of form and machine. It could be argued that the adoption of a 'bottom-up' or connectionist approach to realizing computationalism (such as that based on CAs) necessitates that notions associated with other root metaphors such as organism and context be considered; the organismic interpretation of Physical Symbol Systems and the implied contextualism of cellular automata (section 2.6.1) appears to support this contention. However, it could also be maintained that ideas associated with organism (or integration) and context (or history) are only *epistemologically*-significant when considering the *phenomena emerging* from a computational substrate and not *ontologically*-relevant to consideration of the *substrate* itself. Since this view seems to reflect computationalist thinking in general, and since it is computationalism *as* ontology that is under scrutiny in this study, it has been adopted, viz. the 'world hypothesis' (chapter 1) computationalism is assumed to be derived from the synthesis of two root metaphysical systems, viz. formism and mechanism.

2.7.2. Formism

The root metaphor of formism (or Platonism) is *similarity* (Pepper,42). In formism, objects of perception have independent dual aspects: (i) *particularity* (which is specific) and (ii) *quality* (which is universal). For example, the tree outside my window is a specific tree; however, tree refers to the universal concept or quality of tree-ness which

is independent of any particular tree. Other formistic categories include (iii) *relation*, which is the similarity between pairs of particulars and (iv) *character*, which refers either to a quality or relation.

There are two basic kinds of formism, viz. immanent and transcendent. In *immanent* formism, the basic categories are characters, particulars and participation, that is, ties between characters and particulars. Full appearance of characters in particulars is necessary. In *transcendent* formalism, the basic categories are norms, matter and a principle of exemplification which materializes the norms. Partial appearance of norms in matter is sufficient. However, both kinds of formism acknowledge:

- I categories of forms (characters, norms)
- II the appearance of these forms in nature
- III the connection between categories I and II

Particulars without characters are held to *exist*, particulars with characters are held to *exist concretely*, and forms are said to *subsist*. Hence, particulars exist in the material world, while forms subsist in the world of Platonic ideas.

According to Pepper (1942), causality in formism is the result of

the participation of patterns, norms, or laws in basic particulars through the forms of time and space. (p.175)

Causality is the determination of the characters of certain basic particulars by a law which is set in motion by the characters of other basic particulars which participate in that law. A law, in other words, is a bridge from one set of basic particulars to another set, determining the characters of one set by those of the other. (pp.176-177)

A law is not a basic particular, nor a concrete existent particular (i.e. a single exemplification of the law), nor a collection of concrete existent particulars (i.e. a class). *A law is a form* [and] this is one of the fundamental distinctions between formism and mechanism. (p.177)

The formistic notion of causality finds expression within discrete dynamical systems such as CAs in (i) *states*, which are configurations (shapes, patterns, or forms) that the elements in a system can assume and (ii) *state-transition functions* (or rules), which are meta-configurations (shapes, patterns, or forms) mapping configurations to configurations. The *total state* of a system is the pattern produced by its elements. Additionally, the basin of attraction field in a finite lattice CA (section 2.6.4) can be viewed as a Platonic form. (An infinite lattice CA whose states are algorithmically-compressible (Chaitin,90), that is, can be described using a representation of shorter length than the states themselves, also has a basin of attraction field - although its nodes are not states but the compressed representation of these states - and thus, can also be viewed as a Platonic form).

The main weakness of formism is that it does not lead to a systematic or unified

metaphysics. As Pepper (1942) states,

[Formists/Platonists] regard system as something imposed upon the parts of the world by other parts, so that there is an inherent cosmic resistance in the world to determinate order as well as a cosmic trend to impose it. (p.143)

2.7.3. Mechanism

The root metaphor of mechanism is the machine (Pepper, 1942). Angeles (1981) defines mechanism as the theory that all phenomena are physical and can be explained in terms of material changes, that is, matter in motion. There are two important aspects to mechanism:

- the whole is neither ontologically prior to the parts nor causally efficacious upon them, but merely the sum total (quantitatively and qualitatively) of the interacting parts.
- all phenomena can be explained in terms of the principles by which mechanistic systems, that is, machines, are explained without recourse to intelligence as an operating cause or principle.

This view is consistent with Runes (1960) who defines mechanism as the theory of total explanation by efficient, as opposed to final, causes (chapter 6). Broad (1925) lists four essential characteristics of a 'purely mechanistic' ontology:

- a single kind of *stuff*, all of whose parts are exactly alike except for differences of position and motion;
- a single fundamental kind of *change*, viz. change of position [or location]. (Changes of higher order - e.g. velocity, acceleration - are derived from the change in position);
- a single elementary causal *law*, according to which particles influence each other by pairs;
- a single and simple principle of *composition*, according to which the behaviour of any aggregate of particles, or the influence of any one aggregate on any other, follows in a uniform way from the mutual influences of the constituent particles taken by pairs.

Pepper (1942) provides a detailed description of the notion of mechanism beginning with a description of its basic categories:

Primary categories (effective)

- Field of location
 - Primary qualities
 - Laws holding for configurations of primary qualities in the field (primary laws)
-

Secondary categories (*ineffective*)

- Secondary qualities
- A principle for connecting the secondary qualities with the first three primary or effective categories
- Laws, if any, for regularities among secondary qualities (secondary laws)

Primary and secondary qualities may be differentiated as follows:

Such qualities as alone are relevant to the description of the efficient functions of a machine are historically called *primary qualities*. (p.192)

[Consequently,] qualities which are observed in parts of a machine but are not directly relevant to its action have been called *secondary qualities*. (p.193)

It is important to realize that what is to count as a primary quality is determined by function, that is, a goal (or *telos*). This should be obvious given the fact that machines - which provide the root metaphor underlying mechanism - are artifacts *designed* to fulfil human purposes; hence, the possibility of a connection between mechanism and teleology or intentionality.

Pepper identifies two basic kinds of machines: (i) *discrete*, in which action occurs by contact (for example, the push-pull machine) and (ii) *consolidated*, in which action occurs at a distance (for example, the orbits of planets arising from gravitational fields). These correspond to the two ways in which to view CAs: (i) *local* neighbourhood interaction between cells (FSAs) and (ii) *global* basin of attraction fields describing overall CA behaviour. Discrete mechanism is defined by a polarity of chance (or accident) and necessity: Chance results from the independence of details (for example, time independent of space, one atom being independent from another etc); necessity arises as a consequence of determinism (section 2.8.4). Chance can be identified in CA-computationalism (chapter 5) with the *definition* (or specification) of the CA substrate itself (section 2.7.2); necessity may be identified with the *operation* of the CA once it has been so defined. This is because both initial state and state-transition rule are *givens* (chapters 6 and 7) and hence, contingent when viewed from *within* the CA system¹³ whereas the evolution of CA phenomena is deterministic, that is, driven by necessity. However, according to Pepper (1942), discrete mechanism ultimately leads to *consolidated* mechanism:

¹³ However, if some version of the anthropic principle (chapter 6) holds, the selection of initial state and state-transition rule may be far from arbitrary (contingent), in fact, necessary and highly-specific. Yet, this position is also problematic since if the many-universes interpretation of quantum theory (chapter 4) is valid, it is simply the case that the universe in which human observers exist merely happens to be one universe among potentially many in which the emergence of observers is possible, thereby undermining the uniqueness of the universe inhabited by human beings. The significance of the *givenness* of initial state(s) and law(s) of evolution in the context of the distinction between naturalness (or natural phenomena) and artificiality (or artifactual analogues of natural phenomena) is examined in chapters 6 and 7.

Discrete mechanism is .. internally contradictory. It implies strict similarity and, consequently, the formistic categories. Indeed, not only are laws threatened with the status of forms, but also the atoms [for] what about the similarity of the atoms and the configurations ? The intention is to reduce similarities to configurations of ultimate differentiations of the spatiotemporal field - to draw nature completely and solidly into that field .. To achieve this end, however, [the mechanist] must consolidate his categories. The primary qualities and the laws must become structural features of the spatiotemporal field as intimately involved in it as the dimensions of space with one another. Similarity can then be relegated to the structure of that field and kept from flying into subsistent forms. (pp.211-212)

As will be shown in chapter 5, the shift from discrete to consolidated mechanism finds one of its most sophisticated expressions in the emergentist metaphysics of Alexander (1920), viz. matter, life and mind as Space-Time complexes. It is highly significant, as Pepper goes on to state, that "there are no laws in consolidated mechanism; there are just structural modifications of the spatio-temporal field. And there are no primary qualities, for these are resolved into field laws, which are themselves resolved into the structure of the field." (p.214) Furthermore,

a completely consolidated [that is, unified] universe would be a completely mechanized and internally determined universe. (p.207)

This is consistent with the computationalist position. However, as will be shown in chapter 7, a mechanistic view - and hence, by the same token, a computationalist view - of nature is untenable because of the problem of the irreducibility of secondary qualities to primary qualities (Russell,67). This problem, known as the mind-body problem (chapter 4), 'hard' problem (Chalmers,96) or category problem (chapter 7), is essentially identical to what Pepper (1942) refers to as the 'problem of unresolved discreteness', viz. the inability of mechanism to *explain* how and why secondary qualities should occur given that they are superfluous to a mechanistic ontology¹⁴. Adopting a more modest position based on a *hierarchical* view of phenomenal reality in which the world is merely *ultimately* grounded at the ontological level in a mechanistic monism - a position known as emergent-materialism or simply *emergentism* (Mayr,82) - does not resolve the problem, irrespective of whether or not emergents are held to be causally efficacious. However, as shown in chapter 6, this does not preclude the possibility of *some* (radical) emergentist position being correct. The connection between mechanism and emergentism is examined in chapter 3 in connection with an investigation into the meaning of computational emergence.

2.7.4. Determinism

Angeles (1981) defines determinism as the view that every event has a cause; further, that all things in the universe operate in accordance with causal laws, that is, that everything is absolutely dependent upon and necessitated by causes. For current

¹⁴ Secondary qualities are epiphenomenal and hence, non-causal on conventional mechanism.

purposes, a *cause* may be defined as 'something which brings something about', the latter something being referred to as an *effect* of the cause. A cause can have more than one effect in a world *provided the effects are logically and physically consistent*. For example, switching on an electric light brings about the illumination and the heating of a room; however, switching on a light cannot bring about both the illumination and the darkening of the room since in this case the effects are logically and physically inconsistent. Additionally, an effect can have more than one cause implying a many-one mapping between causes and effects which allows for the possibility of functionalism¹⁵. Aristotle identifies four kinds of causality, viz. material, efficient, formal and final causality (chapter 6); however, materialism recognizes only material and efficient causation which is mapped into the scientific language of particles and the physical laws governing their interactions. Formal systems (and computationalism) retain the idea of material and efficient causation in the notion of axioms (initial states) and inference rules (state-transition rules) respectively.

Philosophically-speaking, there are two kinds of determinism, viz. epistemological and ontological. A system is *epistemologically*-deterministic if its future state can be predicted from a knowledge of its current state and the laws governing the behaviour of the system: Two-body collisions in classical Newtonian systems are epistemologically-deterministic since the outcome of the collisions can be predicted from a knowledge of the position and momentum of the bodies and Newtonian mechanics. A system is *ontologically*-deterministic if effects follow *necessarily* from causes, that is, the relation between causes and effects is either one-one or many-one. (Here an effect refers to the state of a system at a particular instant which contrasts to its usage in the previous example, viz. switching on an electric light, where an effect refers to what might be called a sub-system state. The concept of systems and subsystems is described in chapter 3). Ontological determinism implies that the state of a system *s* at time *t* suffices to fix its state *s'* at time *t*+1 irrespective of whether a *description* of *s'* can or cannot be produced at *t*; current events are completely (that is, totalistically) caused by previous events. It is significant that systems can be ontologically-deterministic and at the same time epistemologically-non-deterministic: For example, *deterministic randomness* (Davies,87) is a characteristic of computational non-linear dynamical systems. However, epistemological non-determinism *becomes* epistemological determinism if the causal sequence of the system can be replayed. This latter point has crucial implications for the possibility of emergence in closed or finite systems as will be shown in chapter 3.

2.7.5. Atomism

The view developed by early Greek Philosophers such as Leucippus, Democritus and Epicurus (5th Century BC) that reality is composed of *atoms*, viz. minute material particles that are the ultimate constituents of all things having properties such as size,

¹⁵ However, as shown in chapter 6, Bunge (1959) maintains that the causal relation is one-one.

shape, position, arrangement and movement as intrinsic. Atoms are eternal, simple, separate, irreducible, unchangeable and infinitely many of them exist moving in empty space. Objects are formed as a consequence of collisions between atoms. (Angeles,81) This translates into the language of CAs as follows: Atoms are identifiable with lattice cells in the 'on' or '1' state (in CAs with binary or two-state FSAs), empty space as cells in the 'off' or '0' state; objects are identifiable with patterned groupings of cell states: For example, the glider structure in Conway's Game of Life (section 2.6.3) is a pattern of five cells in the 'on' state. In making use of the notion of pattern, this interpretation of CAs in atomistic terms ultimately becomes parasitic on formism. However, this is consistent with the claim that computationalism constitutes an eclectic, that is, synthetic, metaphysics (section 2.7.1).

2.7.6. A Note on Process Philosophy

All the previous metaphysical positions, and atomism in particular, assert the ontological primacy of substance over process, that is, of thing (being) over activity (becoming). However, as Rescher (1996) states, "in a dynamic world, *things* cannot do without *processes*. Since substantial things change, their nature must encompass some impetus to internal development. In a dynamical world, *processes* are more fundamental than *things*. Since substantial things emerge in and from the world's course of changes, processes have priority over things." (p.28) Thus, substances are redefined as manifolds or complexes of processes. Consequently, Rescher (1996) maintains that while a process ontology *can* be monistic, a substance ontology is *necessarily* dualistic: "processes without substances are perfectly feasible in the conceptual order of things, but substances without processes are effectively inconceivable." (p.46)¹⁶ A *process* may be defined as

a coordinated group of changes in the complexion of reality, an organized family of occurrences that are systematically linked to one another either causally or functionally. (p.38)

The primacy of process might appear to support an interpretation of computationalism as metaphysically processual: Computationalism is grounded in the notion of computation which can be understood in dynamic terms, viz. a computation is a program in execution or a *process*. This conceptual link between computationalism and process ontology is further strengthened because

the basic idea of process involves the unfolding of a characterizing *program* through determinate stages. The concept of programmatic (rule-conforming) developments is definitive of the idea of process: The unity/identity of a process is the unity/identity of its program [emphasis added]. (p.41)

Additionally, "the shift in orientation from substance to process - from a substantive unity of hardware, of physical machinery, to a processual unity of software, of programming or mode of functioning" (p.108) leads to the view that "processes, seen

¹⁶ This is a consequence of the problem of generating change from stasis which leads to the Zeno paradoxes.

abstractly, are inherently structural and programmatic - and, in consequence, universal and repeatable." (p.74) From such statements, it might appear that the philosophical precedent of computationalism is processism rather than atomism or mechanism. Furthermore, against formism (Platonism), it is held in process ontology that a physical object is identified as the thing it is, "not by a continuity of its material components or its physical *form* but by a processual or *functional* unity [emphasis added]." (p.52); it is unity of law or *functional* typology which is important as contrasted with unity of being (individualized specificity). Finally, process philosophy views nature as characterized by "creative innovation, productive dynamism, and an *emergent* development of richer, more complex and sophisticated forms of natural existence [emphasis added]" (p.101), which is consistent with the concept of computationally emergent artificiality (chapter 5).

However, identifying CA-computationalism with processism is problematic for (at least) five reasons: First, according to Rescher (1996), "the identity of things is discrete (digital); that of processes is continuous (analogic)" (p.53). On this view, CA-computation must be interpreted as non-processual and derivative of processes which are computationally-analogue¹⁷. If a processualism of this kind is adopted, CA-computationalism must be incorrect; second, upon closer examination, it is readily shown that the mechanism underlying processism is very different to that underlying computationalism. For example, Rescher maintains that

process philosophy rejects a pervasive determinism of law-compulsion. Processists see the laws of nature as imposed from below rather than above - as servants rather than master's of the world's existents. Process metaphysics envisions a limit to determinism that makes room for creative spontaneity and novelty in the world (be it by way of random mutations with naturalistic processists or purposeful innovation with those who incline to a theologically teleological position). (p.98)

Processes are not the machinations of stable things; things are the stability patterns of variable processes. (p.99)

To be sure, it is certainly possible to minimize the significance of novelty by conceiving of the role of process in nature as consisting in a fixed number of elemental process types themselves fixed for all time - through whose combination and interplay all other natural processes arise. But such a hard-edged, atomistically stabilitarian view of process does violence to the spirit of the enterprize of process philosophizing .. Processuality does not happen simply at the ground-flow level of things, events and phenomena. The *types* of items at issue can change as well. (p.81)

This is significant since states are defined as ontological absolutes in CAs: A FSA has a finite and static set of states and its functional connectivity to other FSAs (in a standard CA) is uniform (regular, symmetrical), finite and static; hence, the CA structures generated during execution are ontologically reducible to a finite set of static primitives.

¹⁷ On Rescher's view, such processes must be computational since processes are computational (though not necessarily discrete or digital) *by definition* (that is, essentially).

As will be seen in what follows (chapters 3, 6 and 7), this implies that the emergence of structures in CAs is either (i) static and finite ('closed') assuming a finite lattice or (ii) potentially infinite, yet constrained to formal constructions using a fixed alphabet in the case of an infinite lattice (Cariani,89) (Ali,98a); third, the notion of state does not appear within processism. In its place there is the notion of (inner) condition/structure, order or situation. This is highly significant since computationalism, being a variant of mechanism, is either (1) ontologically-externalistic or (2) *topologically*-internalistic (chapter 6); experiential-internalism (chapter 7) is excluded on this metaphysics or interpreted as epiphenomenal. However, processualism, unlike mechanism, is not definitionally-inconsistent with experiential-internalism; fourth, the space-time of processualism is not "a matrix of order imposed on natural process from without by the structure of a process-independent stage on which natural processes must play themselves out." (p.95) However, this classical Newtonian conception of space-time appears to be central to CA-computationalism (chapter 5): The notion of a Cartesian space-time grid is an ontological absolute in CAs; in a processual metaphysics, by contrast, the grid is emergent from physical processes; finally, as Rescher (1996) states, in atomistic systems "the *properties* of substances are never touched by change, which affects only their *relations*" (p.10) and this is certainly the case in CAs. As stated previously, the set of states of a FSA - the equivalent of substance in a CA - are static and finite.

The above arguments lead to the following position: While it may be reasonable to interpret computationally emergent artificial *phenomena* (for example, analogues of matter, life and mind) in processual terms, it is simply not the case that computationalism is consistent with a processual ontology (*noumenal* ground). As Rescher (1996) states, "nature's processes follow patterns - but not in a rigidly programmed and preordained *predetermined* way [emphasis added]" (p.82) and, as will be shown in chapters 6 and 7, this is inconsistent with the ontological determinism associated with computationalism. For this reason, formism, mechanism, determinism and atomism can be viewed as necessary and sufficient in order to classify the ontological essence of computationalism.

2.7.7. Computationalism, Metaphor and Metaphysics

The computer *is* a machine, but a machine with a flexibility derived by abstracting away its functional form; hence, the synthesis or eclectic mix of formism and mechanism in computationalism. This implies that computationalism is a permutation of two logical postulates, in this case of two metaphysical systems, thereby refuting Pepper's assertion of the mere *possibility* of the postulational method (chapter 1). However, it is interesting to consider whether metaphor production *could* occur in a computational substrate. As Pepper concedes,

at the break of the century, when the potentialities of the new symbolic logic were dawning upon men, there were some who expected that mathematical logical systems would yield all that traditional metaphysical systems had, and more too, and would therefore in time completely supplant the

traditional modes of metaphysical thought. These hopes have waned. But the possibility still remains of using the apparatus of symbolic logic as a means of generating world theories. (pp.87-88)

Lugowski (1989) describes a framework for generating metaphor using a "metaphor for metaphors" based on "meta-stable dynamical systems that shift amongst contextually-cued attractor states" (p.355), a notion which readily translates into the language of CA-computationalism (section 2.6.4). Given that metaphysics is ultimately metaphorically grounded and accepting Lugowski's claim for computational metaphor generation, what must a computationalism that supports metaphor (and hence, metaphysics) generation look like ? In other words, what kind of computationalism could give rise to the *concept* of computationalism ? Johansson (1993) argues in favour of defining computationalism in connectionist or fuzzy logic terms, such a position being motivated by the view that information processing in the brain must be 'poetic' in nature if the intrinsic vagueness or indeterminacy associated with metaphors is to be addressed. A simpler approach might involve embedding a 'virtual' machine capable of supporting universal computation in a computational substrate (chapter 5), leading, perhaps, to what might be referred to as 'metaphysics via computation'. According to Pepper,

the idea is to conceive a world theory in the form of a deductive system with theorems derived from postulates. Once [we] obtain such a system .. new world theories might then be generated like geometries by simply adding or dropping or changing a postulate and noting the result in the self-consistency of the system and in the application of the theorems to all the observed facts of the world. (p.88)
