SPECIFICATION and COMPUTATION

The Abstract and the Physical
Specifications of Computations

- Abstract Machines
- Programs
- Specifications in specification languages
- ...
- ....
Putnam’s Puzzle

• What is it for a physical device to be an implementation of an abstract one?
• Presumably, the more computational details that are imposed by the abstract one, the less physical interpretations are possible.
• However, under a certain natural notion of implementation, almost every physical device implements every abstract one.
• We shall examine this puzzle from the perspective of the computer science notion of specification.
SPECIFICATION and IMPLEMENTATION
Normative Force

- A specification tells us what to build.
- Fixes the notion of Correctness for the artefact.
- The artefact maybe abstract or physical but here we are mainly concerned with physical artefacts.
Correct Implementations

- When is a physical device to be taken as a *correct* implementation of an abstract machine?
- What must be the relationship between the specification and the artefact, between the abstract and the physical?
Putnam’s Account of TM Implementation

A ‘machine table’ describes a machine if the machine has internal states corresponding to the columns of the table, and if it ‘obeys’ the instruction in the table in the following sense: when it is scanning a square on which a symbol $s_1$ appears and it is in, say, state $B$, that it carries out the ‘instruction’ in the appropriate row and column of the table (in this case, column $B$ and row $s_1$). Any machine that is described by a machine table of the sort just exemplified is a Turing machine. (Putnam 1960/1975a, 365; cf. also Putnam 1967/1975a, 433–4)
Simple Mapping Account

A physical system $P$ Implements an abstract one $A$ just in case:

I. There is a mapping $I$ from the states of $P$ to the states of $A$.

II. For any abstract state transition $s_1 \Rightarrow s_2$ : if the system is in the physical state $S_1$ where $I(S_1)=s_1$ it then goes into the physical state $S_2$ where $I(S_2)=s_2$. 
Not Gate

- During any time interval, a rock's temperature rises. The rock goes from temperature $T$ to temperature $T+1$, to $T+2$, to $T+3$.
- Now consider a not gate that feeds its output back to itself. The not gate goes back and forth between outputting a ‘0’ and outputting a ‘1’.
- Now map physical states $T$ and $T+2$ onto ‘0’; then map $T+1$ and $T+3$ onto ‘1’.
- According to the simple mapping account, the rock implements a not gate undergoing the computation represented by ‘0101’.
Every Physical System Implements Every Specification

In the absence of restrictions on which mappings are acceptable, such mappings are relatively easy to come by: every physical system (with enough parts) implements every specification (Putnam 1988, Searle 1992).
Restrict Mappings

One way to construct accounts of computation that are more restrictive than the SMA is to impose a constraint on acceptable mappings. There are two main approaches in the literature:

1. Causal: there must be causal connections between the physical states
2. Semantic: no computation without representation

Both contain an element of truth. Our specification perspective throws some light on matters.
THE CAUSAL RESPONSE
Causal Restrictions

- Chalmers/Copeland develop their rebuttal of SMA in terms of *causal* connections. Specifically
- The second part of (ii) is taken to be a material conditional. They take it to be a counterfactual.
- This may be seen as a covering law that the abstract machine lays down about the physical one.
- This interpretation is appropriate for natural artefacts.
A Physical device

We find a device in nature.
Natural Computers

We experiment and find:

1. Input 0 volts and 0 volts.....output 0
2. Input 0 volts and 5 volts ....output 0
3. Input 5 volts and 0 volts ....output 0
4. Input 5 volts and 5 volts ....output 5
Theory Construction

- We claim that it is a logical gate i.e., we build a *theory* about it i.e., to see a physical object or system as a computer is to entertain a *theory* about it.
- **and** gate where 5 is interpreted as true and 0 as false
- **Or** gate where 5 is interpreted as false and 0 as true
Explanation and Prediction

- An interpretation that is only extensionally correct in the sense of the SMA provides no explanation of why the physical device works.
- On the theory proposal, explanation emerges from a uniformity that is built into the generic nature of the theory.
- Correctness is now an empirical claim about the abstract (machine) as a theory of the physical one. This seems to be the underlying force of the causal response.
Mechanistic Accounts

• The theory perspective has links to the mechanistic approach e.g. mechanistic systems in the sense of molecular biology. (Piccinini).

• Here explanation is linked to uncovering the hidden mechanisms of a physical system.

• Indeed, formalisms from TCS (Π calculus) are being used to formalise such mechanisms.
Practice

- In practice, we do not find or discover computational devices we build them.
- One starts with a specification and constructs/builds a physical implementation.
- The actual physical device is then checked against the demands of the abstract specification.
Testing and Verification

- We run tests to determine whether the physical device is *correct* relative to the abstract one.
- Tests are not usually exhaustive. We do not, and generally cannot, consider every possible state transition.
- Very often the specification, or a codicil to it, lays down the suites of testing procedures.
The Empirical Nature of Correctness

- Empirical content emerges from this testing and verification.
- Correctness is taken to be an empirical claim about the physical machine.
- This is the methodological picture for constructed artefacts.
- But this is a very different picture to that provided by the SMA. Indeed, it puts flesh on Copeland’s objections to SMA.
Ex Post Facto

- Copeland argues that the mappings that SMA relies on are illegitimate because they are constructed after the computation is already given.
- In the case of genuine computational descriptions the work of generating successive descriptions of a system's physical dynamics is done by a computer running an appropriate program not by the mapping relation.
The Difference

- The difference between the theory perspective and the specification one concerns what is being tested.
- The same abstract/physical pairing can be seen as either.
- The theory perspective is appropriate when the artefact is natural; the specification one when artefacts are being built.
- But when things go wrong the engineering perspective blames the artefact; the scientific one the theory.
Changing Perspectives.

- Observe that once we see a natural artefact as a computer, the theory is transformed into a functional description— one that enables us to control and employ it.
- On the other hand, when a physical system evolves we may need to rethink its functional description; we may need to subject it to scientific investigation.
The Abstract and the Physical

- The appropriate relationship between the abstract and the physical is not given by SMA but by the methodologies of specification/theory, testing and verification.
- This is an epistemological rather than an ontological account of computation.
- SMA does not offer anything like a characterisation of any naturalistic notion of computation.
THE SEMANTIC RESPONSE
No Computation Without Representation

- Computations must be individuated by some semantic information.
- This is intended to restrict the class of possible mappings.
- But what does semantic mean?
The Syntactic View

- Computation is syntactic
- Who or what makes the judgement that something computes?
- Who or what makes the judgement that something has meaning?
- Rules of Lambda Calculus - rules of set theory
Internal Semantics

- The abstract device provides the rules that the physical device must obey.
- It is normative.
- It provides an operational or internal semantics for the device.
- An abstract machine with a simple machine language would have its machine language given by an internal semantics.
External Semantics

- However, while the internal rules tell us what each of the machines basic operations do, they do not say what the physical machine or computation does (or supposed to do).
- For example, we might need to spell out the specification of an operation that swaps the values in the store of a machine i.e., computations picked out by programs need to be specified over and above the internal semantics.
Levels of Abstraction

- But this distinction, although used by some to argue for the semantic account, is the traditional distinction between the abstract machine and the specification or definition of programs that run on it.
- It is a distinction based upon levels of abstraction: the programs of one language are the basic instructions of another.
- In general CS operates with a notion of levels of abstraction where each level is implemented by the next.
- But whichever level of specification is adopted, we must establish that specifications restrict the class of SMA interpretive mappings.
- It is not obvious how they do so.
Are the computations in David Marr’s theories about physical objects in the visual system?

Or are they completely characterised in terms of the ontology of the programming systems in which the computations are primarily articulated?
Functional Specifications

- There is a distinction between functional and non-functional specifications.
- Are all computationally relevant aspects captured by the functional ones?
- Insisting that the colour of the tape on a physical Turing machine be yellow does constrain the class of mappings.
- But this does not seem to be *computationally relevant*. 
Functional Specifications?

- Functional specifications describe the underlying mechanisms in an *abstract way*.
- They are normative for the construction of physical artefacts.
- But they are generic and so constrain the physical artefact only to the extent that the axioms are satisfied.
- But do they constrain matters to block SMA?
Supervenience

- A supervenes on a set of properties B, if and only if any two objects x and y which share all properties in B (are "B-indiscernible") must also share all properties in A (are "A-indiscernible").
- Do the high level specifications supervene on the low level ones?
- Do they always do so in CS?
- Refinement relationship