John Haugeland – *Semantic Engines: An Introduction to Mind Design*

1. Cognitive Science

- The earliest expression of the *computational view of thought* is due to Hobbes. It is now the single most important theoretical hypothesis in psychology, and the basis of *Artificial Intelligence*. *Cognitive Science* is used as an umbrella term to cover various enterprises (not every scientific theory of cognition) that share the conceptual foundations of *information processing* or *symbol manipulation*.
- The purpose of this chapter is to survey the field and provide background for the other dozen papers on the philosophy of cognitive science that go to make up the book *Mind Design*.
- AI forms a distilled essence of cognitive science, and so attracts the spotlight. The issue is not to do with wetware versus hardware, but whether or not we're considering something designed and specified in terms of a computational structure. The guiding inspiration of cognitive science is that at an appropriate level of abstraction, a theory of natural intelligence would have the same basic form as those explaining sophisticated computers. This makes AI a central and pure form of *psychological* research.
- Haugeland attributes the 300-year delay in taking up Hobbes's insight to the three following dilemmas:
  1. *Metaphysical* problems of mind interacting with matter
  2. *Theoretical* problems of explaining the relevance of meanings without question-begging recourse to a homunculus.
  3. *Methodological* problems of the empirical testability and hence respectability of mentalistic explanations.

  Haugeland claims that the computational idea "slices through these three dilemmas at a stroke", and that this explains its gut-level appeal.
- Haugeland now briefly discusses Descartes's metaphysical dualism, with mind and body being seen as completely different *kinds* of substance, and rehearses their characteristics, saying that dualism has a more intuitive appeal that materialism, its main rival. We find it difficult to ascribe thought and feelings to mere matter, and have difficulty imagining minds and ideas having shape or location. However, materialism avoids the problem of interaction between the mental and the physical. Perception would be impossible in its absence, yet no dualist has an answer to how it works – and mental causation would appear to break the laws of physics, since the motions of bodies are supposed to be completely specified by the physical laws that govern them.
- Is there an analogy between laws of thought and laws of physics? Haugeland thinks not, because we have to struggle to keep ourselves rational while matter obeys physical laws without struggle. So, the roles of rules and laws in the two cases are deeply different. In particular, the semantic aspect – what the thoughts mean – affects the correct application of the rules of reason. Haugeland asks whether this requires some rule-applier that polices the application of the rules like a traffic cop. As an *explanation* of thinking, the introduction of any such homunculus begs the question, for the homunculus itself would have to think, and it is thinking that we're trying to explain.
- The unobservability and essential subjectivity of thought provides the motivation for *behaviourism* as the only thoroughly scientific account of the mind, since otherwise there's no check on the hypothesised intermediates (thoughts, desires,

beliefs) used to explain behaviour. However, Haugeland thinks that trying to explain behaviour without invoking mental processes is just too hard, as is exemplified by the lack of success of the behaviourists. We can't even describe, let alone explain, the phenomena of behaviour without recourse to mentality.

- Consequently, cognitive scientists can be mentalists (non-behaviourists) as well as materialists (non-dualists), and can offer explanations of behaviour in terms of meaning and rule following without recourse to a homunculus.
- Haugeland now gives the example of a chess-playing computer. It has no soul (and no-one would suggest it had), it can cause physical behaviour without violating the laws of physics, and it follows both the rules of the game and rules of thumb (to find the best move) without the need for a homunculus. Additionally, Haugeland can see an analogy between mental states and the internal states of the chess-computer. Basically, we encounter the same difficulties that encourage behaviourism. It is difficult to determine or design processes that consistently give the observed or desired behavioural response, but nevertheless, for any given machine-occasion, there is a determinate answer about which reasonings it in fact went through.
- This symbolises the inspiration of AI, that people just are computers, though Haugeland admits that nothing is that simple.

2. Formal Systems

- Haugeland defines a computer as an automated formal system, so he then proceeds to explain what he means by a formal system before considering how one might be automated.
- ***Formal systems*** are like games in which tokens are manipulated according to rules. Three things need to be specified: (1) the tokens, (2) the starting position and (3) the legal moves. Sometimes there's also a goal or winning position. He gives the example of solitaire.
- Formal systems have three important properties, which makes them *digital*:
    1. They are self-contained.
    2. They are perfectly definite. There are no ambiguities, approximations or judgement calls about whether a position or move is legal.
    3. They are finitely checkable. Proof of legality only requires a finite number of things to be checked.
- Digitalness has the important consequence that two systems that seem quite different can be essentially the same. Haugeland gives the example of a domino-type game analogous to solitaire. This is called formal equivalence, defined as follows:
    1. There's a 1-1 correspondence between positions.
    2. The starting positions correspond.
    3. If two positions are mutually accessible in one system, so are the corresponding positions in the other.

  This is slightly too constraining, but shows that equivalent systems can be very different on the surface provided appropriate correspondences can be found.
- Haugeland finds two forms of complication within formal systems, illustrated by chess and algebra:
    1. There can be different types of token, tokens of the same type being formally interchangeable (as pawns in a chess-set), with these types being determined

by the rules for movement. These types must be perfectly definite and finitely checkable for the system to remain digital.

2. The positions of one lower-level formal system can act as the tokens in another higher level formal system. In algebra, the tokens are equations or formulae, and the positions are lists of them, with the axioms forming the starting position and the rules governing the generation of new formulae (theorems) to add to the list. There are infinitely many such formulae, so how are they finitely checkable? The answer is that algebraic tokens (formulae) are positions in another game – that of well-formed formulae, and these are finitely checkable. Only the *structure* of the molecular tokens is important, so we need only finitely many rules (eg. cancel anything that appears on both sides of an equation). Tokens in the wff game are called atomic tokens, to distinguish them from the tokens in the algebra game, known as molecular or compound tokens.
3. Haugeland points out – from the solitaire/domino isomorphism – that there can be a trade-off between positions and tokens in different formulations of the same game. Hence, the number of types of token isn't a deep property of a formal system.
4. The nesting of one formal system within another makes them easier to understand.

- It may seem odd to include algebra and logic in the same category as chess, because the tokens in the first two games *mean* something while chess-pieces don't. They have the same kind of formality as games and as formal systems are on a par. Meaning has to do with the outside world, and has nothing to do with self-contained formal systems. There are other points of view, dealt with by interpretation and semantics.

3. Axiomatic Formal Systems (Turing Machines and Computers)

- An *automatic* formal system (AFS) is a physical device that automatically manipulates the tokens of some formal system according to its rules. This is what computers are. There are two fundamental problems:
    1. Getting the device to obey the rules (solved).
    2. The control problem: how the device selects which of several legal moves to make in a given position. This is considered in §4.
- Haugeland now describes a Turing machine (TM), in a rather idiosyncratic manner. His version has:
    1. An unlimited number of *storage bins*. Each can contain only one formal token.
    2. A finite number of *execution units*. Each has it's own rule, obeyed when active, which depends on the contents of the in-bin and specifies the next active execution unit and which token to put in the out-bin[1]. There's usually an execution unit that does nothing, at which point the machine stops.
    3. One *indicator unit*. Which shows the active unit and input & output bins.
- This differs from Turing's own definition (Haugeland claims it's a generalisation. For Turing:

[1] Isn't there something wrong here? What specifies, rather than indicates, what the next out-bin and in-bin will be?

1. The storage bins are connected into a tape. Each bin can hold only a single simple token.
2. The in-bin and out-bin are the same bin, and this bin is next to the previous one.

The important thing is that, apart from convenience and simplicity, the difference of formulation makes no difference[2].

- Haugeland explores the relationship between Turing machines and Formal Systems.
  1. Any Turing machines is an automated version of some formal system. Initial contents of bins = starting position; moves = machine steps; the rules are those obeyed by the execution units (he now says that the control problem is solved by rules for resetting the indicator unit).
  2. Any formal system can be formally imitated by some Turing machines .

  Haugeland takes (1) to be obvious and (2) is Church's Thesis, which has been proved for all known well-defined automatic formal systems. The failure to find a general proof is due to the vagueness of "automatic formal system".
- Formal imitation is like formal equivalence, except:
  1. The imitating system has to *choose* the next move. Formally equivalent systems that make the same choices are *dynamically* equivalent.
  2. The imitating system tends to be split into two parts: the *virtual machine* and the *program*. The former directly corresponds to the system being imitated, the latter solves the control problem.
- Haugeland gives an example. A Turing machine T is formally imitating some automatic formal system A. T is made up of:
  (a) a virtual machine V, to which a portion of T's storage bins are dedicated, and which is dynamically equivalent to A. The tokens occupying the bins constitute V's (and correspond to A's) positions.
  (b) The rest of T's bins contain the program and workspace. The program is required because V's rules will differ from T's rules. The program is a finite set of tokens such that when T obeys its own rules using all the bins, it obeys V's rules when using V's bins. The program translates V's rules into T's rules, thought for this to be possible, T's rules need a certain level of versatility.
- The important conclusion is that for any formal system, there is a Turing machine that can be programmed to imitate it.
- Modern computer science has been shaped by Turing's Theorem – that there are special Turing machines – *universal* Turing machines – that can be programmed to imitate any other Turing machine. In particular, a UTM can imitate a TM that is itself imitating an AFS, so that the UTM is indirectly imitating the AFS. Hence, from the Church-Turing thesis, a single suitably programmed UTM can imitate any AFS whatever.
- There are universal machines that aren't formally Turing machines, but since they are universal, they can formally imitate any Turing machine, so it doesn't matter which we use. The important point is that computers just are such universal machines, with the proviso that the quantity of storage, though large, is finite.
- Most programs are written for *virtual* machines, which are themselves being imitated by the hardware machine. The reason is that it is cheaper to build the hardware a certain way and then write compilers so that the hardware can imitate

[2] There seems to be something wrong about his version of Turing's definition as well. Look at Searle (or Turing's own paper).

the software language. There are usually further levels of imitation between the hardware and the programming language, and, of course, the final program is itself imitating some AFS. Since this final machine has been designed for a specific purpose, it will not usually be a universal machine.

- A single physical machine can, at various levels, be imitating any number of other machines, and it makes no sense to ask which one it is "really" imitating. A particular object is a particular machine or formal system only at a certain level of description. This shows that it is foolish to say that computers are "really" just bit-manipulators. This is so at one level, but is of no theoretical consequence.

4. The Control Problem

- Automatic formal systems need to come up with a determinate move, yet many possibilities are open at each step. The decision about which legal move to be made also needs to be automated. We can think of the machine as divided in two separate virtual machines – one generates legal moves, and the other chooses one. Designing the second machine is the *control problem*, the hardest part of automating an interesting formal system.
- Haugeland illustrates the problem by considering a chess computer. We can't just "crunch" the problem because of the *combinatorial explosion*. We need to select only *relevant* alternatives, but don't want to miss brilliancies that only appear to be blunders. We must distinguish *heuristics* from *algorithms*.
  1. An **algorithm** is a rule or procedure which is *guaranteed* to produce a result meeting the required conditions.
  2. A **heuristic** is a rule of thumb that is more or less reliable, but not infallible. Intuition and inspiration may be unconscious heuristics, a point of dispute amongst psychologists.
- In one sense, the rules of automatic systems *must* be algorithms, since they have definitely to determine the machine's next step. Whether the procedure is classified as an algorithm or a heuristic depends on how the desired result is *specified*. Losing a queen for a pawn is usually a blunder, so a reasonable heuristic for winning chess games is "don't exchange your queen for a pawn". However, occasionally a queen sacrifice is a brilliancy, so the rule is only a heuristic when the result is specified as "find the best move". The rule is a trivial algorithm if the result is specified as "avoid losing your queen for a pawn". The rules followed by computers only *have* to be algorithms in this trivial sense, of being algorithms for the heuristics – with the machine infallibly following fallible rules.
- The truth of the claim that a proper-functioning computer can never make a mistake depends on what counts as a mistake – it may never make a mistake following its internal heuristics, but make many mistakes in *the game*.
- Much of AI involves finding *better* heuristics, because almost any method of making a choice counts as a heuristic. *Better* in the sense of more efficient, harder to fool or easier to implement (ie. better algorithms for implementing them).
- However, not all forms of intelligent behaviour have such well-defined goals towards which heuristics are supposed to guide the system.

5. Digital and Analog

- Digital systems are self-contained, perfectly definite and finitely checkable, all according to which moves are, according to the rules, legal is which positions.

- By way of contrast, an analog device doesn't have clearly-defined moves, rules or positions, though it may have states. The "perfectly definite" condition is absent from analog devices, so that it's often unclear what the current states is and what it's supposed to be doing. Even when it's working normally there will be slight inaccuracies and marginal judgement calls. Haugeland's example is of the difference between a switch and a dial.
- Analog devices have value – Haugeland considers scale models, slide rules, the use of laboratory animals and such-like.
- The advantages of (universal) digital systems are (a) versatility, (b) they can often be digitally simulated, making them even more versatile.
- Haugeland thinks grossly misleading the view that any analog device can be digitally simulated to any degree of precision. This is only true when there are relatively few variables and equations, so the simulation can be described in a compact and precise way. Often gross simplifications need to be made to enable a simulation to run at all, and simulating laboratory rats for biomedical testing is out of the question.
- The theoretical advantage of digital systems is to reduce error propagation, especially in large, complex, interdependent systems.

6. Semantics

- Haugeland now distinguishes between formal systems and their *interpretations*, which relate them to the outside world. We can say that the tokens of a formal system *mean* something – that they are *symbols* that *represent* something. An ***interpretation*** is a regular, systematic specification of what all the tokens of a system mean. ***Semantics*** is the general theory of interpretations and meanings. What a token means, and whether it is true or false[3], are semantic properties.
- Semantic properties are not formal properties, and are irrelevant within a formal system, which is self-contained and meaningless. The purely formal and structural characteristics of a system is its ***syntax***.
- So, formal tokens have a dual aspect – the formal / syntactical and the semantic. The relationship between the two aspects is fundamental to modern mathematics and logic, and is the inspiration of cognitive science.
- Haugeland considers the algebra game. It's tokens are true or false relative to an interpretation when compared to the world. The rules of such a system are truth preserving iff any tokens added according to the rules to any position that contains only true tokens will themselves be true.
- The rules of standard logical and mathematical systems are truth preserving relative to their standard interpretations, so guarantee that all theorems are true, given that the axioms (tokens in the starting position) are all true. *If you take care of the syntax, the semantics will take care of itself.*
- Haugeland points out the impact of Gödel's theorem, that no *consistent* formalisation of arithmetic (and, for that matter, many other axiomatic systems) can be *complete*. That is, while all its theorems are true, not all its true tokens are theorems. This is generally agreed to be irrelevant to cognitive science[4].

[3] Do all tokens have such semantic properties? A theorem maybe, but is a chess piece true or false?
[4] See Segal's lecture notes.

- Most mathematical and logical systems are only partially interpreted, since some of their atomic tokens are variables. The important thing is that the rules are truth-preserving whatever interpretation is given to the variables.
- A good example (apart from algebra) of the semantics taking care of itself is a computer. The machine doesn't have to pay attention to semantics, but given an appropriate formal system and interpretation will automatically produce only truths.
- Dennett called an automatic formal system with an interpretation such that semantics takes care of itself a ***semantic engine***. The inspiration of cognitive science and AI is the discovery that semantic engines are possible – that with the right kind of formal system and interpretation, a machine can handle meanings. However, people are both less and much more than automatic truth-preservers.

7. Interpretation and Truth

- In logic and mathematics, the only semantic property that matters is *truth*. While there are other cases where semantic properties other than truth are important in judging interpretations, Haugeland stays with his algebra game for the time being.
- He considers some formally OK but gibberish algebraic tokens (not wffs), under the usual Arabic interpretation. He then considers a non-standard interpretation, but one that converts the tokens into formulae that look like theorems, but which are all false. Finally, he gives an interpretation of the symbols that leads to true equations. This intuitively inclines us to think the final interpretation is "better" than the others, and Haugeland draws out three theoretical considerations from it, and why "truth matters":
  1. Within a formal system, one interpretation is as good as another – all are equally extraneous and arbitrary. So, an interpretation that comes out true provides a special and non-arbitrary choice amongst interpretations
  2. The choice of a true interpretation reflects a genuine relationship between the system and what it is interpreted to be about, allowing inferences both ways between the theorems and the world.
  3. Most importantly, truth matters because wild falsehood amounts to nonsense, the antithesis of meaning. Random interpretations aren't really interpretations at all. An occasional lapse may be understood as meaningful but false, but ubiquitous falsehood means that the tokens have no meaning at all.

8. Interpretation and Pragmatics

- Truth is not the only important semantic property. Haugeland gives many examples of speech acts – questions, commands, expletives, banter, quips – which have no truth values, so some other appropriateness must be what matters. Even in the case of *statements*, such factors as relevance, politeness, and avoiding making an ass of oneself, are important. Some authors prefer to call such conditions pragmatic rather than semantic, but they are all relevant to the acceptability of an interpretation.
- It isn't possible to give a precise definition of what it is for "making sense" to be our criterion for adequate interpretation. Some, such as Davidson, stick to truth, but most agree that further considerations such as the following rough characterisations are relevant conditions for conversational interaction with the automatic formal system "making sense".

1. **Rationality**: (a) there should be easy generation of obvious, logical and common-sense consequences of tokens in the current position and (b) a tendency to alter or eliminate tokens that are in tension or incompatible with the current position. Haugeland admits that the notion isn't tightly defined.
2. **Reliable interaction with the world**: this is via *transducers*, which are kinds of automatic coders and decoders (analogous to sense organs and motor systems) that interact with the physical environment and which add tokens (via inputs) or produce behaviour (via outputs). Tokens corresponding to inputs should be correct, and actions should correctly correspond to the tokens that were the decisions to act. Quine refers to the preservation of "stimulus meaning", though doesn't note a corresponding constraint on action.
3. **Conversational cooperativeness**: Grice. False but relevant answers make more sense than irrelevant truths. The basis is answering the question asked, but there are subtler factors such as manner of response, fit to the wider picture, rudeness, humour, obviousness. Cooperativeness is a matter of degree, involving judgement and compromise, but if totally lacking it leads to conversation becoming nonsense.
4. **Felicity Conditions**: Austin (Searle – *constitutive rules*). Certain speech acts or performances have various prerequisites or presuppositions which, if unsatisfied, mean that there's something wrong with the performance. Haugeland's examples are making unkeepable promises, threatening someone with what they want, commanding without authority, proposing to one's spouse and suchlike. Occasional violations may just be foolish or dishonest, but wholesale disregard leads to nonsense and the interpretation will be deemed unsatisfactory.

- The moral of all this is that failure to make sense implies a faulty interpretation[5]. A successful interpretation of an AFS is finding a way of assigning meanings to its outputs so they make consistent sense given prior inputs and other outputs. Haugeland admits that the above motley "nonasininity-preserving" list of further conditions for making sense doesn't hold out much confidence that the deepest challenge to cognitive science will be quickly solved.

9. Cognitive Science (Again)

- According to cognitive science, intelligent beings just ***are*** semantic engines[6]. That is, they are AFSs with interpretations under which they consistently make sense. If this is true, it puts psychology and AI on an equal footing with people and intelligent computers (if they can be built) being different forms of the same phenomenon. Any semantic engine can be formally imitated by a computer (a universal machine) if we can find the right program. This will guarantee semantic imitation since semantics takes care of itself given appropriate formal behaviour. So, according to cognitive science, AI is just a less messy form of psychology, with all the variables under the experimenter's control.
- There can be two responses to such strong claims of cognitive science. Either (a) people ***aren't*** semantic engines or (b) robotic semantic engines ***aren't*** genuinely

[5] I thought Haugeland had dismissed the notion of the "principle of charity" as a name for the subject under discussion. However, this seems to me to be a relevant title, because the rationale behind the principle is that if your opponent doesn't seem to be making sense, it's because you've adopted the wrong interpretation of his words.
[6] Does this necessarily involve language?

intelligent. There are two main arguments to the effect that cognitive science is fundamentally misconceived.

1. **Hollow Shell**: however well a semantic engine acts as if it understands, it can't really do so because it's missing some vital ingredient "X". A robot based on a semantic engine would be a sham no matter how good it got. Candidates for "X" are ***consciousness***, ***original intentionality*** and ***caring***.
2. **Poor Substitute**: semantic engines aren't even capable of acting as if they understand, and aren't going to get good enough to do so.

- The hollow shell argument is a priori, whereas the poor substitute is a posteriori and open to a "wait and see" strategy[7].
- Haugeland now clarifies and responds to the **Hollow Shell** arguments:
  1. **X = Consciousness**: however smart computers might seem, they can't really understand anything because they aren't conscious. It's natural to suppose something difficult and important has been left out. Haugeland responds by saying that no-one understands consciousness – it's mysterious regardless of your viewpoint. So, how do we know (a) that understnding is impossible without consciousness or (b) that semantic engines won't ever have consciousness?
  2. **X = Original Intentionality**: a semantic engines tokens only have meaning because *we* give them to them; their intentionality is derivative, and they don't mean anything by their tokens any more than books do. Genuine understanding is intentional in its own right and isn't derived from something else. Haugeland's response is that actions prove intentionality. Books don't act, so their intentionality *is* the author's, but this isn't a real counter-example. Robots might be made which act in an intelligent, self-motivated and responsive manner – and seem to act on its opinions, defend them or modify them, which would suggest that they were the robot's *own* opinions. So, the response is similar to the last one; original intentionality depends on whether the object has a suitable structure and / or dispositions relative to its environment, so how do we know semantic engines can't have it?
  3. **X = Caring**: another intuition is that a systems can't really mean anything unless it's beliefs *matter* to it, otherwise it's just generating tokens mechanically. Haugeland thinks this view is just begging the question, since if cognitive science is on the right track, some semantic engines – namely people – can care. So, if we had a robot that *seemed* (in appropriate circumstances) to be sympathetic, embarrassed, disappointed and so on, it would be difficult to claim it was just an unfeeling hunk of metal. We need an intuitive test for what it *is* to have inner qualities if we are justifiably to claim that a semantic engine that acts as if it were intelligent is just a hollow shell.
- Haugeland is more impressed by the **Poor Substitute** strategy. It's argued that computers do well at formal games and such-like, but it is an open question whether they can be programmed beyond game-playing to manifest the sort of intelligence used in everyday life, let alone art, invention and discovery. Cognitive science supposes that they are the same sort of thing as game-playing, but the evidence isn't there yet. We can ask the "X" questions – whether in fact it displays them, and if it doesn't, ask the important question whether there's any pattern of failure that might reveal the source of the problem or the result of the search.

[7] Though a poor substitute objector might then retreat to a hollow shell argument if computers (say) pass the Turing test.