

Miser Project: Interpretation, Representation, Computation, and Manifestation | Orcmid's Live HideOut

Thursday, February 14, 2019 12:43 PM

Clipped from: <https://orcmid.wordpress.com/2019/02/11/miser-project-interpretation-representation-computation-and-manifestation/>

"When a Mathematical Reasoning can be had it's as great a folly to make use of any other, as to grope for a thing in the dark, when you have a Candle standing by you."
— John Arbuthnot, *Of the Laws of Chance*, 1692.

Miser Project conception starts from reasoning with and in certain mathematical structures. That foundation brings to bear mathematical thinking in establishing a model of computation and subsequent reasoning about programs and their dependable employment for practical purposes. It is not necessary to digest all of the mathematics in order to develop and use the eventual software, provided there is trustworthy assurance of suitable quality, an objective of the mathematical route: *someone* has done it and that undertaking is transparent and trustworthy.

I strongly recommend the wider and practical context presented in Edward Ashford Lee's recent book, [Plato and the Nerd](#). For the Miser Project, "theory" is often used in place of Lee's use of "model." The Miser Project use of "model" is reserved for a mathematical- logic situation around interpretations of theories, perhaps a [third flavor](#) of modeling (or theory-building) with respect to the marvelous utility of computers as instruments of human purpose and experience that Lee addresses.

The [Eugenia Cheng](#) book, [The Art of Logic: How to Make Sense in a World that Doesn't](#) looks critically into affairs of the world from a mathematician's perspective. More profound than the narrow application of mathematics in the Miser Project conception of computing, Cheng's video on "[How to Think Like a Mathematician](#)" reveals some connections with the structures portrayed in [boole.txt](#) and the [demonstration of representations and interpretations](#) related to <ob>.

In the course of the Miser Project and demonstration of computational power and limitations, some specialized concepts are depended on. They are elaborated here.

2019-02-11 Initial Public Draft: Lengthy article navigating some concepts that will arise continuously in narration of the computational model and its representation in computational processes. For reference and review as background to further articles.

Preliminaries

- i. [Hark! Is That a Theory I See Before Me?](#)
- ii. [<ob> Primitive Functions](#)
- iii. [Narrowing <ob> for Computational Interpretation](#)
- iv. [Representing Data as Obs](#)
- v. [Representing Functions in <ob>'s Abstract World](#)

1. Recap: Computation-Model Stage-Setting

At this point, these articles address $\langle ob \rangle = \langle Ob, Of, Ot \rangle$ having a domain of discourse that is a flavor of abstract data structure, strictly on theoretical terms.

With regard to computational interpretation, effective-computability is taken as a condition on deductions that are available given a particular mathematical representation of a function or predicate. So far, the transition from an effectively-computable representation in any structure to an operational computational process of any kind is suggested, not demonstrated. We have not escaped from mathematical entities so far in this series of articles, apart from one [programming-language mock-up](#) of an $\langle ob \rangle$ computational interpretation.

[Interpretation of other structures](#) and their effectively-computable primitives via representations in $\langle ob \rangle$ sets the stage for computational interpretation satisfying the $\langle ob \rangle$ computational model. Such transition from other/higher levels to ones tied to the lower-level computational model demonstrates generality of computational interpretation.

It is posited that the interpretation-in- $\langle ob \rangle$ approach is sufficient for computational representation of the same effectively-computable mathematical functions as any other recognized model of computation. Demonstration of such universality (i.e., [Church-Turing computability](#)) and common limitations of that universality is one Miser Project objective: making this aspect of theoretical computing accessible in practical, demonstrable terms for the understanding of computing practitioners and other interested parties.

The $\langle ob \rangle$ model of computation, although very simple, is a sufficient foundation for computational-interpretation by stored-program operations that manifest layers of abstraction. By stored-program is meant taking obs themselves as scripts for operations on obs taken as operands. Such scripts can direct derivation of obs to be taken as further scripts, illustrating how computer programs (e.g., compilers) are used to generate (compile) further computer programs from descriptions in their data (e.g., programming-language texts), extending the reach of computer applications.

Narration of the $\langle ob \rangle$ model of computation is the subject of the next series of blog articles. The model is founded on two function representations: one by **obap.ap(p, x)** and one by **obap.eval(e)**. The operands, p , x , and e , are obs. Given definite obs for operands, any determined result is a definite ob. Dependable manifestation of the computational model in implemented operations of a digital computer rests on empirical determination that the computer's performance yields valid computation-model interpretations. All of that will be explored.

2. Structure, Interpretation, Representation, Computation, and Manifestation

Some informal notions are adapted and restricted in the context of the Miser Project. These specialized concepts assist in capturing the essence of computation and its practical application. This framing is not intended to be the only perspective. It is offered as one useful way to appreciate computation. Briefly, in rough progression.

- **Structure**

mathematical characterization of theoretical entities, in terms of domains, primitive notions, and applied logical theories

- **Interpretation**

- *mathematically*, arrangements by which one mathematical structure is modelled in another using a mathematically-demonstrated correspondence
- *linguistically*, interpretation of a text in a (standard) semantic domain, including computation-model interpretation of programming-language texts
- *empirically*, successfully-engineered achievement of mechanized operations confirmed to satisfy a model of computation; realized computational interpretation
- *socially/scientifically*, identifying objects in physical/social reality as evocative of theoretical entities, and sometimes *vice versa*; taking such determinations as pertaining to relationships in contingent reality

- **Representation**

- *mathematically*, the formulation of mathematical functions in a given structure's theory language, effectively determining the nature of the functions as theoretical entities; *notionally*, equivalent formulations of the same function can be taken as having the same functional interpretation
- *conceptually*, the entities and functions of one structure that serve as interpretations for those of another, thereby representing that other structure; *ideally*, the represented individuals are readily discerned upon inspection of (computationally) derived representations (i.e., definite representations are traceable back to the interpreted entities)
- *reflectively*, a representation may provide multiple entities that are equivalent as valid interpretations of a single entity in another structure (e.g., the propositional algebra simulation by ob.p01.bp in contrast with the subsumed ob.NE.bp interpretation of <bp> in [boole.txt](#))

- **Computation**

- *conceptually*, an orchestrated performance by which a representation of some entity x , the operand/data, is transformed into a representation of y , such that a specified functional relationship $y = f(x)$ is satisfied
- *informally*, the result of a computation, spoken of as the represented entity (e.g., "the square-root of x ", "the estimated fastest route from city A to city B today", or, interactively, "achievement of the end-game in *Tomb Raider*")

- **Manifestation**

- *generally*, the perceptual object of attention recognized in some other object

- or activity
- *operationally*, the act of producing/presenting objects or activities taken as so evocative
- *interactively*, with computer systems, the appearance evoked in the representation of computational results/operation on some medium (e.g., the perceived images on the surface of a computer-operated graphical display)
- *tacitly*, objects perceived as being manipulated via interaction with a computer system operating in keeping with manifestation fidelity

3. Review: More About Structure

Expositions of mathematical structures such as $\langle ob \rangle = \langle Ob, Of, Ot \rangle$ circumscribe particular theoretical entities in a mathematical-logic framework. The general pattern is $\langle s \rangle = \langle Sd, Sf, St \rangle$.

- *Sd* signify the *domain of discourse*, the collection of entities all of a single kind that are the subject of variables, such as x, y, z , in expressions of the structure (with universal quantification, such as with $\forall x \forall y \forall z$, usually implicit)
- *Sf* signify *all* of the functions and predicates with operands and function yields in *Sd*. Individual functions become known by their representation using *St*.
- *St* consist of the expressions that establish primitive notions, axioms, and function representations using a mathematical system of logic. The subject matter is always *Sd* and explicitly-represented elements of *Sf*. Certain functions and predicates may be taken as axiomatic, with all others expressible in terms of those.

For the Miser Project, *St* is typically an application of First-Order Logic, usually with $=$ (FOL= $=$). [This is the case of \$\langle ob \rangle\$.](#)

Distinguishing mathematical structures in this manner allows examination of multiple structures and connections among them. To be explicit about a structure being appealed to in a given discussion, prefixed names such as *ob.c*, *ob.NIL*, *bp.top*, *bvn.comp*, and *ob.bvx.cf* provide clarity in identifying the "host" structure and the entities being represented via such interpretation. [So far](#), in addition to $\langle ob \rangle$, the Boolean Algebra structure $\langle ba \rangle$ and its kin $\langle bp \rangle$, $\langle bn \rangle$, and $\langle bvn \rangle$ have been explored. The curious representation, *ob.bvx*, of *any* Boolean Algebra on a denumerable domain will invite further attention in the computational interpretation of number theory.

Structures evoke theoretical entities. There is no suggestion of material or natural existence. Rather, the *Sd* and *Sf* arise entirely in language *St* and live entirely in mathematics, so to speak. Whatever conceptual notions one might harbor about the subject of a structure, consideration and discussion of the properties of such entities is confined to what can be reconciled entirely via expression and deduction in applied-logic theory *St*, along with any [informal statements](#) that restrict/inform the constructions and valid deductions.

4. Interpretation: Variations on a Theme

Matters of interpretation arise in varied forms with respect to the Miser Project.

Mathematically, "interpretation" is used with regard to a correspondence between one structure, $\langle A \rangle$, and another, $\langle B \rangle$. Typically, there are four parts:

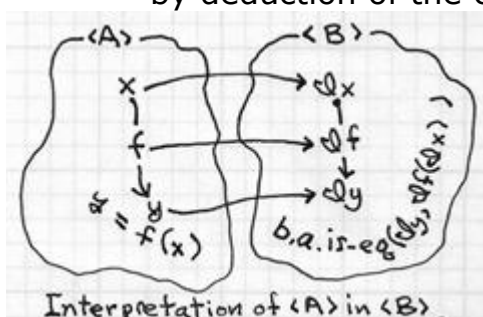
- Correspond each element of $\langle A \rangle$'s domain of discourse with a distinct member of $\langle B \rangle$'s domain. Given x in $\langle A \rangle$, signify its interpretation in $\langle B \rangle$ by Ix .
- Represent functions and predicates in $\langle B \rangle$ that are identified as corresponding interpretations of the primitives of $\langle A \rangle$.
- If the theory of $\langle A \rangle$ is an application of FOL=, interpret " $x = y$ " of $\langle A \rangle$ with $\langle B \rangle$ predicate $b.a.is-eq(Ix, Iy)$.

When

$$b.a.is-eq(Ix, Iy) \Leftrightarrow Ix = Iy$$

in Bt of $\langle B \rangle$, the interpretation is said to be an implementation of $\langle A \rangle$ in $\langle B \rangle$; otherwise, it is said to be a simulation of $\langle A \rangle$ in $\langle B \rangle$.

- The interpretation is *valid* when all deductions in structure $\langle A \rangle$ are mirrored by deduction of the corresponding interpretation in structure $\langle B \rangle$.



Valid interpretation of $\langle A \rangle$ in $\langle B \rangle$ need not capture all features and entities of $\langle B \rangle$. Validity achievement only signifies successful interpretation of $\langle A \rangle$.

An intended interpretation of a structure might be invalid. In that case, the offered interpretation may be defective or the structures $\langle A \rangle$ and/or $\langle B \rangle$ ill-suited to the intended purpose. Resolution of the matter is external to the structures themselves.

Texts or other concrete forms of expression satisfying a formal grammar can have specified interpretations as entities in a structure having computational interpretation. That form of interpretation [will be demonstrated](#) with the parsing of expressions that are interpreted in terms of some construction, such as $\langle ob \rangle$ computation-model operations.

Interpretations are also made between entities of mathematical structures and domains of non-mathematical character, such as observations in nature, predictions of scientific theories, and many practical and artistic situations. Acceptability is an empirical matter and depends on the particular circumstances and disciplines, such as (always-provisional) scientific confirmation at some scale of appearance.

Representation and manifestation are often intertwined with interpretation.

5. Concerning Representational Equivalence

"Representation" is used here with respect to conditions on the structure $\langle B \rangle$ in which interpretation of structure $\langle A \rangle$ is established. Representation is akin to establishment

of a portion of $\langle B \rangle$ that encompasses the interpretation of $\langle A \rangle$ but without restriction to uniqueness. Many entities, the representatives, of the representation of $\langle A \rangle$ in $\langle B \rangle$ may be interpreted as the same entity in $\langle A \rangle$, including the designated interpretation of that entity in $\langle B \rangle$. In this manner, equivalent representatives reflect back to a distinct entity and any of them can be regarded as if interpretations of that entity. It is in this sense that we speak of the interpretation of a representation. Representation in this manner is not arbitrary; validity of representation hinges on preservation of interpretation validity among all substitutions of equivalent representatives. Valid representations are entirely interpretation preserving.

Mathematical representation of functions has been illustrated with examples for $\langle ob \rangle$ and also for functions that serve as the interpretation-in- $\langle ob \rangle$ of functions in other structures, such as those prefixed $ob.p01.bp$ and $ob.bvx$. It is generally the case that there are many representations of a single function, based on the principle of extensional equivalence: when $f(x) = g(x)$ for all x , f and g as defined represent the same function. In these cases, the equivalence is rather sideways, signifying interpretation as the same entity amidst all the functions of the same structure.

In [boole.txt](#), the (section 2.3) $ob.NE.bp$ representation of $\langle bp \rangle$ in $\langle ob \rangle$ is strict with respect to interpretation of the $\langle bp \rangle$ domain entities \perp and \top , along with corresponding function representations of the $\langle bp \rangle$ primitive notions (axiomatic functions). In contrast, the (section 2.5) $ob.p01.bp$ representation preserves the $ob.NE.bp$ interpretation while having every ob represent one of $\langle bp \rangle$ \perp and \top . The $ob.p01.bp$ functions corresponding to $\langle bp \rangle$ primitive notions properly simulate the interpretation. The function $ob.p01.bp.comp(x)$ and agreement on canonical form are revealing in respect to interpretation $ob.NE.bp$ consistency and interpretation preservation of the $ob.p01.bp$ representation.

The [boole.txt](#) representations $ob.p01.bp$ and (section 3.4) $ob.bvx$ demonstrate that the same ob can serve in many interpretations and representations. It is not intrinsically-apparent which representation (whether $ob.p01.bp$, $ob.bvx$, or something else entirely) is intended. When particular canonical forms are observed, it is more suggestive yet not conclusive. (Nevertheless, gifted computer-software maintainers are excellent at inference of the purpose of program aspects and identifying consistent repairs for representation breakdowns. That's based on domain knowledge and the context in which the software is applied as well as cues in the code.)

It is valuable to express derived obs in a form that signifies what they are intended representatives for, what they represent other than merely obs as themselves, in specific situations. Such discrimination of representations and related cases involving tracking of interactive-operation states and ephemeral entities are beyond the $\langle ob \rangle$ model; they'll be investigated once the "pure" $\langle ob \rangle$ computational model has been explored. For now it is noteworthy that obs themselves do not reveal intended representations and that serves computational representation quite well as a computational-model foundation.

6. Choreography of Computation

*Computational processes are abstract beings that inhabit computers. ...
A computational process ... cannot be seen or touched. It is not*

composed of matter at all. However, it is very real.
 — *Building Abstractions with Computers, Chapter 1 in*
"Structure and Interpretation of Computer Programs," ed.2

Mathematical structures can be thought of as static: "standing still." There is a domain of discourse, such as Ob , and (functional) relationships; deductions can remain to be established. Just the same, nothing moves. It is as if it is all there in the first place (whether or not there's any actual "there" there).

"Computational interpretation" has been insinuated as the movement from the mathematical stability of a structure such as $\langle \text{ob} \rangle$ to some mechanical process that is carried out, different than the deductions and qualification as effectively-computable, yet consistent with that computability. Something moves, something processes.

Although computational procedures can be animated in a manner that allows the process's stages to be observed, that is not so easy in writing about it. And that's not so great in an elaborate computation with representations far removed from the nature of the (familiar) entities being simulated. We need an useful depiction that somehow suggests the essence of computational processes.



The depiction of a computational mechanism in operation will be illustrated with a diagrammatic pattern, whether animated or not. The rotating gear and flash of power is a pattern for depicting computational performance put into operation. This could be a human performance, although our focus is on digital computers and their computational procedures.

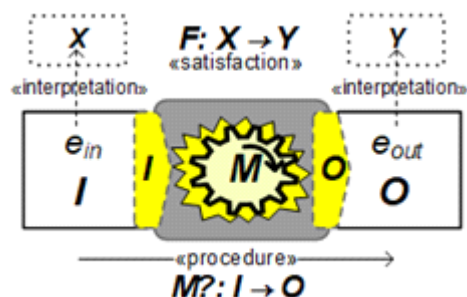


Consider that the performing machine operates on some (relatively) tangible form of input to be processed, yielding production of an output result that depends on nothing else than the input and the performance of the machine. Here the **I**, **M**, and **O** signify the particular input format, mechanism, and output format. The subscripted e_{in} and e_{out} refer to specific instances of an input encoding and an output encoding.



Input and output formats need to couple properly with the nature of the computing mechanism and its performance. Conformance of the input-output formats with provisions of the mechanism are signified by sockets that couple the input-output with the respective aspects of the mechanism symbol. In this simple input-processing-

output diagramming, it is a condition that the specification of the mechanical-operation process have the input and output characterized in the same terms — levels of abstraction — as the input and output formats are described.

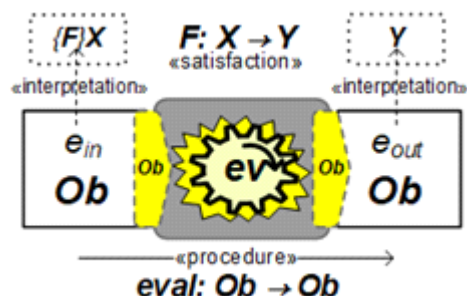


The idea of encoding is critical in grasping what the computation amounts to and also how we consider the use of the computation by representing (encoding) entities of interest.

At the level of the device, M , the $M?:I \rightarrow O$ symbolism is short-hand for the function by which every input, e_{in} has a determined output e_{out} in accordance with the assumed procedure's performance at that level. (This is itself likely to be an abstraction above the physical mechanism, whatever its form.)

In making use of the device, above, there is also a superior level of abstraction. Consider interpretation of some e_{in} as values, x , of some kind X , with interpretation of the derived e_{out} as values, y , of kind Y . With respect to those interpretations, the input and the derived output are taken as satisfying the functional relationship symbolized as $F:X \rightarrow Y$. The e_{in} qualifies as a representation of a chosen x . The diagrams feature interpretation over representation, and draw the arrows as interpretations from the representations because it is the interpretation that is considered the intention for selection of a (representative) encoding.

To operate computations on the pattern above, an user must select a machine that provides a suitable M and its $M?:I \rightarrow O$ operation, derive inputs that represent chosen members of X in order to achieve an output that is representative of a desired result in Y .



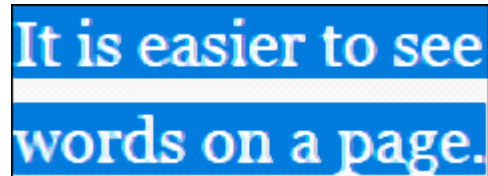
If M is *programmable* in some manner, there is more variety of choice for intended usage. Computational interpretation of **obap.eval(e)** allows representation of expressions that can be considered to express F and X together in satisfying $F:X \rightarrow Y$ with the computed output.

Variations on this pattern arise as Miser Project conceptualization progresses. The key

feature is distinguishing what a computation *is* versus what a computation *is for* (purposive use) and how that figures in representation validity and powerful layering of abstractions/representations. The gaps between what the computations are versus what they are used for will increase along with greater utility, usability and invisibility of the machinery.

7. Manifestation and Tacit Awareness

There is a point with computers where the experience of using the computer is as manifestation of something that is evoked by the exhibition of computational-processing activity.

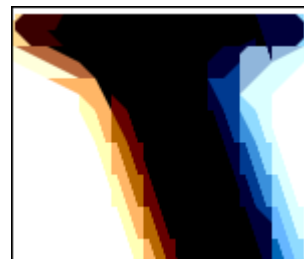


It is easier to see how we manifest something in looking at something rather static, such as words on a page. In reading, the automatic tendency is to discern a sense of the text without much examination word-by-word, unless this is not your language or there are other circumstances that bring attention to the individual words.



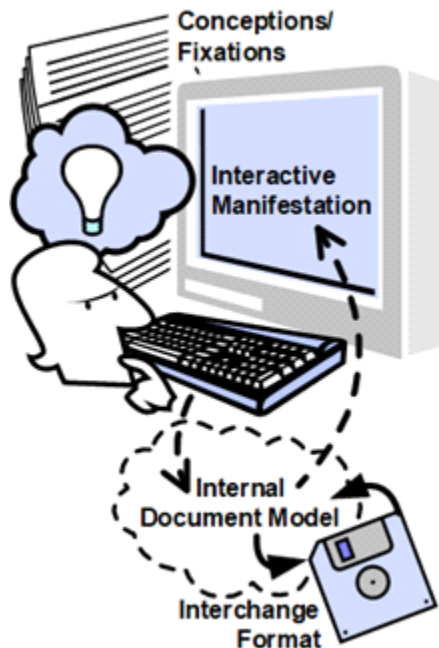
That is different than looking at individual words in isolation and then the individual letters and then the marks on which the letters are built.

Consider that when we focus on the sense that occurs to us, that is the object of focal attention/awareness. Subordinate attention/awareness is on observation of the substrate, the individual words and perhaps the sense of those in isolation.



When the sense is stripped away completely, at a level so subordinate that there is no suggested sense (as in the enlarged tip of an electronic rendition of “w”), we can discern how at the level of the electronic device’s operation, the manifestation of letters, words, and sentences – statements – is achieved. In this case, the manifestation has been engineered such that miniscule electronically-generated features are organized to present the text that is chosen to carry a communication.

When text is in your natural language and you are practiced at reading it, there is difficulty in not automatically grasping some sense of the text. Effort is required to strip away the focal and bring awareness to the subordinate; the tacit reaction is reading the text, even for a single word.



Tacit recognition is important in how manifestations are evoked with so much fidelity that the manifestation is perceived as being manipulated and preserved by computer interaction. The accomplishment is a wonder of engineering of subordinate processes to achieve such fidelity without interference (except when it doesn't).

In the example of electronic documents and modern word-processing software, the levels of representation and interpretation that are navigated are typical. The interactive manifestation is unrecognizable in the file format that is used for saving and interchanging a representation of the document. Computer software enacts some form of internal document model by which the interchange format and the manipulatable manifestation are coordinated. The process does not intrude on perception of the manifest document and the author/user perception of the conceptual document and how it might appear in print or some other fixed medium.

Even simplified word-processing models are beyond the immediate reach of the <ob> model of computation. There will be appeals and demonstrations of manifestation, starting at lower levels of computer operation before higher-level ones are considered.

One case of tacit understanding will arise early. It is easy, as a computer programmer, to recognize a formula that has some mathematical appearance as expression of a computation; this mistaken attribution is encouraged by the borrowing of mathematical notation for programming-language purposes. Since <ob> and the model of computation are unfamiliar in a programming context, that confusion is avoidable for a time. With familiarity using the Miser Project coding notation (oFrugal) for computation of obs, tacit understanding will likely kick in and we'll need to keep that harmless.

Onward.