F.*E*.*D*. Vignette #16 --

The Dialectic of 'Modern Computerware' --

Systematically Presented via a 4-Symbol Expression.

by Miguel Detonacciones

Author's Preface. The purpose of **F.*E*.*D***. Vignette **#16** is to provide another simple "worked example" of a ***dialectical equation-model*** and of the standard ***E*.*D***. method of solution for ***dialectical equation-models*** in general.

A Note about the On-Line Availability of Definitions of F.E.D. Key Technical Terms. Definitions of ***Encyclopedia Dialectica*** technical terms, including of ***E*.*D***. 'neologia', are available on-line via the following URLs --

http://www.dialectics.org/dialectics/Glossary.html

https://www.point-of-departure.org/Point-Of-Departure/ClarificationsArchive/ClarificationsArchive.htm

-- by clicking on the links associated with each such term, listed, in alphabetic order, on the web-pages linked-to above.

Links to definitions of the ***Encyclopedia Dialectica*** special terms most fundamental to this vignette are as follows --

«***aufheben***»
https://www.point-of-departure.org/Point-Of-Departure/ClarificationsArchive/Aufheben/Aufheben.htm

Diachronic vs. ***Synchronic***
http://point-of-departure.org/Point-Of-Departure/ClarificationsArchive/Synchronic/Synchronic.htm

Dyadic Seldon Function as '''***Self-Reflexive Function***'''
http://point-of-departure.org/Point-Of-Departure/ClarificationsArchive/SeldonFunctions/SeldonFunctions.htm

Dyadic Seldon Function as '***Self-Iteration***'
http://www.dialectics.org/dialectics/Glossary_files/%27Dyadic%20Seldon%20Functions%27_as_%27Self-Iterations%27,_vs._Standard_%27Other-Iterations%27.jpg

Historical or ***Diachronic Dialectics***
http://point-of-departure.org/Point-Of-Departure/ClarificationsArchive/HistoricalDialectics/HistoricalDialectics.htm

$_{\mathbf{N}}\mathbf{Q}$ ***dialectical arithmetic / algebra***
http://www.dialectics.org/dialectics/Correspondence_files/Letter17-06JUN2009.pdf

Systematic or ***Synchronic Dialectics***
http://point-of-departure.org/Point-Of-Departure/ClarificationsArchive/SystematicDialectics/SystematicDialectics.htm

-- and we plan to expand these definitions resources as the ***Encyclopedia Dialectica Dictionary Project*** unfolds.

[Note: '''Arithmetical Quantifiers''' vs. 'Arithmetical Qualifiers'. In the phrase "**3** apples", we term "**3**" the "arithmetical ["pure"-]***quant***ifier", and "apples" the '''***ontological***''' -- or ***kind*** of thing -- '''***qual***ifier'''. In the phrase "**3** pounds of apples", we term "pounds" the '***metrical***[-unit] ***qual***ifier' -- or '''unit of measure ***qual***ifier''' -- ***quant***ified by the **3**, which, together, '***quanto-qual**ify*' the '***ontological qual***ifier', "apples". A key use-value of the ***dialectical arithmetics*** is to provide algorithmic, ideographical-symbolic systems for the various kinds of 'arithmetical ***qual***ifiers', both with and without the co-presence of '''arithmetical ***quant***ifiers'''.].

Introduction. This model is a bit more challenging than the model of "TV-Series", presented earlier in this **F.*E*.*D***. Vignettes sub-series, but only because it requires some "domain-expertise" -- or, at least, some "domain familiarity" -- with respect to the domain of modern electronic digital computer systems -- both hardware systems and software systems.

We will, in this vignette, use the **F.*E*.*D***. 'first dialectical algebra' to construct, and to "solve", a "heuristic", 'intuitional' model of a systematic presentation of the domain of 'modern computerware' -- encompassing core aspects of both its *hardware* and its *software* sub-domains.

Herein we mean, by the word, '''systematic''' in the phrase '''systematic presentation''', a presentation of the major kinds of "entities" that exist in this domain -- by means of categories that classify those entities by their *"kinds"*, i.e., as '''*ontology*''', or as *"kinds* of things" -- in strict order of rising complexity, starting from the simplest category, and moving, step-by-step, from lesser to greater complexity, until we reach the most complex category for this domain, or for the purposes of this example.

The model we will build will generate descriptions of these categories in that strict, systematic order of rising complexity.

This will be a "snapshot" model, a "synchronic" model that takes the contemporary slice of time -- or at any rate, a recent-past slice of time -- and algorithmically generates descriptions of categories for entities that presently exist, or that might presently exist, for the model's domain, in their systematic order, as described above -- a '''***Systematic Dialectic***''' model.

Our model here will ***not*** be a "chronology" model, or "diachronic" model, like 'The Psychohistorical Equation of Human Social Formations Meta-Evolution', in which the units of *earlier* categories are described as actually, e.g., physically, ***constructing***, through their ***activity*** as *"**causal agents**"*, or *"**subjects**"*, the units of *later* categories, categories whose units ***did not exist*** until that construction took place.

That is, it will ***not*** be a model of a 'self-advancing' ***historical progression*** of ontology, with each historical epoch containing both old ontology, inherited from past historical epochs, and new ontology, ontology that had ***never appeared before*** -- in past historical epochs, as well as '''hybridizations''' of old and new -- ***until*** the *later* epoch in question. That is, the model constructed herein will ***not*** be an '''***Historical-Dialectical***''' model.

We will apply a documented, standard solution procedure to "solve" this model -- to determine what actual category each of these generated category-descriptions refers to, or to determine which, if any, of these category-descriptions describe "empty categories", i.e., 'combinatorial *possibles'* that *actually* do ***not*** exist for this domain -- at least not presently.

To get started, we must determine the starting-point -- the point-of-departure -- for our systematic model.

This starting category will be the seed of our whole progression of generated category-descriptions, influencing every category that follows, as it is their "controlling source", the "ever-present origin" of all that follows from it.

The heuristic rule for getting started is to ask oneself "**¿**What is the ***least complex*** kind of thing, the ***simplest*** kind of thing, which exists in this domain***?***" -- in our case, in the domain of 'modern computerware' -- and to then find the answer to that question, based upon one's prior knowledge of, or familiarity with, this domain.

The answer to this starting question that we will pursue in this example is the following: The "**bit**" is the simplest ancestor, the ultimate unit, of modern, electronic, digital 'computerware', ingredient in every one of the more complex units of that domain.

Single, circuit '**on**'-or-'**off**', '**1**' or '**0**', digital, Boolean, binary, base **2** arithmetic "**bits**" will be our base units.

Therefore, the category which we shall name **bits** is our starter category, and we shall symbolize it, in our ***specific*** category-algebra model, via the second letter of that name, as $\underline{\mathbf{i}}$, or as $\underline{\mathbf{q}}_{\mathbf{i}}$, identifying that ***specific*** category with the ***generic*** first category symbol of our ***generic*** category-arithmetic model, namely, with the symbol $\underline{\mathbf{q}}_{\mathbf{1}}$, in an identification, an "interpretation", or an "assignment" [all denoted by '**[---)**'] that we indicate by writing: $\underline{\mathbf{i}} \equiv \underline{\mathbf{q}}_{\mathbf{i}}$ **[---)** $\underline{\mathbf{q}}_{\mathbf{1}}$.

Our model then, will be an equation, that looks like this --

$$\underline{)\text{-}|\text{-}(}_{s} = \underline{i}^{2^{s}} = \underline{\text{bits}}^{2^{s}}$$

-- with the variable **s** indicating the **s**tep in our systematic presentation that the 'accumulation of categories', denoted by $\underline{)\text{-}|\text{-}(}_{s}$, represents.

Stage 0. Our initial **s**tep -- **s**tep $s = 0$ -- contains only our starting category, $\underline{i} \equiv \underline{q}_{i}$ --

$$\underline{)\text{-}|\text{-}(}_{0} = \underline{i}^{2^{0}} = \underline{i}^{1} = \underline{i}$$

because **2** "raised" to the power **0** -- 2^{0} -- is just **1**, and because $\underline{i}$ "raised" to the power **1** is just $\underline{i}$.

Stage 1. It is when we get to **s**tep $s = 1$ that our equation-model gives us something initially "unknown" -- and, therefore, something '''algebraical''', not just '''arithmetical''' -- to "solve-for" --

$$\underline{)\text{-}|\text{-}(}_{1} = \underline{i}^{2^{1}} = \underline{i}^{2} = \underline{i} \times \underline{i} = \underline{q}_{i} \times \underline{q}_{i} \equiv \underline{q}_{i} + \underline{q}_{ii} = \underline{i} + \underline{q}_{ii}$$

-- because **2** "raised" to the power **1** -- 2^{1} -- is just **2**, and because our rule of arithmetic for multiplying a generic category, call it $\underline{q}_{x} \equiv \underline{x}$, "by", or "into", itself, is, simply: $\underline{q}_{x} \times \underline{q}_{x} \equiv \underline{q}_{x} + \underline{q}_{xx} \equiv \underline{x} + \underline{q}_{xx}$.

Herein, the symbolic element $\underline{q}$ denotes the ***generic*** category '**q**ualifier'.

The subscript symbolic elements that come after it are used as ***specific*** category descriptors.

¿But how do we determine what the resulting, initially "unknown" category, or 'category-description', here $\underline{q}_{ii}$, ***means?***

Well, the ***generic*** rule to "solve-for" the categorial ***meaning*** of such symbols is that, if we know what is meant by category $\underline{q}_{x} = \underline{x}$, then the symbol $\underline{q}_{xx}$ often describes a category each of whose units is an '**x** ***OF*** **x**s', that is, a category for a different kind of units, called '***meta*-x**s', each such 'meta-unit' being made up out of a multiplicity of **x**s.

To be ***specific*** with this rule, $\underline{q}_{ii}$ specifies a category each of whose units is a '**bit** ***OF*** **bit**s', that is, a '***meta*-bit**', such that each '***meta*-bit**' is made up out of a multiplicity of **bit**s. Our experience of 'modern computerware' suggests that such units do indeed presently exist.

That 'category-description' describes the category of 'multi-bit' units -- of **bytes**, i.e., of computer ***characters***, such as ASCII characters -- which we therefore symbolize, again, via the second letter of its name, by $\underline{y}$.

We may "assert" our solution as follows: $\underline{q}_{ii} = \underline{q}_{y} \equiv \underline{y}\ [\text{---})\ \underline{q}_{2}$.

Again, what is ***dialectical*** about the relationship between $\underline{i}$ and $\underline{i}^{2}$, or $\underline{i} \times \underline{i}$, or $\underline{ii}$, or $\underline{i}$ ***of*** $\underline{i}$, or $\underline{i}(\underline{i})$, the relationship of what we call '***meta-unit-ization***', or '***meta*-«*monad*»-*ization***', between $\underline{i}$ and its already presently existing, '**supplementary** other', $\underline{y}$, is that this relationship is a synchronic «***aufheben***» relationship: each single unit of the **bytes** category is a ***negation***, and also a ***preservation***, by way of also being an ***elevation to*** *the* / ***forming*** *the* **bytes** category / level / scale, of a ***whole*** [*sub-*]***group*** of ***units*** of the **bits** category / level / scale.

So, our full solution to the step $\mathbf{s} = 1$ equation of our model is --

$)\text{-}|\text{-}(_1 \; = \; \underline{i} + \underline{y} \; = \;$ **bits** + **bytes**.

If this model is working right, **bits** will be the ***simplest*** category of the domain of 'modern computerware', and **bytes** will be the ***next more complex*** category of that domain.

Stage 2. ¿What additional 'category-specifications' do we generate in our next step, step $\mathbf{s} = 2$, that need "solving-for"?

Let's find out:

$$)\text{-}|\text{-}(_2 \; = \; \underline{i}^{2^2} \; = \; \underline{i}^4 \; = (\underline{i}^2)^2 \; = (\underline{i} + \underline{y})^2 \; = \; (\underline{i} + \underline{y}) \times (\underline{i} + \underline{y}) \; = \; \underline{i} + \underline{y} + \underline{q}_{yi} + \underline{q}_{yy}.$$

This result arises by way of two key rules of categorial algebra, plus the ***general*** rule for multiplication when one category is multiplied by a different category [we used a ***special*** case of this ***general*** rule, for the case where the same category is multiplied by itself, in step $\mathbf{s} = 1$, above] --

1. $\underline{q}_b \times \underline{q}_a \; = \; \underline{q}_a + \underline{q}_{ba} \; = \; \underline{a} + \underline{q}_{ba}$; ***special*** case: $\underline{q}_b \times \underline{q}_b \; = \; \underline{q}_b + \underline{q}_{bb} \; = \; \underline{b} + \underline{q}_{bb}$.

2. $\underline{q}_a + \underline{q}_a \; = \; \underline{q}_a$; the same category, added to itself, does not make "two" of that category; one "copy" of each category is sufficient; two or more "copies" of any category would be redundant for the purposes of categorial algebra.

3. There is no $\underline{q}_x$ such that $\underline{q}_a + \underline{q}_b \; = \; \underline{q}_x$; *different* categories, added together [as opposed to being '''multiplied'''], ***do not reduce*** to a single category, as in the case of the proverbial '**apples** + **oranges**', or **a** + **o**.

Well, we already know how to "solve-for" $\underline{q}_{yy}$.

It describes a category of '**bytes *OF* bytes**' -- a category each of whose units is a '**byte *OF* bytes**', i.e., each of which is a '***meta*-byte**', such that each such '***meta*-byte**' is made up out of a typically heterogeneous multiplicity of **byte**s, e.g., a "character string" typically consisting of a sequence of different -- or mostly different -- ASCII characters.

Our experience of 'modern computerware' suggests that such units do indeed presently exist.

That category-description describes the category of 'multi-byte' units -- of computer "**words**", e.g., the components of a computer-program's commands -- which we therefore symbolize, this time via the first letter of its name, by $\underline{\mathbf{w}}$.

We may "assert" our solution as follows: $\underline{q}_{yy} \; = \; \underline{q}_w \; \equiv \; \underline{w}$ **[---)** $\underline{q}_4$.

Our step $\mathbf{s} = 2$ equation-model, as we have solved it so far, thus now looks like this --

$)\text{-}|\text{-}(_2 \; = \; \underline{i}^{2^2} \; = \; \underline{i}^4 \; = \; \underline{i} + \underline{y} + \underline{q}_{yi} + \underline{w}$ [Note emerging pattern: $\underline{i}^2$ generates **2** categories, $\underline{i}^4$, **4** categories]

-- since we still have not yet determined which actual category of the 'modern computerware' domain is described by the algorithmically-generated symbol $\underline{q}_{yi}$ -- if any, i.e., if $\underline{q}_{yi}$ is not an "inoperative term" for this domain, i.e., representing the "empty category" for this domain.

When, as a component of $(\underline{\mathbf{i}} + \underline{\mathbf{y}}) \times (\underline{\mathbf{i}} + \underline{\mathbf{y}})$, the "higher-complexity" category, $\underline{\mathbf{y}}$, operates upon / "multiplies" the "lower-complexity" category, $\underline{\mathbf{i}}$ --

$$\underline{\mathbf{y}} \times \underline{\mathbf{i}} = \underline{\mathbf{i}} + \underline{\mathbf{q}}_{\mathbf{yi}}$$

-- ***generically*** speaking, the categorial relationship to be called to the user's attention by this operation, in this 'categorial algebra', is, again, a synchronic «***aufheben***» relationship, this time between $\underline{\mathbf{i}}$ and $\underline{\mathbf{q}}_{\mathbf{yi}}$. It calls the user to search that user's knowledge and memory of the domain in question -- in this ***specific*** case, the domain of 'modern computerware' -- for a category which represents an "uplift" of category $\underline{\mathbf{i}}$ entities to the level of the entities native to category $\underline{\mathbf{y}}$, thereby "canceling" the $\underline{\mathbf{i}}$-type entities concerned, at their own native level, but, by the same token, "preserving" those "special" category $\underline{\mathbf{i}}$ entities that **q**ualify for this "hybrid" category, combining $\underline{\mathbf{y}}$ and $\underline{\mathbf{i}}$ **q**ualities, in the relationship of "elevation" of those category $\underline{\mathbf{i}}$ entities up to within the level typical of category $\underline{\mathbf{y}}$ entities. Thus, the additional category thereby presented, $\underline{\mathbf{q}}_{\mathbf{yi}}$, signifies exceptional $\underline{\mathbf{i}}$ units, that "double as" $\underline{\mathbf{y}}$ units, or that "masquerade as" $\underline{\mathbf{y}}$ units, or that "exist in the way that, normally, only $\underline{\mathbf{y}}$ units exist".

For example, if we were building a systematic-dialectical model of written English, with $\underline{\mathbf{L}}$ denoting the category of **Letters** of the English alphabet, and with $\underline{\mathbf{W}}$ denoting the category of written English **Words**, then the category-symbol $\underline{\mathbf{q}}_{\mathbf{WL}}$ would stand for the category of individual English **L**etters that also **q**ualify as English **W**ords, e.g., 'a' and 'I'.

In this ***specific*** case, this means that a unit of the $\underline{\mathbf{q}}_{\mathbf{yi}}$ category is a **bit** unit that passes for a **byte** unit.

¿Do any such units presently exist, in the domain of 'modern computerware'***?***

¿E.g., are there any computer programming languages in which a single **bit** unit can mimic a whole **byte** unit***?***

Yes -- **F**.***E***.***D***. research has identified, we believe, just such entities. In a later version of this text, we will document their existence.

Were we to find no instances of such units in present existence, then "category" $\underline{\mathbf{q}}_{\mathbf{yi}}$ *might* be an instance of the generic "***empty category***", denoted ● -- connoting "full zero", or "existential zero" -- for this particular domain, and we *might*, in that case, "assert" our solution as follows: $\underline{\mathbf{q}}_{\mathbf{3}}\ \mathbf{(---]}\ \underline{\mathbf{q}}_{\mathbf{yi}} = ●\ \mathbf{[---)}\ \underline{\mathbf{q}}_{\mathbf{0}}$.

We *might* therefore write our *full* solution for step $\mathbf{s} = \mathbf{2}$ as --

$$\underline{\mathbf{)\text{-}|\text{-}(}}_{2} = \underline{\mathbf{i}}^{2^{2}} = \underline{\mathbf{i}}^{4} = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{\mathbf{yi}} + \underline{\mathbf{w}} = \underline{\mathbf{i}} + \underline{\mathbf{y}} + ● + \underline{\mathbf{w}} = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{w}} =$$

bits + **bytes** + **words**.

But let's keep our 'categorial-combinatorially ***possible***' category $\underline{\mathbf{q}}_{\mathbf{yi}}$ around for a while longer, since I believe that we have turned up concrete instances of just the kinds of entities that this category-description describes in our research --

$$\underline{\mathbf{)\text{-}|\text{-}(}}_{2} = \underline{\mathbf{i}}^{2^{2}} = \underline{\mathbf{i}}^{4} = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{\mathbf{yi}} + \underline{\mathbf{w}}.$$

<u>Stage 3</u>. ¿What additional ‘category-specs.’ do we generate in our next step, **<u>s</u>**tep $\mathbf{s} = \mathbf{3}$, that need “solving-for”***?***

Let’s see:

$$\mathbf{)\!-\!|\!-\!(}_3 = \underline{\mathbf{i}}^{2^3} = \underline{\mathbf{i}}^8 = (\underline{\mathbf{i}}^4)^2 = (\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}}) \times (\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}}) =$$

$$\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{q}}_{ww}.$$

We already know -- from past experience, narrated above -- how to “solve-for” $\underline{\mathbf{q}}_{ww}$.

It describes a category of ‘**<u>w</u>ord**s ***<u>OF</u>*** **<u>w</u>ord**s’ -- a category each of whose units is a ‘**<u>w</u>ord** ***<u>OF</u>*** **<u>w</u>ord**s’, i.e., each of which is a ‘***meta*-word**’, such that each such ‘***<u>meta</u>*-word**’ is made up out of a typically heterogeneous multiplicity of at least two [different] **word**s. Our experience of ‘modern computerware’ suggests that such units do indeed presently exist.

That ‘category-description’ describes the category of ‘multi-**<u>w</u>**ord’ units -- of computer “**<u>commands</u>**” -- the components of a computer-program, e.g., for a minimal, “unary”, operator **<u>c</u>ommand**, with at least one **<u>w</u>ord** representing the “operator” or “function” of that **<u>c</u>ommand**, and operating upon at least one *other* **<u>w</u>ord** representing the “operand” or “argument” of that **<u>c</u>ommand** -- which we therefore symbolize, via the first letter of its name, by $\underline{\mathbf{c}}$.

We may “assert” our solution as follows: $\underline{\mathbf{q}}_{ww} = \underline{\mathbf{q}}_{c} \equiv \underline{\mathbf{c}}\ [\text{---})\ \underline{\mathbf{q}}_8$.

Our **<u>s</u>**tep $\mathbf{s} = \mathbf{3}$ equation-model, as we have solved it so far, thus now looks like this --

$$\mathbf{)\!-\!|\!-\!(}_3 = \underline{\mathbf{i}}^{2^3} = \underline{\mathbf{i}}^8 = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{c}}$$ [<u>Note</u>: $\underline{\mathbf{i}}^8$ generates **8** categories]

-- since we have not yet determined which actual categories of the ‘modern computerware’ domain are described by the algorithmically-generated ‘category-description’ symbols $\underline{\mathbf{q}}_{wi}$, $\underline{\mathbf{q}}_{wy}$, and $\underline{\mathbf{q}}_{wyi}$, ***if any***.

But we already know how to characterize the ***possible*** categories that these three category-symbols “call for”, viz.:

- $\underline{\mathbf{q}}_{wi}\ [\text{---})\ \underline{\mathbf{q}}_5$ “calls for” the category of a kind of bit units that stand in for computer word units.
- $\underline{\mathbf{q}}_{wy}\ [\text{---})\ \underline{\mathbf{q}}_6$ “calls for” the category of a kind of byte units that stand in for computer word units.
- $\underline{\mathbf{q}}_{wyi}\ [\text{---})\ \underline{\mathbf{q}}_7$ “calls for” the category of a kind of $\underline{\mathbf{q}}_{yi}$ units that that stand in for computer word units.

¿Do any such units presently exist, in the domain of ‘modern computerware’***?***

- ¿Are there any computer programming languages in which a single bit unit can function as a whole word***?***
- ¿Are there any computer programming languages in which a single byte unit can function as a whole word***?***
- ¿Are there any computer programming languages in which a single $\underline{\mathbf{q}}_{yi}$ unit can function as a whole word***?***

Were we to find no instances of such units in present existence, then the “categories” $\underline{\mathbf{q}}_{wi}$, $\underline{\mathbf{q}}_{wy}$, and $\underline{\mathbf{q}}_{wyi}$ *might* all be instances of the generic “***empty category***”, ●, and we *might* “assert” our solution as follows:

$$\underline{\mathbf{q}}_{wi} = \underline{\mathbf{q}}_{wy} = \underline{\mathbf{q}}_{wyi} = \bullet\ [\text{---})\ \underline{\mathbf{q}}_0.$$

We *might* therefore write our *full* solution for step $\mathbf{s} = \mathbf{3}$ as --

$$\mathbf{)\!-\!|\!-\!(}_3 = \underline{\mathbf{i}}^{2^3} = \underline{\mathbf{i}}^8 = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{c}} =$$

$$\underline{\mathbf{i}} + \underline{\mathbf{y}} + \bullet + \underline{\mathbf{w}} + \bullet + \bullet + \bullet + \underline{\mathbf{c}} = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{w}} + \underline{\mathbf{c}} =$$

<u>bits</u> + **<u>bytes</u>** + **<u>words</u>** + **<u>commands</u>**.

But let’s keep all of our ‘categorial-combinatorially ***possible***’ categories around for a while longer --

$$\underline{)\text{-}|\text{-}(}_{3} = \underline{i}^{2^{3}} = \underline{i}^{8} = \underline{i} + \underline{y} + \underline{q}_{yi} + \underline{w} + \underline{q}_{wi} + \underline{q}_{wy} + \underline{q}_{wyi} + \underline{c}.$$

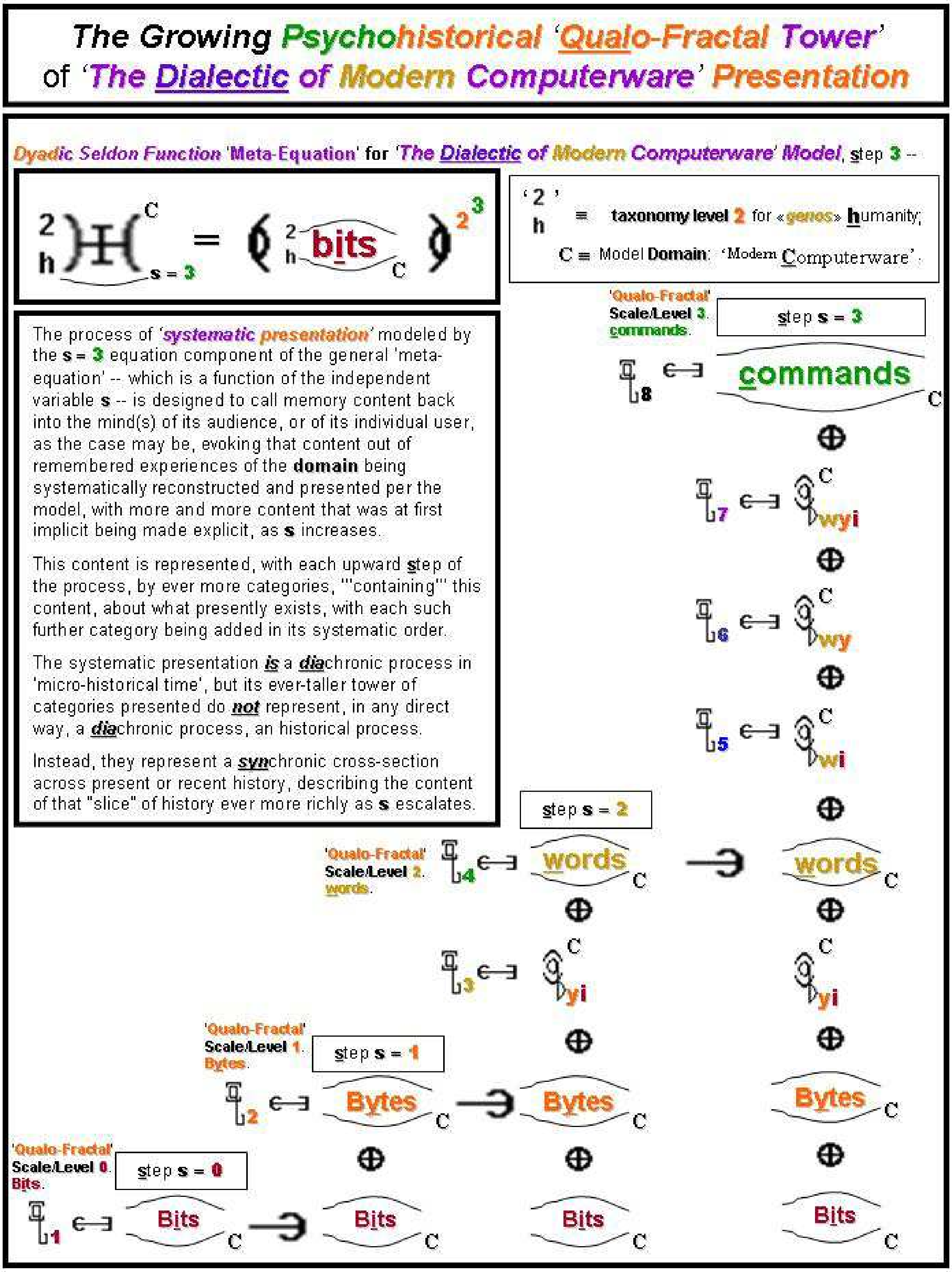
Our categorial progression so far can be summarized textually as below, and pictorially as on the following page.

The ‘qualo-fractal content-structure’ of this ‘psychohistorical dialectic’ to **s**tep **3** can be summarized as follows --

Computer Commands “contain” **Computer Words**, “containing” **Computer Bytes**, “containing” **Computer Bits**.

The “four symbolic-elements expression” for this model to this **s**tep is thus $\underline{i}^{2^{3}}$ [four if we count the underscore under the $\underline{i}$ as a separate “symbolic-element”].

The meaning mnemonically compressed into the **4** symbolic-element expression $\underline{i}^{2^3}$ can be depicted as follows --



The Growing *Psychohistorical* '*Qualo-Fractal Tower*' of '*The Dialectic of Modern Computerware*' *Presentation*

Dyadic Seldon Function 'Meta-Equation' for '*The Dialectic of Modern Computerware*' *Model*, step **3** --

$$ {}^{2}_{h})\!\!H\!\!(^{C}_{s=3} = \left({}^{2}_{h}\text{bits}_{C} \right)^{2^{3}} $$

'2' h ≡ taxonomy level **2** for «*genos*» **h**umanity; C ≡ Model **Domain**: 'Modern Computerware'.

The process of '*systematic presentation*' modeled by the **s** = **3** equation component of the general 'meta-equation' -- which is a function of the independent variable **s** -- is designed to call memory content back into the mind(s) of its audience, or of its individual user, as the case may be, evoking that content out of remembered experiences of the **domain** being systematically reconstructed and presented per the model, with more and more content that was at first implicit being made explicit, as **s** increases.

This content is represented, with each upward step of the process, by ever more categories, '"containing"' this content, about what presently exists, with each such further category being added in its systematic order.

The systematic presentation ***is*** a ***dia***chronic process in 'micro-historical time', but its ever-taller tower of categories presented do ***not*** represent, in any direct way, a ***dia***chronic process, an historical process.

Instead, they represent a ***syn***chronic cross-section across present or recent history, describing the content of that "slice" of history ever more richly as **s** escalates.

'Qualo-Fractal' Scale/Level 3. commands.
step s = 3
commands
wyi
wy
wi
step s = 2
'Qualo-Fractal' Scale/Level 2. words.
words
yi
'Qualo-Fractal' Scale/Level 1. Bytes.
step s = 1
Bytes
'Qualo-Fractal' Scale/Level 0. Bits.
step s = 0
Bits

Stage 4. ¿What additional ‘category-descriptions’ do we generate in our next step, **s**tep $\mathbf{s = 4}$, that need “solving-for”***?***

We [re-]generate the following [old and] new categories / ‘category-descriptions’:

$$)\text{-}|\text{-}(_{4} = \underline{\mathbf{i}}^{2^{4}} = \underline{\mathbf{i}}^{16} = (\underline{\mathbf{i}}^{8})^{2} = (\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{c}})^{2} =$$

$$\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{c}} + \underline{\mathbf{q}}_{ci} + \underline{\mathbf{q}}_{cy} + \underline{\mathbf{q}}_{cyi} + \underline{\mathbf{q}}_{cw} + \underline{\mathbf{q}}_{cwi} + \underline{\mathbf{q}}_{cwy} + \underline{\mathbf{q}}_{cwyi} + \underline{\mathbf{q}}_{cc}.$$

We already know how to “solve-for” $\underline{\mathbf{q}}_{cc}$.

It describes a category for ‘**c**ommand**s** ***OF*** **c**ommand**s**’ -- a category each of whose units is a ‘**c**ommand ***OF*** **c**ommand**s**’, i.e., each of which is a ‘***meta*-command**’, such that each typical such ‘***meta*-command**’ is made up out of a multiplicity of **command**s. Our experience of ‘modern computerware’ suggests that such units do indeed presently exist.

That category-description describes the category of ‘multi-**c**ommand’ units -- of computer “**programs**” -- which we therefore symbolize, via the first letter of its name, by $\underline{\mathbf{p}}$.

We may “assert” our solution as follows: $\underline{\mathbf{q}}_{cc} = \underline{\mathbf{q}}_{p} \equiv \underline{\mathbf{p}}$ **[---)** $\underline{\mathbf{q}}_{16}$.

Our **s**tep $\mathbf{s = 4}$ equation-model, as we have solved it so far, thus now looks like this --

$$)\text{-}|\text{-}(_{4} = \underline{\mathbf{i}}^{2^{4}} = \underline{\mathbf{i}}^{16} =$$

$$\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{c}} + \underline{\mathbf{q}}_{ci} + \underline{\mathbf{q}}_{cy} + \underline{\mathbf{q}}_{cyi} + \underline{\mathbf{q}}_{cw} + \underline{\mathbf{q}}_{cwi} + \underline{\mathbf{q}}_{cwy} + \underline{\mathbf{q}}_{cwyi} + \underline{\mathbf{p}}$$

-- since we have not determined which ***actual*** categories of the ‘modern computerware’ domain, ***if any***, answer to the ‘combinatorially’ ***possible***, mechanically-generated symbols $\underline{\mathbf{q}}_{ci}$, $\underline{\mathbf{q}}_{cy}$, $\underline{\mathbf{q}}_{cyi}$, $\underline{\mathbf{q}}_{cw}$, $\underline{\mathbf{q}}_{cwi}$, $\underline{\mathbf{q}}_{cwy}$, and $\underline{\mathbf{q}}_{cwyi}$. But we do know how to characterize the ***possible*** categories that these seven category-symbols “call for”, viz.:

- $\underline{\mathbf{q}}_{ci}$ **[---)** $\underline{\mathbf{q}}_{9}$ “calls for” the category of a kind of single bit units that function as command units.
- $\underline{\mathbf{q}}_{cy}$ **[---)** $\underline{\mathbf{q}}_{10}$ “calls for” the category of a kind of single byte units that function as command units.
- $\underline{\mathbf{q}}_{cyi}$ **[---)** $\underline{\mathbf{q}}_{11}$ “calls for” the category of a kind of $\underline{\mathbf{q}}_{yi}$ units that function as command units.
- $\underline{\mathbf{q}}_{cw}$ **[---)** $\underline{\mathbf{q}}_{12}$ “calls for” the category of a kind of single word units that function as command units.
- $\underline{\mathbf{q}}_{cwi}$ **[---)** $\underline{\mathbf{q}}_{13}$ “calls for” the category of a kind of $\underline{\mathbf{q}}_{wi}$ units that function as command units.
- $\underline{\mathbf{q}}_{cwy}$ **[---)** $\underline{\mathbf{q}}_{14}$ “calls for” the category of a kind of $\underline{\mathbf{q}}_{wy}$ units that function as command units.
- $\underline{\mathbf{q}}_{cwyi}$ **[---)** $\underline{\mathbf{q}}_{15}$ “calls for” the category of a kind of $\underline{\mathbf{q}}_{wyi}$ units that function as command units.

¿Do any such units exist, in the domain of ‘modern computerware’***?***

- ¿Are there any computer programming languages in which a single bit unit functions as a command unit***?***
- ¿Are there any computer programming languages in which a single byte unit functions as a command unit***?***
- ¿Are there any computer programming languages in which a single $\underline{\mathbf{q}}_{yi}$ unit functions as a command unit***?***
- ¿Are there any computer programming languages in which a single word unit functions as a command unit***?***
- ¿Are there any computer programming languages in which a single $\underline{\mathbf{q}}_{wi}$ unit functions as a command unit***?***
- ¿Are there any computer programming languages in which a single $\underline{\mathbf{q}}_{wy}$ unit functions as a command unit***?***
- ¿Are there any computer programming languages in which a single $\underline{\mathbf{q}}_{wyi}$ unit functions as a command unit***?***

Were we to find no instances of such units presently in existence, then the “categories” $\underline{\mathbf{q}}_{ci}$, $\underline{\mathbf{q}}_{cy}$, $\underline{\mathbf{q}}_{cyi}$, $\underline{\mathbf{q}}_{cw}$, $\underline{\mathbf{q}}_{cwi}$, $\underline{\mathbf{q}}_{cwy}$, and $\underline{\mathbf{q}}_{cwyi}$, *might* all be instances of the generic “***empty category***”, ●, and we *might* “assert” our solution as follows:

$$\underline{\mathbf{q}}_{ci} = \underline{\mathbf{q}}_{cy} = \underline{\mathbf{q}}_{cyi} = \underline{\mathbf{q}}_{cw} = \underline{\mathbf{q}}_{cwi} = \underline{\mathbf{q}}_{cwy} = \underline{\mathbf{q}}_{cwyi} = \bullet \ \mathbf{[---)}\ \underline{\mathbf{q}}_{0}.$$

We *might* therefore write our *full* solution for step $\mathbf{s} = \mathbf{4}$ as --

$$)\text{-}|\text{-}(_{4} = \underline{\mathbf{i}}^{2^{4}} = \underline{\mathbf{i}}^{16} =$$

$$\underline{\mathbf{i}} + \underline{\mathbf{y}} + ● + \underline{\mathbf{w}} + ● + ● + ● + \underline{\mathbf{c}} + ● + ● + ● + ● + ● + ● + ● + \underline{\mathbf{p}} = \underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{w}} + \underline{\mathbf{c}} + \underline{\mathbf{p}} =$$

bits + **bytes** + **words** + **commands** + **programs**.

But let's keep all of our 'categorial-combinatorially ***possible***' categories around for a while longer --

$$)\text{-}|\text{-}(_{4} = \underline{\mathbf{i}}^{2^{4}} = \underline{\mathbf{i}}^{16} =$$

$$\underline{\mathbf{i}} + \underline{\mathbf{y}} + \underline{\mathbf{q}}_{yi} + \underline{\mathbf{w}} + \underline{\mathbf{q}}_{wi} + \underline{\mathbf{q}}_{wy} + \underline{\mathbf{q}}_{wyi} + \underline{\mathbf{c}} + \underline{\mathbf{q}}_{ci} + \underline{\mathbf{q}}_{cy} + \underline{\mathbf{q}}_{cyi} + \underline{\mathbf{q}}_{cw} + \underline{\mathbf{q}}_{cwi} + \underline{\mathbf{q}}_{cwy} + \underline{\mathbf{q}}_{cwyi} + \underline{\mathbf{p}}.$$

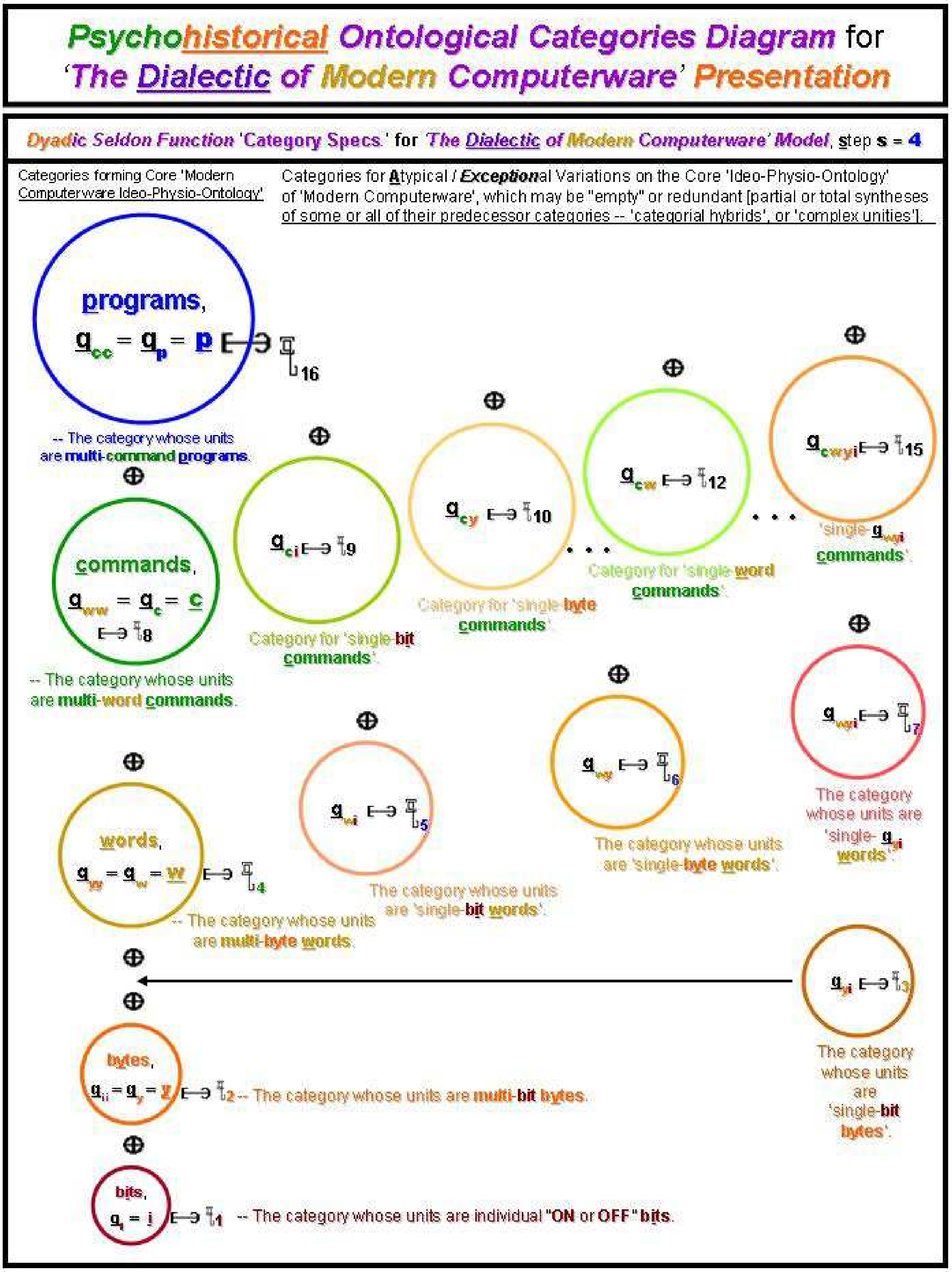
Our categorial progression so far can be summarized textually as below, and pictorially as on the following page.

The 'qualo-fractal content-structure' of this 'psychohistorical dialectic' to **s**tep **4** can be summarized as follows --

Computer Programs "contain" **Computer Commands**, which "contain" **Computer Words**, which "contain" **Computer Bytes**, which "contain" **Computer Bits**.

The "four symbolic-elements expression" for this model to this **s**tep is thus $\underline{\mathbf{i}}^{2^{4}}$ [four if we count the underscore under the $\underline{\mathbf{i}}$ as a separate "symbolic-element"].

The meaning mnemonically compressed into the **4** symbolic-element expression $\underline{i}^{2^4}$ can be depicted as follows --


Psychohistorical Ontological Categories Diagram for
'The Dialectic of Modern Computerware' Presentation
Dyadic Seldon Function 'Category Specs.' for 'The Dialectic of Modern Computerware' Model, step s = 4
Categories forming Core 'Modern Computerware Ideo-Physio-Ontology'
Categories for Atypical / Exceptional Variations on the Core 'Ideo-Physio-Ontology' of 'Modern Computerware', which may be "empty" or redundant [partial or total syntheses of some or all of their predecessor categories -- 'categorial hybrids', or 'complex unities'].
programs,
$q_{cc} = q_p = p \mapsto ⥣_{16}$
-- The category whose units are multi-command programs.
commands,
$q_{ww} = q_c = c \mapsto ⥣_8$
-- The category whose units are multi-word commands.
$q_{ci} \mapsto ⥣_9$
Category for 'single-bit commands'.
$q_{cy} \mapsto ⥣_{10}$
Category for 'single-byte commands'.
$q_{cw} \mapsto ⥣_{12}$
Category for 'single-word commands'.
$q_{cwyi} \mapsto ⥣_{15}$
'single-q_{wyi} commands'.
words,
$q_{yy} = q_w = w \mapsto ⥣_4$
-- The category whose units are multi-byte words.
$q_{wi} \mapsto ⥣_5$
The category whose units are 'single-bit words'.
$q_{wy} \mapsto ⥣_6$
The category whose units are 'single-byte words'.
$q_{wyi} \mapsto ⥣_7$
The category whose units are 'single-q_{yi} words'.
$q_{yi} \mapsto ⥣_3$
The category whose units are 'single-bit bytes'.
bytes,
$q_{ii} = q_y = y \mapsto ⥣_2$
-- The category whose units are multi-bit bytes.
bits,
$q_i = i \mapsto ⥣_1$
-- The category whose units are individual "ON or OFF" bits.

Stage 5. ¿What additional ‘category-descriptions’ do we generate in our next step, **s**tep $\mathbf{s} = \mathbf{5}$, that need “solving-for” in terms of existing kinds of ‘modern computerware’ units, and their categories**?**

With **5** as our **s**tep value, our Seldon Function calls our attention, again, to the following **16** recurring categories, plus to the following additional **16** ‘category-descriptions’ arising in this **s**tep for the first time, thus yet to be “solved-for” --

$$\underline{)\text{-}|\text{-}(}_5 \;=\; \underline{i}^{2^5} \;=\; \underline{i}^{32} \;=\; (\,\underline{i}^{16})^2 \;=$$

$$(\underline{i} + \underline{y} + \underline{q}_{yi} + \underline{w} + \underline{q}_{wi} + \underline{q}_{wy} + \underline{q}_{wyi} + \underline{c} + \underline{q}_{ci} + \underline{q}_{cy} + \underline{q}_{cyi} + \underline{q}_{cw} + \underline{q}_{cwi} + \underline{q}_{cwy} + \underline{q}_{cwyi} + \underline{p})^2 \;=$$

$$\underline{i} + \underline{y} + \underline{q}_{yi} + \underline{w} + \underline{q}_{wi} + \underline{q}_{wy} + \underline{q}_{wyi} + \underline{c} + \underline{q}_{ci} + \underline{q}_{cy} + \underline{q}_{cyi} + \underline{q}_{cw} + \underline{q}_{cwi} + \underline{q}_{cwy} + \underline{q}_{cwyi} + \underline{p} +$$

$$\underline{q}_{pi} + \underline{q}_{py} + \underline{q}_{pyi} + \underline{q}_{pw} + \underline{q}_{pwi} + \underline{q}_{pwy} + \underline{q}_{pwyi} + \underline{q}_{pc} + \underline{q}_{pci} + \underline{q}_{pcy} + \underline{q}_{pcyi} + \underline{q}_{pcw} + \underline{q}_{pcwi} +$$

$$\underline{q}_{pcwy} + \underline{q}_{pcwyi} + \underline{q}_{pp}.$$

We do already know how to “solve-for” $\underline{q}_{pp}$.

It describes a category for ‘**programs** ***OF*** **programs**’ units -- a category each of whose units is a ‘**program** ***OF*** **programs**’, i.e., each of which is a ‘***meta*****-program**’, such that each typical such ‘***meta*****-program**’ is made up out of a heterogeneous multiplicity of **programs**. Our experience of ‘modern computerware’ suggests that such units do indeed presently exist.

That category-description describes the category of ‘multi-**p**rogram’ units -- of computer “**software systems**” -- which we therefore symbolize, via the first letter of that name, by $\underline{s}$.

We may “assert” our solution as follows: $\underline{q}_{pp} \;=\; \underline{q}_{s} \;\equiv\; \underline{s}\ [\text{---})\ \underline{q}_{32}$.

Our **s**tep $\mathbf{s} = \mathbf{5}$ equation-model, as we have solved it so far, thus now looks like this --

$$\underline{)\text{-}|\text{-}(}_5 \;=\; \underline{i}^{2^5} \;=\; \underline{i}^{32} \;=$$

$$\underline{i} + \underline{y} + \underline{q}_{yi} + \underline{w} + \underline{q}_{wi} + \underline{q}_{wy} + \underline{q}_{wyi} + \underline{c} + \underline{q}_{ci} + \underline{q}_{cy} + \underline{q}_{cyi} + \underline{q}_{cw} + \underline{q}_{cwi} + \underline{q}_{cwy} + \underline{q}_{cwyi} + \underline{p} +$$

$$\underline{q}_{pi} + \underline{q}_{py} + \underline{q}_{pyi} + \underline{q}_{pw} + \underline{q}_{pwi} + \underline{q}_{pwy} + \underline{q}_{pwyi} + \underline{q}_{pc} + \underline{q}_{pci} + \underline{q}_{pcy} + \underline{q}_{pcyi} + \underline{q}_{pcw} + \underline{q}_{pcwi} +$$

$$\underline{q}_{pcwy} + \underline{q}_{pcwyi} + \underline{s}$$

-- since we have not yet determined which ***actual*** categories of the ‘modern computerware’ domain, ***if any***, are described by the ‘combinatorially’ ***possible***, algorithmically-generated further **15** symbols, yet to be “solved-for” --

$\underline{q}_{pi}$, $\underline{q}_{py}$, $\underline{q}_{pyi}$, $\underline{q}_{pw}$, $\underline{q}_{pwi}$, $\underline{q}_{pwy}$, $\underline{q}_{pwyi}$, $\underline{q}_{pc}$, $\underline{q}_{pci}$, $\underline{q}_{pcy}$, $\underline{q}_{pcyi}$, $\underline{q}_{pcw}$, $\underline{q}_{pcwi}$, $\underline{q}_{pcwy}$, and $\underline{q}_{pcwyi}$.

But we do know how to characterize the ***possible*** categories that these **15** category-symbols "call for", viz.:

- $\underline{q}_{pi}$ **[---)** $\underline{q}_{17}$ "calls for" the category of a kind of single bit units that function as whole program units.
- $\underline{q}_{py}$ **[---)** $\underline{q}_{18}$ "calls for" the category of a kind of single byte units that function as program units.
- $\underline{q}_{pyi}$ **[---)** $\underline{q}_{19}$ "calls for" the category of a kind of $\underline{q}_{yi}$ units that function as whole program units.
- $\underline{q}_{pw}$ **[---)** $\underline{q}_{20}$ "calls for" the category of a kind of single word units that function as program units.
- $\underline{q}_{pwi}$ **[---)** $\underline{q}_{21}$ "calls for" the category of a kind of $\underline{q}_{wi}$ units that function as whole program units.
- $\underline{q}_{pwy}$ **[---)** $\underline{q}_{22}$ "calls for" the category of a kind of $\underline{q}_{wy}$ units that function as whole program units.
- $\underline{q}_{pwyi}$ **[---)** $\underline{q}_{23}$ "calls for" the category of a kind of $\underline{q}_{wyi}$ units that function as whole program units.
- $\underline{q}_{pc}$ **[---)** $\underline{q}_{24}$ "calls for" the category of a kind of single command units that function as program units.
- $\underline{q}_{pci}$ **[---)** $\underline{q}_{25}$ "calls for" the category of a kind of $\underline{q}_{ci}$ units that function as whole program units.
- $\underline{q}_{pcy}$ **[---)** $\underline{q}_{26}$ "calls for" the category of a kind of $\underline{q}_{cy}$ units that function as program units.
- $\underline{q}_{pcyi}$ **[---)** $\underline{q}_{27}$ "calls for" the category of a kind of $\underline{q}_{cyi}$ units that function as whole program units.
- $\underline{q}_{pcw}$ **[---)** $\underline{q}_{28}$ "calls for" the category of a kind of $\underline{q}_{cw}$ units that function as program units.
- $\underline{q}_{pcwi}$ **[---)** $\underline{q}_{29}$ "calls for" the category of a kind of $\underline{q}_{cwi}$ units that function as whole program units.
- $\underline{q}_{pcwy}$ **[---)** $\underline{q}_{30}$ "calls for" the category of a kind of $\underline{q}_{cwy}$ units that function as whole program units.
- $\underline{q}_{pcwyi}$ **[---)** $\underline{q}_{31}$ "calls for" the category of a kind of $\underline{q}_{cwyi}$ units that function as whole units.

¿Do any such units presently exist, in the domain of 'modern computerware'***?***

- ¿Are there any computer systems in which a single bit unit functions as a program unit***?***
- ¿Are there any computer systems in which a single byte unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{yi}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single word unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{wi}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{wy}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{wyi}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single command unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{ci}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{cy}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{cyi}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{cw}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{cwi}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{cwy}$ unit functions as a program unit***?***
- ¿Are there any computer systems in which a single $\underline{q}_{cwyi}$ unit functions as a program unit***?***

Were we to find no instances of any such units, then the "categories" --

$\underline{q}_{pi}$, $\underline{q}_{py}$, $\underline{q}_{pyi}$, $\underline{q}_{pw}$, $\underline{q}_{pwi}$, $\underline{q}_{pwy}$, $\underline{q}_{pwyi}$, $\underline{q}_{pc}$, $\underline{q}_{pci}$, $\underline{q}_{pcy}$, $\underline{q}_{pcyi}$, $\underline{q}_{pcw}$, $\underline{q}_{pcwi}$, $\underline{q}_{pcwy}$, and $\underline{q}_{pcwyi}$

-- *might* all be instances of the ***generic*** *"**<u>empty</u> category**"*, ●, and we *might* "assert" our solution as follows:

$$\underline{q}_{pi} = \underline{q}_{py} = \underline{q}_{pyi} = \underline{q}_{pw} = \underline{q}_{pwi} = \underline{q}_{pwy} = \underline{q}_{pwyi} = \underline{q}_{pc} = \underline{q}_{pci} = \underline{q}_{pcy} = \underline{q}_{pcyi} = \underline{q}_{pcw} = \underline{q}_{pcwi} = \underline{q}_{pcwy} = \underline{q}_{pcwyi}$$

$$= \bullet \ \text{[---)} \ \underline{q}_{0}.$$

We *might* therefore write our *full* solution for step $\mathbf{s} = \mathbf{5}$ as --

$$)\text{-}|\text{-}(_{5} = \underline{i}^{2^{5}} = \underline{i}^{32} =$$

$$\underline{i} + \underline{y} + \bullet + \underline{w} + \bullet + \bullet + \bullet + \underline{c} + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \underline{p} +$$

$$\bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \bullet + \underline{s} =$$

$$\underline{i} + \underline{y} + \underline{w} + \underline{c} + \underline{p} + \underline{s} =$$

<u>bits</u> + **<u>bytes</u>** + **<u>words</u>** + **<u>commands</u>** + **<u>programs</u>** + **<u>software systems</u>**.

But let's keep all of our 'categorial-combinatorially ***<u>possible</u>***' categories around for a while longer --

$$)\text{-}|\text{-}(_{5} = \underline{i}^{2^{5}} = \underline{i}^{32} =$$

$$\underline{i} + \underline{y} + \underline{q}_{yi} + \underline{w} + \underline{q}_{wi} + \underline{q}_{wy} + \underline{q}_{wyi} + \underline{c} + \underline{q}_{ci} + \underline{q}_{cy} + \underline{q}_{cyi} + \underline{q}_{cw} + \underline{q}_{cwi} + \underline{q}_{cwy} + \underline{q}_{cwyi} + \underline{p} +$$

$$\underline{q}_{pi} + \underline{q}_{py} + \underline{q}_{pyi} + \underline{q}_{pw} + \underline{q}_{pwi} + \underline{q}_{pwy} + \underline{q}_{pwyi} + \underline{q}_{pc} + \underline{q}_{pci} + \underline{q}_{pcy} + \underline{q}_{pcyi} + \underline{q}_{pcw} + \underline{q}_{pcwi} +$$

$$\underline{q}_{pcwy} + \underline{q}_{pcwyi} + \underline{s}.$$

A step $\mathbf{s} = \mathbf{6}$ self-iteration would end with an '''algebraic''' category-***<u>un</u>***known described by the 'category-description' symbol $\underline{q}_{ss}$ **[---)** $\underline{q}_{64}$.

Because I believe that no 'systems of systems' -- no 'software meta-systems' '''of the second degree''', made up out of a heterogeneous multiplicity of software systems '''of the first degree''' -- are going to be familiar to many of our readers, if any, I'm declaring presentation **<u>s</u>**tep $\mathbf{s} = \mathbf{6}$ to be "'non-present''', and terminating the narration of this model here, even though some of the "cross product" category-descriptions, "crossing" category $\underline{\mathbf{s}}$ with each of the thirty-one predecessor categories of step $\mathbf{s} = \mathbf{5}$, might turn out to have actualized meaning / not to have the value 'full zero', ●.

Our categorial progression so far can be summarized textually as below.

The '<u>qualo-fractal</u> content-structure' of this 'psychohistorical dialectic' to **<u>s</u>**tep **5** can be summarized as follows --

<u>Computer Software Systems</u> "contain" **<u>Computer Programs</u>**, which "contain" **<u>Computer Commands</u>**, which "contain" **<u>Computer Words</u>**, which "contain" **<u>Computer Bytes</u>**, which "contain" **<u>Computer Bits</u>**.

The "four symbolic-elements expression" for this model is thus $\underline{i}^{2^{5}}$ [four if we count the underscore under the $\underline{i}$ as a separate "symbolic-element"].