[redacted]d [redacted]g [redacted]s" [redacted]n [redacted]g [redacted]r [redacted]s: [redacted]ty [redacted]m [redacted]r [redacted]t [redacted]on [redacted]n [redacted].

[redacted]s [redacted], [redacted]ty [redacted]t [redacted]" [redacted]c [redacted]d [redacted]d [redacted]. [redacted]s [redacted]ty [redacted]t [redacted]n [redacted]: [redacted]y [redacted]t [redacted]ally [redacted]e [redacted]d [redacted]on [redacted]c [redacted]s [redacted]en [redacted]c [redacted]. [redacted]al [redacted]h [redacted]lly [redacted]e [redacted]n [redacted]. [redacted]e [redacted]t [redacted]al [redacted]n. [redacted]l [redacted]t [redacted]l [redacted]: [redacted]s [redacted].

[redacted]d [redacted]s [redacted]'s [redacted]" [redacted]g [redacted]m [redacted]s [redacted]h [redacted]m [redacted]s [redacted]s [redacted]s [redacted]n [redacted]). [redacted]y [redacted]d [redacted]f [redacted]t [redacted], [redacted]h [redacted]h [redacted]r [redacted]y [redacted]r. [redacted]h [redacted]r [redacted]r [redacted]t" [redacted]tic [redacted]". [redacted]n [redacted]. [redacted]d [redacted]t [redacted]g [redacted]" [redacted]s [redacted]y [redacted]d [redacted]" [redacted]—see [redacted]A [redacted])

[redacted]t [redacted]. [redacted]y [redacted]m [redacted]s [redacted]r) [redacted]. [redacted]e [redacted]n [redacted]: [redacted]n, [redacted]e [redacted]s [redacted]d [redacted]e [redacted]s. [redacted]n [redacted].

[redacted]on [redacted]cs [redacted]— [redacted].

Appendix III.

ELUCIDATION OF TOTAL ATTRIBUTE DISTANCE AND THE FINITE EXPONENTIAL OPERATOR

By David McGoveran

Alternative Technologies
150 Felker Street Suite E
Santa Cruz, CA 95060

A number of papers by myself and (both jointly and independently) by Pierre Noyes have made use of the concepts of total attribute distance, the finite exponentiation operator, and the transport operator. Each of these depend on a method of counting that is unfamiliar to most readers. Inasmuch as the papers have generally been couched in terms of bit strings and the original explanation of these concepts was not elucidated in terms of bit string exemplars, the reader has been left with an undue burden of interpreting a difficult and new conceptual basis for counting as presented in the ordering operator calculus. In some cases, our efforts have contained errors which created a further muddle.

It is my intent in this note to correct the problem, an obligation which I have too long and unintentionally avoided. I wish to point out from the beginning that the original reference was never intended to explain the issues which have risen from applications of the ordering operator calculus. Instead, a "Foundations II" paper has been long planned and was intended to treat the concepts of (1) "non–Euclidean" d–spaces—the definition of a metric introduced in Foundations is positive definite, an unnecessary and generally untrue restriction in non–Euclidean spaces; (2) a mathematical mechanics for dealing with the interactions of ordering operators—this is essential for a thorough understanding of my derivation of the fine structure constant and related problems; and (3) a more general exposition of the relationship between a model of a system and its representation.

Within the current context, this last item is most important. In general, there are always two ways to express context information which will preserve the statistics of a given model. This first is the most difficult part of the ordering operator calculus to keep in mind—the interpretation of the symbols is context sensitive. A context sensitive mathematics has been generally abhorred in both the mathematical and the scientific communities: I hope to show that it has its uses. The second way of expressing context information is to invest a separate symbol for each specific context, under the assumption that the semantics is separable from the syntax. The ordering operator calculus allows this only under the conditions set out by the Separability Lemma.

A significant aspect of the ordering operator calculus is the modeling methodology. A major concern of mine in developing the calculus was to be able to build two arbitrary and not necessarily complete representations of a given system in a common language and then to express the degree to which information preserving transformations will be of unequal representational power—one of the systems will not be rich enough to express all the concepts expressible in the other. This leads to the concept of hidden information.

Understanding hidden information is a good thing: it shows how new information can arise between the interaction of two systems. This is especially true in terms of the statistics of the interaction, as we shall see below. In fact, the combinatorial hierarchy itself is best understood in these terms from my point-of-view—there is a model of an underlying generation scheme characterized by the sequence 3, 7, 127, ... and there is the model of a vector space by which this generation is to be represented. The interaction between the two gives certain statistics and provides a "stop rule" for the combinatorial hierarchy.

Now to the subject at hand. I will not try to relate this treatment to physics—I leave that for my colleagues. Nonetheless I will speak in terms of bit strings with the hope that the topic will be clearer and that translation into useful physics will be easier.

First consider the population of binary bit strings of size N composed of the symbols 1 and #. Each 1 in these strings will represent the occurrence of one distinct event or object which belongs to a certain equivalence class of such events or objects (I will use event forthwith to conserve space and typing). Each # represents a non-event. This is different from saying that # represents a non-occurrence. Each # conveys no information whatsoever about the event—it only holds a place where such an event might have occurred, but did not as far as we know. Thus the "event" is unknown, and we cannot say that it is the complement of a 1 type event (1-event for short). The # just pads each string of k 1's to size N, their position being unimportant. The 1-type equivalence class will be said to contain P (or Q) distinct events—this is called the number of increments I in Foundations.

Suppose that each 1-event is labeled to designate its distinctness in the equivalence class. A common way to do this is to use ordinal labels and create the bit string so that the ordinal labels are in ascending order positionally right-to-left.

When the bit strings are created by sampling from the equivalence class without replacement, we can drop the labels and simply use the ordinal position as an implied label. Then we need some place holder for positions, that are never filled unless the sampling is exhaustive or the size of the sample k is less than P . If, however, we allow for sampling with replacement, the labels cannot be dropped if we are to distinguish strings.

Suppose we ask how many strings can be generated by sampling the 1-events with replacement subject to the constraint that the resultant strings each contain k 1's. Since the # symbols do not convey information with which we are interested for the moment, they can be ignored—dropped from the string altogether: what matters to us is the size of the sample k and the size of the equivalence class.

$$(1_4\#1_3 1_2 1_1\#) = (1_4 1_3 1_2 1_1)\ . \tag{1}$$

Then, for a bit string containing k 1's, the number of bit strings possible is just

$R(N, k, P)$:

$$R(N, k, P) = P^k \ . \tag{2}$$

Now suppose that we have reasons to believe that those #'s are important. There are two statistics that can be given immediately: the number of permutations and the number of combinations.

Consider the number of combinations $C(N, k)$ of k 1's in a string of size N. $C(N, k)$ is just the number of distinguishable strings where the occurrence of #'s matters, but the 1–events are not themselves labeled. The idea there is that it is the context of a 1–event in relation to the #'s and other 1–events that serves to identify it as a specific member of the equivalence class. Thus, if 1' is an artificial label to distinguish it from 1 strictly for purposes of illustration, then:

$$(1\#1') = (1'\#1) \neq (\#1'1) = (\#11') \tag{3}$$

where $\neq$ means "is not identical to". The point here is that a 1–event in a specific context is unique and a distinct member of the equivalence class. So permutating 1–events only serves to change their identity—it does not generate a string distinguishable from the original string.

Under this interpretation, the number of distinguishable strings is just $C(N, k)$:

$$C(N, k) = \frac{N!}{k!(N-k)!} \ . \tag{4}$$

Now consider the number of arrangements of all strings of size N with k 1's were we able to distinguish a 1–event in spite of its distinctness being defined by context, i.e. the distinction between 1 and 1' is know to us. Then:

$$(1\#1') \neq (1'\#1) \neq (\#1'1) \neq (\#11') \tag{5}$$

for counting purposes only. Under this interpretation, the total number of string (both distinguishable and indistinguishable) is just $P(N, K)$:

$$P(N, k) = \frac{N!}{(N-k)!} \ . \tag{6}$$

Now, regardless of the number of #'s in a string, the frequency probability of distinguishable strings for fixed N and k is just:

$$\frac{C(N, k)}{P(N, k)} = \frac{1}{k!} \ . \tag{7}$$

Notice that dependence on N vanishes.

Now back to our value for $R(N,k,P)$. Let P be bounded from above by the maximum number of distinct 1's to be found in the construction of the sample space of strings given by $P(N,k)$. We can now ask the key question. On the average and for fixed N, k, and P, what is the number of strings in the population of strings constructed by sampling with replacement which are distinguishable in the sense given by $C(N,k)$? This number is obviously:

$$R(N,k,P) * \left[\frac{C(N,k)}{P(N,k)}\right] = \frac{P^k}{k!} \ . \tag{8}$$

Three points of direction:

(1) In constructing the transport operator, the increment I is replaced by an operator $e\ d/dp$ since it is, in the general case, dependent on the particular parametrization of the coordinate x^i. The summation of terms like (8) for all values of k from 0 up to some K leads to the finite exponential. This corresponds to constructing a network of discrete Feynman paths where each real node is represented by a 1 and each "imaginary" node is represented by a #. It is a 1–dimensional discrete Feynman kernel. Note that the #'s are essential to the statistics. [Aside: The transport operator was constructed in non–Euclidean d–space, as were the Lorentz transformations—a fact I failed to make explicit in Foundations.]

(2) When constructing the Dirac, it is essential to note that the P (right turns strings) and Q (left turns strings) are constructed independently. They are allowed to mesh because of the #'s in each string which preserve a global context or ordering. If the number of distinguishable P and distinguishable Q strings is suitably normalized to the population of all possible strings (this will be constrained by the physics), then the joint probability of the P and Q strings being distinguishable is found by multiplying the independent probabilities.

(3) A representation of ordering operators which I have been using for some time is that of the generator or walk of a directed graph. Any particular directed graph can be represented by an $N \times N$ transition matrix: all nodes are given ordinal labels. There is then one row and one column in the matrix for each node and a 1 in a cell represents a connection from the row node to the column node. In this context, it is interesting to note that $P(N, k \leq N)$ is the number of sub–matrices with exactly one 1 in each of the N rows and exactly one 1 in each of the k columns. This completes the mapping from bit–strings to ordering operators and simultaneously shows that the permutations correspond to a special orthogonal decomposition of all possible ordering operators.

[redacted]ix [redacted]).

[redacted]s [redacted]y