INTRICACY versus COMPLEXITY

A: Introduction

The word < complex > has various interpretations and is not a <measure> of the difficulty of solving an <intricate> problem .

The formal interpretation of complexity is confined to a partition of the universe of problems in two sets using the <Touring machine> as the boundary. criterion .

A problem is either solvable by the <Touring machine> or is not .

Here, the suggestion is to create a measure similar to a physical extensive measure like masse, volume, energy, entropy, cost and build the various conjugated intensive functions similar to density (masse/volume), temperature (energy/entropy) , price (cost/quantity) .

Some typical specific function derived from <intricacy> are :
intricacy/time, intricacy/cost , intricacy/number of cycles, intricacy/information .

Problems could be compared by their respective <intricacy> or by the many conjugated <specific intricacies> that can be constructed .

B: General Assumptions

B:0 Reserved words.

Some words are reserved for the sake of clarity and precision .

<Entity>, EN, is any thing that can be subjected to observation and eventually capable of acting e.g.: living or virtual beings, artefacts and also acts, conjectures, images, assumptions, judgments . EN is a substantive word . Existence and observation are distinct concepts and an entity is declared real if observable and virtual if conceived but not yet observed .

<Forma>, all entities can receive , support, contain and envoy information , <forma> . The truthfulness of <forma> varies from absolute truth to absolute falsehood . The transportation speed of <forma> is that of the conveyer, the entity that contains or supports that <forma> .

<Language>, LG, is a set of sentences and rules, endowed with a structure . It is very rare that all sentences Px of language LGa can be translated to sentences Qy of language LGb and inversely that Px is the retro version of Qy .Translation <deforms> <forma> : traduttore or traditore .Two languages are equivalent if the following condition is observed : for al sentences of both languages , $Pv = V^{\circ}(V(Pv))$, V means version and V° is the retro version and Pv is a sentence of one of the two languages .

<Problem>, PB, means a question that can be described in some language, LC, not necessarily a formal language and an answer or solution is required .

<Program>, PG, is the translation of a problem, PB, into a formal language, namely a computer language .

B:1 Universal sets, U .

All universal sets are assumed finite . All computers operate with words with a limited number of characters (0,1) and this restricts the precision of the word to the first N characters.

B:1:1 Universal set of Entities, UEN . UEN is the universal set of entities that intervene as actors (agents, ents, things, etc) and all other observable ents that participate in the problem. The entity, ENpq is a member of a set ENp contained in UEN. UEn is a finite set .

B:1:2 Universal set of Attributes , UAT. Entities have proprieties, predicates and attributes (adjectives) . UAT is the universal set of attributes and is a finite set. The attribute, ATgh is a member of the set of attributes, Atg , contained in UAT.

B:1:3 Universal Set of Actions , UAC . Computer programs are translated in hardware language which is <understood> by the computer and a finite set of <elementary tasks> can be executed . These elementary tasks will be named actions or acts. The act, ACkj, is a member of the set of acts, Ack, that is contained in the finite universal set , UAC .

B:1:4 Universal Set of elementary sentences of a Formal Language , UFL . The problem, PR, must be translated and described in a formal language, namely a computer language . The universal set of <elementary sentences> of the computer language , ULG is

given and finite . An elementary sentence, FLmn is a member of the set of elements , FLm contained in UFL . Composite sentences can be formed but in a numerable number .

B:1:5 Universal Set of Species, USP . It is assumed that all <species>, SPm, are homogeneous which means that a formal similarity of its members is observed . The concept of similarity implies the equivalence of its members and that is achieved by choosing a finite set of attributes that all members of SPm must possess . SPmj is a member of the specie, SPm, contained in USP .

C: The Measure of <intricacy>

C:1 Objectives

The measuring of <intricacy> is assumed to satisfy the following objectives :

C:1:1 All qualified programmer and computer operator will be able to calculate the <intricacy> of the program if the method proposed is correctly applied .

C:1:2 The method contains a certain number of <invariants> that are chosen arbitrarily. Each university or research department may choose a different set of <invariants> but agreements made between establishments may contribute to the adoption of common set . In time, eventually, an a universal set will be adopted .

C:1:3 The method includes the partition of the program and many different partitions can be conceived and the choice is made 'ad libitum' by the computer operator . Different partitions may induce different intricacies , see C:1:5 .

C:1:4 The evaluation of the value of the intricacy of a program must be independent of the operator provided that the invariant parameters and the partition chosen are the same and the method is correctly applied .

C:1:5 The method is evolutive, even if the parameters are the same, a better program or a different a partition of an old program not yet tried may reduce the intricacy .

C:1:6 The solution of a program may not be the solution of a problem an account of the eventual errors made in the translation of the problem into a program.

C:2 The Method

C:2:1 Definitions and Symbols

A modem computer is taken as the referential instrument .

The virtual or real problem to be translated in computer language will be symbolised by PB.

The program describing the problem, PB, will be referred as PG .

The program used to evaluate the <intricacy> is symbolised by I(PG) or I:PG .

C:2:2 Partition of the Program PG .

A program can be understood as a structured set of <acts> as for instance : reading a char, an integer or a real either from disk or memory or executing a sum, product , etc. etc. .

The universal set of <acts>, UA, is finite and UA represent all the available elementary operations . All <complex acts> are build with a finite number of members of UA .

Each <act> is a member of a <specie>, SPk , see B.1:5 .

Obviously, no <act> can be a member of two different <species> .

The set {SPk : k} $\subseteq$ USP , where k is the index of all species that participate in solving the program PG . The set {SPk : k} performs a partition of PG .

C:2:3 Specific <intricacy>, $\hat{I}$, of a specie, SPk, is $\hat{I}$:SPk or $\hat{I}$(SPk) .

The <intricacy> of all members of a given <specie> is the same on account of the similarity of the members which is a basic assumption and this common value of specific <intricacy>, $\hat{I}$:SPk, is a fundamental attribute of the <specie>, SPk.

The values of $\hat{I}$:SPk are arbitrarily chosen and this task is important but unavoidable .

The cardinals of USP and SPk are of the same range of values because difficult problems imply, in general, the use of all the members of USP .

C:2:4 Intricacy of a Set.

The intricacy of a subset, SPkj, of the <specie> SPk , SPkj $\subseteq$SPk, is defined as the product of the specific <intricacy> of the set, $\hat{I}$:Sk, and the cardinal of the subset, SPkj, e.g.:

I:S Pkj =Î:S Pk .card(S Pkj). This definition follows the usual method of creating a linear measure . Other non linear function can be adopted .

C:2:5 Intricacy of the program PG .

A simple and linear formula permits the evaluation of the <intricacy> of PG : I:PG=ΣΣNij x Î:SPi , where Î:SPi is the symbol of the specific intricacy of the specie S Pi and Nij is the number of times a member of the specie SPi intervenes in the process.

C:2:6 Stopping a Run

It is very rare that a running program completes the task and stops . In general, the program must be stopped and an adequate procedure should be implemented . Typical criteria for stopping procedures are for instance : the error of the calculation or the difference between to successive values is less then an arbitrarily value, the number of cycles or the time consumed and of course by the computer operator . See : D:6, N1 and N7.

D: Comments

D:1 Measure of <intricacy>

If all <species> have the same <intricacy, the formula boils down to the sum of all members of all <species> multiplied by a parameter .

The number of times an elementary <task> is activated must be counted . The counter may be installed every time the task is performed or indirectly at the level of procedures and functions provided that is known the number of times each elementary task intervenes . The counting & classification operations are done by the computer that runs the program .

D:2 Descriptions, Problems and Programs

The description of the <real problem> in a computer language is the cause of many difficulties namely : introduction of ambiguities, voids of information, unjustifiable cycling, etc , all these imperfection may convert a rather simple problem in a very large and even an unsolvable one .

Different programs, PG1..PGj, with different values of intricacies, I:PG1..I:PGj, can be build to solve a well stated problem, PB . The program with the minimum intricacy, will be chosen to represent the intricacy of the problem PB ,

The intricacy, I:PB, of a problem is a time diminishing function depending on better descriptions or new programs .

D:3 Counting task

Essentially the time to run a program is proportional to the cardinal of the universal set of elementary <acts> ,UT .

Instead of UT it is possible to construct a set , UTi, that includes procedures, functions and elementary <acts> . The counting will be much reduced and simplified using Uti instead of UT because the specific intricacy, Î: , of the procedures and functions included in UTi are previously evaluated .

D:4 Invariant Parameters

The definition of a <specie> is dependant of the concepts and definitions of similarity of the <acts> and degree of homogeneity of the <specie> .

These concepts should be examined taking in consideration that the main objective is to build a set with a reasonably comparative intricacy . Some typical cases are presented in note N5 .

D:5 Elements and Composites

The concepts of element , (holon) and aggregation of elements, (composite) is essentially arbitrary e.g.: the elements of the classical chemistry , the members of a family , the musical notes of a song , the dots of a printing or the pixel of a digital camera are all elements but also quite different entities and all could be further partioned .

When an agreement is reached regarding the set of <elemental entities> , H, then all other entities must be considered aggregations and compositions build exclusively with the members of the set H .

The word <holon> will be used instead of <element> to enhance the unavoidable arbitrary nature of the choice of the set H ..

D:6 The stopping of programs

In general, programs are stopped when some external rule is fulfilled , for instance : number of cycles, computation time , a given value of a stop function . See N1and N7.

NOTES

N1:The choice of the <holonic> set of the Computer language ,H .

It is assumed that all <acts> executed by a computer can be constructed with the members of a finite set of <acts> that will correspond ,1-1, to the holonic set , H, of the computer language .

H0	void		
H1	read an hard disk		
H2	write on a hard disk		
H3	read the memory		
H4	write on the memory		
H5	execute the operation	A , addition,	
H6	"	"	S , Subtraction,
H7	"	"	M, multiplication,
H8	"	"	D , Division.
H9	"	"	U , union, .
H10	"	"	M, intersection
H11	"	"	N , involution , named <not> if the universal set has only 2 elements .

A more accurate study of computation would either increase or decrease the number of holonic acts or even suggest an altogether new set .

The general assumption is that all acts or tasks a computer is able to perform can be constructed with the chosen set H of 12 <holonic acts> . .

N2: The <intricacy> of a member of the set H .

The tasks included in H are very different, take dissimilar time to accomplish and imply costs associated to the instruments and computers used .

The <specific intricacy>, $\hat{I}j$, of each Hj , j in [0..11] , can be chosen by different criteria, namely :

a: $\hat{I}j = 1$ for all j in [0..11] . It is a very simple choice and totally independent of the hardware , the time consumed or the economic value .

b: $\hat{I}j = T$, the time to accomplish the elementary task ,j . This method depends strongly of the computer but some independence could be attained if the set of $\hat{I}j$ initially chosen would be applied independently of the computer used .

c: $\hat{I}j = E$, the economical values of the elementary acts j . The same comments expanded in b: are also apply able .

N3: The <intricacy> of a Program .

Some examples of composite tasks are in order :

a: Summation and multiplication of a given number of terms .

b: Integration with a given number of steps .

c: Tasks to be performed till a conditional stop instruction intervenes .

d: Procedures and function that are frequently used .

The main purpose is to reduce the counting of occurrences ..

Some preparation should be performed, namely :

§ The program is analysed and partitioned in a reasonable number of sub-programmes , procedures, functions and simple tasks . The general rules of a partition must be fulfilled , namely the parts are disjoint and their union reconstructs the program .

§ The intricacy , I:Pk of each part , Pk, of the partition was calculated previously and a counter was associated to measure the number, Nk, of times the part Pk was used .

§ Usually, the intricacy, I of the program is a linear function of the parts : $I:PG=\sum \hat{I}:Pk \cdot Pk$

§ The parts being disjoint does not mean that the number of times, N, that each part is used must be different .

§ If N is the same for a subset of parts, only I counter is implemented .

N4: Choice of <holons> .

§ Logic <acts> (and , or, neither , nor, etc) should they be include in the same <specie> ? or it would be better to create two <species> ((and, or) ; (neither, nor)) or even 4 <species> (and, or, neither, nor) . The solutions presented are progressively more homogeneous but the number of species grows from 1 to 4

§ The set (+, -, *, /) should be considered one <specie> ?..Are the members of the sets (+, -, *, /) and (and, or, neither, nor, not) sufficiently similar regarding <intricacy> to justifying the creation of 1 <specie>

§ The sets disc (read ,write) and memory (read, write) should be distinct . If considered distinct the problem is to define their relative values .

An example of attribution of <intricacy> of values is presented in note N5

N5: Attribution of <intricacy> values to tasks .

The need of a specialist in hard ware is essential and what follows must be understood as a simple exercise to explain the method .

Some tasks are typical : read/write in memory, read/write in disc, logic and algebraic functions , etc. the time and memory that is used is quite different and the <intricacy> of these tasks should be adjusted accordingly . Symbols : I = intricacy ï = unit of intricacy .

I(read memory) = I(write memory) =5 ï but I(read disk) = I(write disk) =15 ï

I(sum real) = I(sum integer) =6 ï but I(sum complex)=22 ï

I(sum vector)= dimension of the space x10 ï

I(log real) = 35 ï I(exp real) = 24 ï I(Exp(n.Log(x)) = 45 ï

I(A and B) = 9 ï I(not(A) = 7 ï I(neither(A, B)) = 14 ï

The programmer can apply the same method and evaluate functions and procedures that are consistently used in the program .

The final objective is to build a set of tasks, the holonic tasks, the correspondent values of <intricacy> and the. number of counters to be included if the program .

The product of the number of times an holonic task is invoked multiplied by its value gives the intricacy contribution of that task . In fine, the sum of the contribution of all tasks is the intricacy of the program . Note that specific contribution Î is generally the relation : I / number of cycles and Î may converge .

N6: Convergence versus intricacy

The translation or description of a problem into a computer language, L, is not always feasible and some comments are appropriate :

1: The tasks are innumerable or some sentences have forms that have no correspondence in CL , eventually, some partial images of the system can be translated in CL . .

2: The problem is about a system, SY, intrinsically unstable and divergent and certainly non linear although translate able into a computer language. A stop procedure must be included . In some cases , the intricacy, I:, may converge, I:(step n+1) – I:(step n) tends absolutely to zero .

3: The system S does not converge but after some steps the trajectory of the system is confined to a numerable set of singly connected domains and the system jumps from one domain to another . A counting operation should be implemented to obtain the <residence probability> of the system in each domain . In this case the program can be stopped when the regions above referred are reasonably defined The <specific intricacy> Î = I / number of cycles and the residence probabilities of each attractor will be considered attributes of the system .

4: The system S converges absolutely to one state and the intricacy converges , never the less a stop procedure should be implemented .

5: When intricacy, I:, increases at each new cycle , it is suggested to study the evolution of a specific intricacy , Î: as for instance : intricacy per cycle : Î: = I: / Nc , Nc is the number of cycles . Both I and Nc increase at each step but Î may converge to a finite value.

N7: Problems and Programs .

The purpose of N7 is not to solve problems like the following ones :

1: is it possible to translate every real physical problem into a formal language .
2: do formal languages synthesise the information contained in the available data
3: can a real or virtual computer solve all problem written in a computer language ,
4: is it always feasible to write a program to identify if a program will stop spontaneously , without recourse to experimentation ..

The points 1 and 2 are examples of the importance of the mathematical instrumentation used , for instance problems described by hyper geometric or Navier -Stokes functions that may be considered NP problems, if complex derivatives are employed and not the usual integer derivatives can be converted in a polynomial problem .

The images of the reality are very imperfect , considering the following motives :

a:: computers work with N-strings of 0 and 1 and N is finite . All real number can be converted to an integer number with the same precision .

b:: experimental observations are evaluated with errors either represented by probability distributions or limits (max,min) .

c:: physical models are constructed with the assumed condition that the <remaining Universe> not referred or measured does not interfere with the observation or experimentation .

d:: humans are not endowed with brains that could cope or digest a model of the Universe that would explain every thing .

Scientific knowledge is a collection of regional models and their respective formulas .

N8: Langnages .

All living beings , <biotas> , B, have painfully created one or more virtual symbolic structures to describe the surrounding world , friends and foes, their actions his reactions ,thoughts and dreams .

Without these structures <biotas> and their aggregations would have great and many difficulties to survive and replicate.

These structures will be referred as <languages>, L, and are essential both to an acceptable life in a community but also to a profitable and neighbourly interaction with the outside world .

The translation in a given language, L, of o real world act, fact, idea , object , is an imperfect and subjective operation and the finite set of sentences, {Sa..Sg}, of the language L is a more or less deformed image, Img, of the reality .

Not withstanding the unavoidable imperfection of images, biota, B, must act or react taking in consideration not only the set {Sa..Sg} but also the set {Sp..Su} of many sentences that describe past experiences that B kept in memory or in external artefacts.

Preparing a suitable reaction is building a list of acts, Ab, that B considers an adequate and <rational> reaction but are not necessarily the best . .

The action of biota, B, Ab, can be summarized by the expression : Ab = Ob({Sa..Sg} , {Sp..Su}) .

The outside world, W, will react to the image of the action A, Img(A), created by W . The reaction of W results of the interpretation of the image , Img(Ab) and not of the action Ab.

The reaction of W will be build in similar way and Aw=Ow({Zd..Zk},{Zn..Zv}) , where {Zd..Zk} is the set of sentences that describe the image of Ab and {Zn,.Zv} is a sub-set of the memorised past information of W . The rules Ow and Ob should be the same but may be applied differently conveying distinct meanings .

This verbal <loquation> may diverge and culminate in war or converge to peace .

N9: Compnter Language

Boolean language (2-adic logic) , the universal set has 2 elements are {0,1}, the operators are : 2 binary , $\cap$, $\cup$ (disjunction and union) and 1 unary , $\sim$, (negation) .

A n-adic logic and n finite , the universal set has n elements are {0..n-1} . the operators are : 2 binary , $\cap$, $\cup$ (disjunction and union) and 1 unary , $\sim$, (order involution) .

N10: Classification of Complexity

Some attributes or proprieties of the program can be used to classify its complexity and by extension the complexity of the problem . A good example is :

A1: Logarithm of the execution time, LT, a real number .
A2: Logarithm of the memory capacity needed by the program, LC, a real number ,
A3. The program is polynomial or not, NP , a 2 valued Boolean ,
A4: The required computational capacity grows exponentially with time, XT, a 2 valued Boolean .

A5: The required computational capacity grows exponentially with capacity, XC, a 2 valued Boolean .

The ordered set {A3,A4,A5} enables the partition of a problem in 8 classes :

{0,0,0} , {1,0,0} , {0,1,0} , {0,0,1} , {1,0,1} , {1,1,0} , {0,1,1} , {1,1,1}

If A1 or A2 are included then let a1 and a2 be 2-valued Boolean functions representing that inclusion . The ordered set is now {a1,a2,A3,A4,A5} and the partition would have 31 classes :

{0,0,0,0,0} = [5:0] = (void) {1,0,0,0,0} = [5:1] = 5 {1,1,0,0,0} = [5:2] = 10

{1,1,1,0,0} = [5:3] = 10 {1,1,1,1,0} = [5:4] = 5 {1,1,1,1,1} = [5:5] = 1

Classification is not a measure and does not quantify complexity , only A1(time) and A2 (memory capacity) are variables that may be used as quantifiers .

Both, A1 and A2, are strongly dependent of the computer used and the log function eliminates the possibility of a linear measure .

The introduction of specific – functions is eventually a better solution .

N11: Transvection, (product), T .

If <intricacy> is applied to non linear functions and multi dimensional curvilinear spaces then note 2 should be read but if only scalar functions and one dimensional vectors are used then note 2 is irrelevant .

Transvection is the generalisation of the product of vectors to affin non linear geometry . An affinor corresponds to a tensor not necessarily symnetric and transvection corresponds to a product of vectors or tensors . .

Some concepts and symbols are needed to enable the expose .

a: A_{abcdef} is an affin 6-dimensional space and the order of the directions of the space is the order of ab..f

b: $W_a^{bc}{}_{def}$ is an affinor covariant in 4 directions, (a,d,e,f) and counter variant in 2 dimensions (b,c) .

c: $U^{a}{}_{bc}^{def}$ is an affinor conjugated to affinor $W_a^{bc}{}_{def}$.

d: The application of the transvection operator, T, to all 6 directions, abcdef,, of the pair of affinors $(W_a^{bc}{}_{def} , U^{a}{}_{bc}^{def})$ is a scalar, $SC = T_{abcdef}(W_a^{bc}{}_{def} , U^{a}{}_{bc}^{def})$

e: Note that $T_{acf}(W_a^{bc}{}_{def} , U^{abcdef})$ is not scalar but an affinor $V_{..}^{bcd..}$

f: For instance, V^{abc} is a counter variant vector that represents a volume in a 3 dimensional equi–volume space and density, D_{abc} , is covariant vector representing a density and both conform a pair of conjugated affinors (V^{abc} , D_{abc}) . Applying a product, (transvection), T_{abc} , to the pair of affinors a scalar is obtained , $M = T_{abc}(V^{abc} , D_{abc})$, known as mass . .

g: When what is known or measured is M and V then the value of D can be obtained very simply by the expression : $D_{abc} = M / V^{abc}$.

h: The specific propriety of a substance is more typical then the quantity , namely the density is a propriety of specie and mass and volume are associated to the quantity or amount participating in the process .

i: <Intricacy> , I: and the number of cycles, N, are both extensive functions and both grow as long as the program is running but the <specific intricacy>, $\hat{I} = I/N$ may tend to a finite value.

$\hat{I}$ is the important attribute of the program and not I .

N12: Real to integer conversion and Computation of π .

Real numbers are converted in an integer with the same approximation by means of change of unity e.g. : the real number, 234,56789 meters , with a standard error of 0,01 meters would be converted into 23456 centimetres .

The computation of π is a good example because the value of <intricacy> is , in general, divergent and tends to infinity . The value of π depends of the precision imposed by the problem , e.g.: π can be 3,14 or 3,1416 and in general 3, ab..n and the error would be less then $\Delta\pi = 0.0..(n+1)$. The value of <intricacy> , I:, increases monotonically with the inverse of Δπ and the product of I: by Δπ can be of some interest . In general, three typical cases may occur namely : the product diverges, converges to a finite value and sometimes to zero .

Symbols

UEN	Universal set of Entities	$Enpq \in ENp \subseteq UEN$
UAT	Universal set of Attributes ,	$ATgh \in ATg \subseteq UAT$
UAC	Universal set of Actions ,	$ACkj \in ACk \subseteq UAC$
UFL	Universal set of elementary sentences,	$FLmn \in FLm \subseteq UFL$
USP	Universal Set of <species>	$SPab \in SPa \subseteq USP$

PR	Problem , described in a non formal language ,
PG	Program written in a formal language .
I:	<Intricacy> of a Program e.g.: I:(PG) or I:PG .
Î:	<Specific Intricacy> e.g.: Î:(PG) = I(PG) / an extensive propriety of PG
<act>	Actions , tasks , e.g.: elementary <acts> or multiple <acts>
<holon>	similar to the concept of elements of a set .
L, LG	Language , CL means computer language.
T, TR	Transvection is a generalised product of vectors or affinors, see note N2 .

Bibliography :

As this is a working paper the suggestion is limited to the book of John D.Barrow translated in Portuguese under the rigorous direction of Professor J. Felix Costa .

The extensive list of references of this book is a trove of information .

" IMPOSSIBILIDADE" , Os limites da ciencia e a ciencia dos limites, Bizâncio .