In Quest of Information in Algorithmic Processes[*]

Anatol Slissenko
Laboratory for Algorithmics, Complexity and Logic (LACL),
University Paris East Créteil (UPEC), 61 av. du Gnl de Gaulle
94010 Créteil France
slissenko@u-pec.fr

Abstract: Intuitively, an algorithm that is computing the value of a function for a given input, is extracting an information from the input and is processing this information. During this work the information being processed 'converges to the result'. We show by examples that one can measure this information convergence. No general theory is presented, though some questions that arise on the way towards such a theory are discussed. We start with an approach based on approximations of the graph of the computed function; we call this approach semantical. It works only for algorithms that mainly calculate but do not analyze the information being processed. We extend this approach by using syntactical considerations. The principle underlying the estimations of information convergence, is that of 'maximal uncertainty' that we impose on the algorithm under analysis.

The problem of information convergence brings up conceptual questions "What is information?", "What is uncertainty?" related to similar questions more and more intensively studied in philosophy.

Keywords: algorithm, information, entropy, information convergence

Introduction

The motivation to study algorithmic processes from informational viewpoint comes from theoretical curiosity aimed at a better understanding of the work of algorithms. This may give a better insight into how to design algorithms, how evaluate their complexity, what is practical algorithm and what is practical problem.

Intuitively, an algorithm, when processing its input, is extracting and transforming information. What is this information and how to measure its evolution? The paradoxicality of mathematics is that mathematics does not consider information in our intuitive sense. What is called information in mathematics (we speak about the traditional viewpoint) is entropy, i.e., the average value of the information function in probabilistic models where all we know about a piece of information is its probability. In philosophy literature, e.g., (Adriaans, 2013; Floridi, 2013) (I do not pretend to be knowledgeable about philosophy papers on the subject, so the cited references are to be taken accordingly), one can find arguments that for the analysis of various aspects of information a more detailed view on the subject is necessary.

In our context it is useful to distinguish (in the flavor of distinguishing "data", "semantics of information" and "quantity of information" in philosophy literature):

- information,
- knowledge contained in an information,
- quantity of information.

An information (a datum) is usually received as a signal, and most signals can be coded by mathematical structures. For our limited context we consider as a piece of information, an admissible interpretation of a vocabulary in logical sense (this is illustrated by examples below).

The knowledge contained in an information has no absolute meaning. It depends on the inference system that is used to extract the knowledge and on the queries to the knowledge, i.e., on what one wishes to know. In our context the knowledge is represented by simple, easily formalizable properties of the function that is being computed. In philosophy one can find larger

[*]Partially supported by French "Agence Nationale de la Recherche" under the project EQINOCS (ANR-11-BS02-004).

and more diverse visions of semantics of information; we mention some of them, that may be related to the problem of information convergence, in Conclusion.

The quantity of information is the main subject under consideration in this paper. The traditional notion of entropy is used to measure the information convergence though it gives values that are not always sufficiently intuitive; but at present I do not know a better way to do it. The entropy is defined due to a probabilistic space. In our case there is no randomness (note that the classical probability theory is a special part of the measure theory, and neither needs any intrinsic notion of randomness). We choose probabilistic measures just to model the information convergence. This choice is governed by a principle of *maximal uncertainty*, that says that the algorithm under study has the minimal certainty (the minimal knowledge) about the next local value to compute (as well as about the final result).

1. Information and Entropies in Mathematics: a Very Brief Reference

As it was mentioned in Introduction, in mathematics information is traditionally defined in terms of entropy. We recall here the standard notion of entropy (some other are listed at the end of this section) in terms of the 'partition' approach, e.g., (Martin & England, 1981), that is more general than the classical approach based only on probabilistic distributions. The partition approach is more relevant to our context.

Let $(\Omega, \mathcal{Z}, \boldsymbol{P})$ be a probabilistic space, where Ω is a set, $\mathcal{Z}$ is a sigma-algebra over this set (in our context we may consider $\mathcal{Z}$ to be the set of all subsets of a countable set Ω), and $\boldsymbol{P}$ is a probabilistic measure, that is $\boldsymbol{P}(\Omega)=1$ and $\boldsymbol{P}(\cup_i Z_i)=\sum_i \boldsymbol{P}(Z_i)$ for any countable set $\{Z_i\}_i$ of disjoint elements Z_i of $\mathcal{Z}$. The elements of Ω are often called *outcomes* and that of $\mathcal{Z}$ are called *events*. In examples we deal only with finite spaces related to the inputs of a fixed size.

Let ζ be a partition of Ω (i.e., a set of disjoint subsets of Ω whose union is Ω). Due to our remark concerning $\mathcal{Z}$, any partition consists of measurable sets, moreover we limit ourselves to *finite partitions*. One may look at such a partition as at a probabilistic classification of elements of Ω. Here 'probabilistic' can be seen as a way to describe 'partial knowledge'.

Information function $\mathcal{I}(\zeta)$ of a partition ζ is a function over Ω that is equal to $-\log \boldsymbol{P}(z)$ on any element $z \in \zeta$. So this function is constant for all $\omega \in \Omega$ that are in $z \in \zeta$.

For a partition into m subsets of equal probability $\frac{1}{m}$, the value of the information function is $\log m$ everywhere. This value $\log m$ can be interpreted as the number of bits sufficient to determine a particular 'class' of the classification represented by the partition. If an element of partition has measure $\frac{1}{m^2}$ then the information function on this 'class' is $2\log m$, and within the previous interpretation one needs more bits to represent this less probable 'class', however on the average this is 'compensated' by the low probability of this class.

In mathematics this information function plays a secondary role, and the main 'informational' characteristic of a partition is the average of the information function, i.e., *entropy*:

$$H(\zeta) = -\sum_{z\in\zeta} \boldsymbol{P}(z)\log \boldsymbol{P}(z). \tag{1}$$

The entropy is also viewed as a measure of uncertainty (or, in some way, of knowledge).

For the case of the uniform distribution mentioned just above, the entropy is $\log m$ that gives the maximal uncertainty in the description of events (elements of partition). On the other hand, if we know that one event has probability $(1-\varepsilon)$, and hence the other ones are little probable (here ε is assumed to be 'very small') then the uncertainty of the outcome of this particular event is $-((1-\varepsilon)\log(1-\varepsilon)+\varepsilon\log\varepsilon)$ that is close to 0.

In applications, *conditional entropy* of a partition ζ with respect to a partition η is indispensable:

$$H(\zeta/\eta) = -\sum_{A\in\eta, B\in\zeta} P(B\cap A)\log\frac{P(B\cap A)}{P(A)} \tag{2}$$

Mutual information of two partitions ζ and η is defined in terms of entropy:

$$I(\zeta;\eta) = H(\zeta) - H(\zeta/\eta) = H(\zeta) + H(\eta) - H(\zeta\vee\eta), \tag{3}$$

where $\zeta\vee\eta$ is *refinement* of ζ and η, i.e., a partition constituted by all pairwise intersections of elements of ζ and η.

In particular, $I(\zeta;\zeta) = H(\zeta)$, so the information of ζ is identified with its entropy.

Another entropy (closely related to the previous one) is *epsilon-entropy* of bounded metric spaces (Kolmogorov & Tikhomirov, 1959). It depends on a parameter $\varepsilon > 0$, and is the minimal number of balls of radius ε that cover the space. The value $\log N(\varepsilon)$, where $N(\varepsilon)$ is this number, is ε-*entropy* of the space. It can be interpreted as the minimal number of bits sufficient to ε-approximately represent any point of the space. This entropy is exploited to describe information complexity of algorithms in (Slissenko, 2011, 2012), though the obtained description is not convincing.

There are many other entropies in mathematics, e.g., Rényi entropy, graph entropy, volume entropy of a compact Riemannian manifold, von Neumann quantum entropy. We need none of them here.

Philosophy gives a wide variety of ideas how to treat the contents or semantics of information data or what is the knowledge brought by the information data (e.g., see (Barwise & Seligman, 1997; Floridi, 2013)), but these ideas are not enough quantitative to be applied directly to the analysis of information convergence of algorithms.

2. Algorithm. Trace. Examples.

An algorithm is viewed here as a set of its traces sliced by inputs of a given size. As we deal with the analysis of concrete algorithms, we do not need any general definition of algorithm in terms of sets of traces. To diminish the number of occurrences of terms "algorithm" and "function computed by the algorithm", we denote by $\mathfrak{A}$ an arbitrary algorithm of our general definitions, and by F the function it computes. The domain and range of any function f is denoted respectively $\boldsymbol{dm}(f)$ and $\boldsymbol{rn}(f)$. By default, these notations concern the inputs of some fixed size n.

2.1. Vocabulary of Algorithm. Traces.

Traces of $\mathfrak{A}$ are defined over its vocabulary. Sometimes we speak about a specification of F; this is a logical formula, also over a vocabulary, and the two vocabularies have the same input and output functions (we make precise this classification of functions just below).

A vocabulary $\mathcal{V}$ is a pair $(\mathcal{U},\mathcal{F})$, where $\mathcal{U}$ is a list of symbols for sets and $\mathcal{F}$ is a list of symbols for functions (and predicates). A symbol either has a fixed interpretation (then it is *pre-interpreted*) or not (then it is *abstract*). In the latter case its interpretation can be any, according to its type. We assume that all sets are pre-interpreted, and they are $\mathbb{N}$ (natural numbers), $\mathbb{Z}$ (integers), $\mathbb{B} =_{df} \{0,1\}$, the set $\mathbb{B}^*$ of strings over $\mathbb{B}$, an alphabet $\mathcal{A}$ with at least two letters: $\alpha =_{df} |\mathcal{A}| \geq 2$, Boolean values $\boldsymbol{Bool} = \{\text{true}, \text{false}\}$ and simple finite subsets of these sets that will be clear from the context.

Among the pre-interpreted functions there are constants of the just mentioned pre-interpreted types (e.g., concrete integers), addition and subtraction of integers, order relations over numbers, Boolean operations and some other simple basic operations that will be clear from the context.

All the pre-interpreted functions are *static*, i.e., they have fixed interpretations. The abstract functions are usually *dynamic*, i.e., they can be changed by $\mathfrak{A}$. In our case the arguments of abstract functions are natural numbers that play the role of indexes of variables in programming (e.g., as in the representation of arrays).

The dynamic functions are partitioned in *input* ones that cannot be updated by $\mathfrak{A}$ and *internal* ones that may be updated by $\mathfrak{A}$. Internal functions are in their turn partitioned into *output* functions (that contain the result) and into *proper internal* functions.

Initially all internal functions have some value, the same for all inputs, e.g., 'undefined'. Any input, that represents an argument of F, is a substructure of $\mathcal{V}$ (formally, an input is an interpretation of a fixed subset of $\mathcal{V}$). We assume that the size of inputs is defined so that it is polynomially related to its bit length under more or less straightforward coding. For example, the size of a word w over $\mathcal{A}$ is by default its length $|w|$.

A *trace* of $\mathfrak{A}$ is a sequence of operations executed by $\mathfrak{A}$ starting from the first step. This sequence, that is denoted $\boldsymbol{tr}_{\mathfrak{A}}(X)$ or $\boldsymbol{tr}(X)$ for input X, consists of *updates* $f(\eta) := \theta$, where f is an internal function, and of *guards* $\Phi(\eta_1, \dots, \eta_m)$, where Φ is a formula over $\mathcal{V}$, and η, θ, $\eta_1, \dots, \eta_m$ are terms. In our examples, Φ is equality, inequality or order relation.

We give to updates and guards the following (philosophical) interpretation: an update is a transformation of data without analysis of its meaning, but a guard is a way how $\mathfrak{A}$ extracts knowledge from the small pieces of obtained data, available only locally.

A computed function can have many components that should be output individually (this is the case of the convolution considered below). In other words, not only an input but also an output may happen to be a list of functions, an array or a more complicated structure. By default, we consider traces for inputs of size exactly n.

All the traces are assumed to be finite, its length, i.e., the computational complexity, for an input X is denoted $|\boldsymbol{tr}(X)|$.

2.2. Examples

Binary convolution.

This function is the most essential part of binary multiplication.

Input: A pair of strings of the same length $x, y \in \mathbb{B}^n$, $n \geq 1$; n is the *size* of the input. Their bits are denoted respectively $x(0), x(1), \dots, x(n-1)$ and $y(0), y(1), \dots, y(n-1)$.

Output: (we assume for simplicity that $x(i) = y(i) = 0$ for $n-1 < i \leq 2n-2$):

$$z(x, y, k) = z(k) = \sum_{i=0}^{i=k} x(i)y(k-i), \quad 0 \leq k \leq (2n-2). \tag{4}$$

These values (they are *components* of the output) are computed by the following algorithm, that we denote $\mathfrak{C}$. It uses one nullary proper function u and for each k, algorithm $\mathfrak{C}$ computes

$$u := x(0)y(k);\ u := u + x(1)y(k-1); \dots ; u := u + x(k)y(0),$$

in a loop, and assigns the last value as the output value of the kth component of the result, i.e., the value of $z(k)$ of (4). So two loop counters are used, one over k (an external loop), and another one to calculate $z(k)$ for a given k (an internal loop).

For further references we need partial results :

$$z_j(x, y, k) = \textstyle\sum_{i=0}^{i=j} x(i)y(k-i), 0 \leq k \leq (2n-2), 0 \leq j \leq k.$$

The only input datum that is analyzed from the point of view of knowledge extraction is the size of the input. No other knowledge about inputs is involved, the values of input words are processed 'blindly'. According to our philosophical interpretation the bits of the inputs are used as information but without knowledge extraction. So the main activity of $\mathfrak{C}$ is a calculation without any analysis of what is calculated.

Palindromes.

This is a trivial algorithm of recognizing the symmetry of words.

Input: A string w over alphabet $\mathcal{A}$. The size n of the input is the length of w. We assume that n is even (just for technical simplicity).

For the palindrome problem we use the following notations (for $i \leq j$):

$w^{=}(i..j) =_{df} \Big(w(i) = w(n-i+1) \wedge w(i+1) = w(n-i) \wedge \ldots \wedge w(j) = w(n-j+1)\Big)$, i.e., the segment of w in positions $[i..j]$ is symmetric to the corresponding segment in positions $[n-j+1, n-i+1]$, i.e., equal to the inverse of the word in the latter positions;

$w^{\neq}(i..j) =_{df} \neg w^{=}(i..j)$, i.e., $w^{\neq}(i..j)$ says that

$w(i) \neq w(n-i+1) \vee w(i+1) \neq w(n-i) \vee \ldots \vee w(j) \neq w(n-j+1)$.

Output: $r = 1$ if w is a palindrome, i.e., $w^{=}(1..\frac{n}{2})$, and $r = 0$ otherwise.

Below in the context of the palindrome problem we use the following notations:

$\nu =_{df} \frac{n}{2}$, $w^{=}(i) =_{df} w^{=}(i..i)$, $w^{\neq}(i) =_{df} w^{\neq}(i..i)$.

We consider a simple algorithm $\mathfrak{P}$ that uses only one proper internal function l as a loop counter. Starting from $l = 1$, algorithm $\mathfrak{P}$ checks $w(l) = w(n-l+1)$, and if this guard is false then it outputs $r := 0$, otherwise if $l = \nu$ it outputs $r := 1$, and if $l < \nu$ then it augments the loop counter $l := l + 1$ and repeats the checking of the equality of characters.

Contrary to the previous case of $\mathfrak{C}$, algorithm $\mathfrak{P}$ does not transform anything related to the contents of the input data, but mainly computes guards that in our philosophical interpretation is a knowledge extraction.

maxPS (maximal prefix-suffix).

This example is, in some way, the most instructive in our discussion of difficulties of describing the information convergence. Here again there are few calculations with the contents of the inputs but many guard computations, and for the second algorithm, that is denoted $\mathfrak{F}_1$, there are calculations related to a proper internal function φ. These calculations are based on an intensive use of guards unlike to the calculations of $\mathfrak{C}$.

The maxPS problem is simple: given a word, find the maximal (longest) prefix, different from the entire word, that is also a suffix of the word.

Input: A string w over an alphabet $\mathcal{A}$. The size n of the input is its length that we consider to be even for technical simplicity.

As above $\nu =_{df} \frac{n}{2}$.

Output: $r = \max\{k : 0 \leq k \leq (n-1) \wedge w(1..k) = w(n-k+1..n)\}$, i.e., r is the length of the longest prefix of w, different from w itself, that is also a suffix of w.

We consider two algorithms for maxPS: a straightforward one $\mathfrak{F}_0$ with complexity $\mathcal{O}(n^2)$, and another one $\mathfrak{F}_1$ with complexity $\mathcal{O}(n)$.

Notation: $\varphi(w,m)=_{df} \max\{k : 0 \leq k \leq (m-1) \wedge w(1..k) = w(m-k+1..m)\}$, here $0 \leq m \leq n$ and $\varphi(w,0)=0$.

Algorithm $\mathfrak{F}_0$ acts straightforwardly. Starting from $h = n-1$ it checks whether $w(1) = w(n-h+1), w(2) = w(n-h+2), \ldots$ in this order. If it reaches $w(h) = w(n)$ then $r := h$ otherwise if $h > 1$ algorithm goes to $h := h-1$, if not, i.e., if $h = 1$ and $w(1) \neq w(n)$ then $r := 0$.

For an input word $w_1 = aa\ldots ab = a^{n-1}b$ algorithm $\mathfrak{F}_0$ makes $\geq c \cdot n^2$ steps for some natural constant $c > 1$ that depends of the details of description of the algorithm and that is not important. Indeed, in order to find that $r \neq (n-1)$ algorithm makes $(n-1)$ comparisons $w(i) = w(i+1)$ for all $1 \leq i \leq (n-1)$. After that it makes $(n-2)$ comparisons $w(i) = w(i+2)$ and finds that $r \neq n-2$, and so on, up to the last comparison $w(1) = w(n)$ that is not true, and hence it sets $r = 0$.

One can notice that after the first $(n-1)$ comparisons there is enough knowledge to conclude that $r = 0$ but no appropriate knowledge extraction is applied by $\mathfrak{F}_0$. However, the algorithm continues to work and finally converges to the result. Due to what information is it converging? (This question is clarified in section 3).

Algorithm $\mathfrak{F}_1$ (that can be found in standard textbooks of algorithmics) recursively calculates $\varphi(w,m)$ for all m starting from $m = 1$; initially $\varphi(w,0) = 0$.

Denote by the same letter φ an internal function of $\mathfrak{F}_1$ of type $[0..n] \to [0..n-1]$ that represents $\varphi(w,m)$ as $\varphi(m)$. Its initial value is $\varphi(0) = 0$.

Denote by $\varphi^k(m)$ the kth iteration of $\varphi(m)$, $k \geq 1$: $\varphi^1(m) = \varphi(m)$ and $\varphi^{k+1}(m) = \varphi(\varphi^k(m))$. For technical simplicity set $\varphi^0(m) = -1$ for all m, and $\min\emptyset = 0$.

Suppose that $\varphi(m)$ is defined, and $m < n$. Algorithm $\mathfrak{F}_1$ computes
$\varphi(m+1)$ as $\varphi^s(m)+1$, where $s = \min\{k : w(\varphi^k(m)+1) = w(m+1)\}$.

Clearly, this computing of $\varphi^s(m)$ takes $\mathcal{O}(s)$ steps. The whole complexity of $\mathfrak{F}_1$ is linear.

We illustrate the work of $\mathfrak{F}_1$ for the following 4 inputs of length n, where $a, b \in \mathcal{A}$, and $a \neq b$.

Set $w_0 = a^n$, $w_1 = a^{n-1}b$, $w_2 = a^{i-1}ba^{n-i}\ldots b$, where $i \leq \nu$, and $w_3 = a^{i-1}ba^{n-i}\ldots b$, where $i > \nu$.

One can see that $\varphi(w_0,n) = n-1$, $\varphi(w_1,n) = 0$, $\varphi(w_2,n) = i$, $\varphi(w_3,n) = 0$. The evolution of φ for w_1 explains also the evolution of φ for the other just mentioned inputs. The evolution of $\varphi(w_1,m)$ is presented in the table below. This table consists of two unrelated subtables: the one over the first 3 lines, the other one over the remaining two lines.

position	1	2	...	$n-2$	$n-1$	n
character	a	a	...	a	a	b
φ	0	1	...	$n-3$	$n-2$	0
index k of iteration $\varphi^k(n-1)$	$n-1$	$n-2$	...	2	1	
$\varphi^k(n-1)$	0	1	...	$n-3$	$n-2$	

Here we see again that there is enough information to define the result without calculating the iterations $\varphi^k(n-1)$ because the computed properties $\varphi(n-1) = n-2$ and $w(n-1) \neq w(n-2)$ uniquely define the word w_1.

3. Information Convergence

Although the idea to use metrics to describe the information complexity of an algorithm looks attractive, it stumbles over serious difficulties. Some attempts to do it are described in my papers

(Slissenko, 2011, 2012). In these papers the metrics are defined over the set of traces of an algorithm $\mathfrak{A}$ generated by the inputs of a size n. However, this approach has grave shortcomings. The obtained evaluations of epsilon-entropy are hard to interpret. Moreover, in the spaces of traces the epsilon-entropy is always bounded by the cardinality of the domain of $\mathfrak{A}$, and the complexity of algorithm is not much related to this cardinality. In particular, an algorithm can have a high complexity over a very meager domain, and inversely, an algorithm over a big, 'fat' domain can be very simple.

These shortcomings of this particular approach do not mean that epsilon-entropy of the set of traces cannot give more productive approaches.

A more attractive idea mentioned in (Slissenko, 2011), is to find a measure of similarity in terms of some kind of basis of all traces of $\mathfrak{A}$. This idea has not been yet developed.

In the same papers there was described an approach how to measure the convergence of an algorithm towards its result in entropy terms. This approach may be seen as a *semantical* one as it is based on the analysis of the graph of F. It works for algorithms that are calculating without analyzing what is being done. Below we illustrate this approach by the example of convolution.

The semantical approach, as it is based on estimations of entropy, needs a probabilistic measure on the set of inputs. The measure is introduced following the principle of maximal uncertainty mentioned at the end of Introduction.

We assume that $\mathfrak{A}$ plays against an adversary that tries to minimize the capacity of $\mathfrak{A}$ to find the result. Thus, all outputs are equiprobable. So for a given component f of F the measure of all $f^{-1}(v)$ should be the same for all $v \in \boldsymbol{rn}(f)$.

We denote by $g[X, t]$ the latest value of internal function g in trace $\boldsymbol{tr}(X)$ not later than at instant t; notation $g[t] = u$ is explained inside the formula (5).

Suppose that $\mathfrak{A}$ has calculated $g[t] = u$ for an internal function g. What does this information give, if $\mathfrak{A}$ wishes to decide whether $f(X) = v$ or not (for some $v \in \boldsymbol{rn}(f)$)? In terms of the just introduced measure the information concerning $f(X) = v$ is given by $-\log$ of the following probability:

$$\boldsymbol{P}(f = v \,|\, g[t] = u) = \boldsymbol{P}(\{X : f(X) = v\} \,|\, \{X : g[X, t] = u\}) = \frac{\boldsymbol{P}(f = v \ \wedge \ g[t] = u)}{\boldsymbol{P}(g[t] = u)}. \tag{5}$$

One can see that this probability is expressed in terms of the graphs of the involved functions, no syntax is really used (only the names of functions appear in the expression).

Convolution (Semantical information convergence).

As it was mentioned above, for this example $\boldsymbol{P}(z(k) = v) = \frac{1}{k+1}$ for each $0 \le v \le (k+1)$.

We evaluate what formula (5) gives for $z(x, y, k) = v$ if $\mathfrak{C}$ has found $z_i(x, y, k) = u \le v$:

$$\boldsymbol{P}(z(k) = v \,|\, z_j(k) = u) = \frac{\binom{j+1}{u}\binom{k-j}{v-u}}{(k+2)\binom{k+1}{v}\frac{\binom{j+1}{u}}{(k+2)}\sum_{u \le \alpha \le (k+1)} \frac{\binom{k-j}{\alpha-u}}{\binom{k+1}{\alpha}}} =$$

$$\frac{\binom{k-j}{v-u}}{\binom{k+1}{v}\sum_{u \le \alpha \le (k+1)} \frac{\binom{k-j}{\alpha-u}}{\binom{k+1}{\alpha}}} = \frac{\binom{k-j}{v-u}}{\binom{k+1}{v}\sum_{u \le \alpha \le (k-j+u)} \frac{\binom{k-j}{\alpha-u}}{\binom{k+1}{\alpha}}} \underset{j \to k}{\longrightarrow} 1$$

here we consider the evolution of u in any trace computing $z(k)$, in this case $u = u(j) \underset{j \to k}{\longrightarrow} v$. This formula is correct for correct values of u, v, k, i, i.e., for such values that $0 \le u \le v$, $0 \le i \le k$. Otherwise, the value of probability is 0. Clearly, u goes to v when i goes to k.

However, this semantical approach may not work for algorithms whose computation essentially exploits guards, that is, knowledge extraction.

Palindromes (Information convergence.)

Here the probabilistic measure attributes $\frac{1}{2}$ to the set of palindromes as well as to the set of non-palindromes. Inside these sets the measure is uniform (though this may be not the best choice for the set of non-palindromes).

If $\mathfrak{P}$ has calculated $w^{=}(1..i)$, $1 \le i \le \nu$, then taking into account that the palindromes constitute a subset of $\{w : w^{=}(1..i)\}$, we have (here and below we omit technical details of calculations and leave only those final results that are relevant to the context):

$$\boldsymbol{P}(r = 1 \,|\, w^{=}(1..i)) = \frac{\boldsymbol{P}(w^{=}(1..\nu))}{\boldsymbol{P}(w^{=}(1..i))} =$$

$$\frac{1}{2 \cdot \Big(\boldsymbol{P}(w^{=}(1..\nu)) + \boldsymbol{P}(w^{=}(1..i) \wedge w^{\neq}(i+1..\nu))\Big)} =$$

$$\frac{1}{2 \cdot \left(\frac{1}{2} + \frac{\alpha^{i}(\alpha^{n-2i} - \alpha^{\frac{n-2i}{2}})}{2(\alpha^{n} - \alpha^{\nu})}\right)} = \frac{1}{1 + \frac{\alpha^{n-i} - \alpha^{\nu}}{\alpha^{n} - \alpha^{\nu}}} = \tag{6}$$

$$\frac{1}{1 + \frac{1}{\alpha^{i}(1 - \alpha^{-\nu})} - \frac{1}{\alpha^{\nu} - 1}} \approx \frac{1}{1 + \alpha^{-i} - \alpha^{-\nu}} \tag{7}$$

We see from the exact formula (6) or from the approximate formula (7) that the probability that the input is a palindrome goes to 1; we can even evaluate the speed of information convergence to 0 when i goes to its limit value ν if we take $-\log$ of the last expression: $c \cdot (\alpha^{-i} - \alpha^{-\nu})$, here $c = \frac{1}{\ln 2} \approx 1.44$ (this value is not important). However, this estimation is not compatible with the viewpoint that the algorithm plays against an adversary who prevents it to predict the result until the last instant.

Look at the probability to have non-palindrome (that is 1 minus the previous one):

$$\boldsymbol{P}(r = 0 \,|\, w^{=}(1..i)) = \frac{\boldsymbol{P}(w^{=}(1..i) \wedge w^{\neq}(i+1..\nu))}{\boldsymbol{P}(w^{=}(1..i))} = \frac{\frac{\alpha^{n-i} - \alpha^{\nu}}{\alpha^{n} - \alpha^{\nu}}}{1 + \frac{\alpha^{n-i} - \alpha^{\nu}}{\alpha^{n} - \alpha^{\nu}}} \approx \frac{\alpha^{-i} - \alpha^{-\nu}}{1 + \alpha^{-i} - \alpha^{-\nu}} \tag{8}$$

Here the probability that the input is a non-palindrome goes to 0 whatever expression in line (8) we use. Again it is counter-intuitive as the output can be any up to the last instant.

On the other hand, the entropy of the partition into the involved sets of inputs

$$\frac{1}{1 + A(i)} \log \frac{1}{1 + A(i)} + \frac{A(i)}{1 + A(i)} \log \frac{A(i)}{1 + A(i)}, \tag{9}$$

where $A(i) = \alpha^{-i} - \alpha^{-\nu} \underset{i \to \nu}{\longrightarrow} 0$, says that the uncertainty diminishes (goes to 0) as a result of the work of the algorithm. This is closer to our intuition. However, this formula (9) does not give a clear vision of the speed of convergence to the result.

We can look at the question of convergence differently, involving some syntax in the analysis. We follow the principle of maximal uncertainty that was mentioned above: we choose probabilities to maximize the uncertainty that the algorithm should resolve. This time we apply this principle not only to the graph of F but also to the logical formula that governs the choice of guards of $\mathfrak{A}$ when it is processing a particular input. This formula consists of an ordered set of literals, and the order is used by the algorithm we consider.

We illustrate the principle of maximal uncertainty for $\mathfrak{P}$ just below and for $\mathfrak{F}_k$ (algorithms for maxPS, $k = 0, 1$) after that.

At the beginning the adversary (who foresees the work of the algorithm) chooses with equal probability one of the following 'defining formulas' (one can see that they are conjunctions of the guards verified by $\mathfrak{P}$ for appropriate inputs):

$$\left.\begin{array}{l} \sigma_1 :\ w(1) \neq w(n) \\ \sigma_2 :\ w^{=}(1..1),\ w(2) \neq w(n-1) \\ \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \\ \sigma_\nu :\ w^{=}(1..\nu-1),\ w(\nu) \neq w(\nu+1) \\ \sigma_{\nu+1} :\ w^{=}(1..\nu)\ \ (w \text{ is palindrome}) \end{array}\right\} \tag{10}$$

We consider each passage from σ_i to σ_{i+1} as one step; technically, $\mathfrak{P}$ carries out other manipulations when moving from σ_i to σ_{i+1}, so the real number of steps of $\mathfrak{P}$ for this passage is $\mathcal{O}(1)$, but, clearly, we can ignore it.

In order to arrive at $r = 0$ the algorithm should have $\neg\sigma_1 \wedge \ldots \wedge \neg\sigma_{s-1} \wedge \sigma_s$ for some $s \leq \nu$. At each step the uncertainty should remain maximal, hence, the whole probability to arrive at $r = 0$ is $\frac{1}{2}$, and the choices of σ_s, $s \leq \nu$, among the remaining $(\nu - s + 1)$ ones are equiprobable with probability $\frac{1}{2(\nu-s+1)}$ (here $\frac{1}{2}$ comes from the just mentioned probability of $r = 0$). The probability to get $r = 1$ is also $\frac{1}{2}$. Thus, the entropy after s steps is

$$-\Big(\frac{\nu-s+1}{2(\nu-s+1)}\log\frac{1}{2(\nu-s+1)} + \frac{1}{2}\log\frac{1}{2}\Big) =$$
$$\frac{1}{2}(1+\log(\nu-s+1)) + \frac{1}{2} = 1 + \frac{1}{2}\log(\nu-s+1), \quad s \leq \nu \tag{11}$$

We see that the entropy goes down to 1 as the logarithm of the number of steps s when s approaches ν, and formula (11) gives a clearer vision of the convergence than formula (9). Formula (11) is not applicable to step $(\nu+1)$, but it is not necessary – the entropy after this step is 0. What may look intuitively unsatisfactory, is that the speed of convergence is described in terms of a logarithm of some function of s but not in terms of a linear function of s.

The latter observation suggests that *the classical entropy may not be the best way to measure uncertainty in our context.*

maxPS (Information convergence.)

First, we look at semantical convergence that, taken straightforwardly as above, does not work well for $\mathfrak{F}_k$, $k = 0, 1$. It may be useful to have an idea of the measure defined along the lines of the beginning of this section as $\boldsymbol{P}(\varphi^{-1}(v)) = \frac{1}{n}$, $0 \leq v \leq (n-1)$ (though this is not used below). The cardinality of $\varphi^{-1}(v)$ can be evaluated recursively starting with $|\varphi^{-1}(n-1)| = \alpha$.

We look at the semantical convergence taking $w = a^{n-1}b$ as input. Any of $\mathfrak{F}_0$ and $\mathfrak{F}_1$, quickly converges to the result during the first $(n-1)$ comparisons (we need no details, just note that after each comparison $w(i-1) = w(i)$ one value of φ is excluded). Indeed, when an algorithm $\mathfrak{F}_k$, arrives at $w(n-2) = w(n-1)$ and $w(n-1) \neq w(n)$, this information contains a knowledge that suffices to determine that the input is of the form $a^{n-1}b$ with $a \neq b$. Thus, according to this knowledge, the output is 0. But the algorithm, though it implicitly has this information in two local values, does not extract this knowledge about the output as it was just explained. The algorithm is proceeding differently. So though semantically the result is known, the algorithm continues to work, and during this further work *there is no semantical convergence*. However, $\mathfrak{F}_k$ converges to the result. Then what other convergence does take place?

Any $\mathfrak{F}_k$ constructs its own defining formula for the result. So we can estimate its convergence 'towards this defining formula'. For words in $z \in \varphi^{-1}(0)$ the formula used by $\mathfrak{F}_0$ is

$$\left.\begin{array}{l} \xi_{n-1}: \ z(1) \neq z(2) \vee z(2) \neq z(3) \vee \ldots \vee z(n-1) \neq z(n) \\ \xi_{n-2}: \ z(1) \neq z(3) \vee z(2) \neq z(4) \vee \ldots \vee z(n-2) \neq z(n) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \xi_2: \ z(1) \neq z(n-1) \vee z(2) \neq z(n) \\ \xi_1: \ z(1) \neq z(n) \end{array}\right\} \tag{12}$$

For the general case $z \in \varphi^{-1}(v)$ we have

$$\left.\begin{array}{l} \xi_{n-1}: \ z(1) \neq z(2) \vee z(2) \neq z(3) \vee \ldots \vee z(n-1) \neq z(n) \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ \xi_{v+1}: \ z(1) \neq z(v+1) \vee z(2) \neq z(v+2) \vee \ldots \vee z(n-v) \neq z(n) \\ \overline{\xi}_v (\Leftrightarrow \neg\xi_v): \ z(1..v) = z(n-v+1..n) \end{array}\right\} \tag{13}$$

In order to find the value of $\varphi(z)$ for an input z (suppose $\varphi(z) = v$) algorithm $\mathfrak{F}_0$ constructs a sequence of inequalities, at least one from each ξ_i, $(v+1) \leq i \leq (n-1)$, and the equalities $\overline{\xi}_v$.

Look at the convergence of $\mathfrak{F}_k$ towards the defining formula for input $w = a^{n-1}b$ from the viewpoint of the principle of maximal uncertainty. We try to choose a model that gives an intuitively acceptable explanation of the convergence (other models are also imaginable).

Any $\mathfrak{F}_k$, $k = 0, 1$, starts its work with the verification of ξ_{n-1} from left to right. As all the values of φ are equiprobable (then the uncertainty of final result is maximal), the probability of $\neg\xi_{n-1}$ is $\frac{1}{n}$, and that of ξ_{n-1} is $(1 - \frac{1}{n}) = \frac{n-1}{n}$. For the same reason of maximizing the uncertainty, the probability to have $z(1) \neq z(2)$ is $\frac{n-1}{n(n-1)} = \frac{1}{n}$. If $z(1) = z(2)$ then probability to have $z(2) \neq z(3)$ becomes slightly bigger: $\frac{n-1}{n(n-2)} = \frac{1}{n} + \frac{1}{n(n-2)}$. And so on: if $\bigwedge_{i \leq p} z(i-1) = z(i)$ then the probability to have $z(j) \neq z(j+1)$ for $j \geq p$ is $\frac{n-1}{n(n-p)} = \frac{1}{n} + \frac{p-1}{n(n-p)}$.

After ξ_{n-1} has been established, the value $\varphi(z) = n-1$ is excluded, the number of remaining values $\varphi(z)$ becomes $(n-1)$, and the probability of each becomes $\frac{1}{n-1}$.

After that $\mathfrak{F}_0$ and $\mathfrak{F}_1$ work differently.

Algorithm $\mathfrak{F}_0$ consecutively checks all ξ_v, starting with ξ_{n-1} . During this processing, after $\bigwedge_{i \leq p} z(i-1) = z(i+n-s)$ has been established, the probability to have $z(j) \neq z(j+n-s)$ for $j \geq p$ is $\dfrac{s-1}{s(s-p)}$, and there are $(s-p)$ such possibilities.

Thus the entropy of this distribution is

$$\begin{aligned} -\left((s-p) \cdot \frac{s-1}{s(s-p)} \log \frac{s-1}{s(s-p)} + \frac{1}{s} \log \frac{1}{s}\right) &= \\ -\left(\left(1 - \frac{1}{s}\right) \log \frac{1}{s-p} + \left(1 - \frac{1}{s}\right) \log \left(1 - \frac{1}{s}\right) + \frac{1}{s} \log \frac{1}{s}\right) &= \\ \log s - \frac{\log s}{s} + \left(1 - \frac{1}{s}\right) \log \left(1 - \frac{p}{s}\right) - \left(1 - \frac{1}{s}\right) \log \left(1 - \frac{1}{s}\right) + \frac{\log s}{s} &= \\ \log s + \left(1 - \frac{1}{s}\right) \log \left(1 - \frac{p}{s}\right) - \left(1 - \frac{1}{s}\right) \log \left(1 - \frac{1}{s}\right) &= \\ \log s - \left(1 - \frac{1}{s}\right) \log \frac{s-1}{s-p} & \end{aligned} \tag{14}$$

Here $p \to (s-1)$ and $s \to 1$ give the speed of diminishing of the uncertainty in terms of this evolution of s and p. The convergence by p is very slow and 'explains' the complexity $\mathcal{O}(n^2)$ of $\mathfrak{F}_0$.

The convergence of $\mathfrak{F}_1$ is the same as that of $\mathfrak{F}_0$ only when $\mathfrak{F}_1$ processes ξ_{n-1}. After that there is no p, algorithm $\mathfrak{F}_1$ excludes one value of φ at each step (that consists of the calculation of $\varphi^{(n-s+1)}(n-1)$ from $\varphi^{(n-s)}(n-1)$ and of the comparison of the appropriate characters), and the uncertainty goes down only due to s, thus much faster. We omit technical details.

Conclusion

The examples above illustrate some ideas that may contribute to the development of a general theory of information convergence of algorithms. The same examples reveal some, but not all, difficulties on this way. Algorithms working with more complicated data structures, e.g., wave algorithms for shortest paths in weighted graphs, show that the situation may be more intricate than in the examples above (for definiteness we take Dijkstra algorithm when speaking about the shortest path problem). In those examples the probabilistic measure used in the evaluation of the appropriate entropy is based, first, on the partition of the domain of the computed function F (defined at the beginning of section 3), that does not depend on the algorithm $\mathfrak{A}$ under study, and, second, on the guards checked by $\mathfrak{A}$. The partition of the domain of F is much more complicated for the shortest path problem than for any of the examples considered in the paper. The analysis of the guards of $\mathfrak{A}$ is, as well, more difficult.

Our analysis of guards deals with ordered conjunctions of the guards that are evaluated by $\mathfrak{A}$ one by one in the respective order. These guards are specific for each input. In our examples this dependence is simple, and not too diverse. The conjunctions of guards are conjunctions of equalities and inequalities of characters of inputs in easily describable positions. For convolution and palindromes these are respectively formulas (4) for $z(x,y,k)$ and (10). The both are easily related to the specification of the problem in logical terms, and thus, to the graph of the computed function. All this facilitates the assignment of probabilities to the events of a trace, and thus, facilitates the evaluation of the entropy. Formula (13) of maxPS is slightly less clearly related to the problem specification, and thus to the graph of F but however, is not too complicated to be analyzed.

If we look at the algorithm for the shortest paths problem we see that the syntax to be analyzed is not directly related to the graph of the function (nor to the specification), and what is worse, is rather sensitive to particular inputs, and involves internal functions of the algorithm (namely, the length of the current partial paths). The guards calculated by the algorithm, guide a non-trivial choice of edges and vertices that can be very dependent on the input being processed. The uncertainty to maximize is in the choice of the next edge to proceed, and this is very specific to the algorithm and to its particular input. This shows that the general case needs more subtle analysis than the case of our examples.

There may be other approaches to the information convergence. Concepts developed in philosophical literature may prove to be useful guides for this. We mention, following (section 1.5, Barwise & Seligman, 1997), some definitions of what is "$F(r)$ carries information that $G(s)$ to a person with a prior knowledge $\mathcal{K}$". In these definitions F and G are properties, and $\mathcal{K}$ remains an abstract parameter. The formulations below may need explanation (see (section 1.5, Barwise & Seligman, 1997)), but however, they give a good amount of intuition without additional comments.

– Dretske's Information Content: the conditional probability of $G(s)$ given $F(r)$ (and $\mathcal{K}$) is 1 (and less than 1 given $\mathcal{K}$ alone).

– Possible Worlds Information Content: in all possible worlds compatible with $\mathcal{K}$ and in which $F(r)$, there takes place $G(s)$ (and there is at least one possible world compatible with $\mathcal{K}$ in which $\neg G(s)$).

– State-space Information Content: in every state compatible with $\mathcal{K}$ in which $F(r)$, there takes place $G(s)$ (and there is at least one state compatible with $\mathcal{K}$ in which $\neg G(s)$).

– Inferential Information Content: the person could legitimately infer that $G(s)$ from $F(r)$ together with $\mathcal{K}$ (but could not from $\mathcal{K}$ alone).

In some way, *Dretske's Information Content* has a flavor of our semantical approach, and *Possible Worlds Information Content* has a flavor of the mixed semantical/syntactical approach. As for *Inferential Information Content*, it may be interesting to note that inference system can be used to introduce a metric and thus, epsilon-entropy in the set of formulas that represent the knowledge and its evolution, see (Slissenko, 1991).

The quantitative notion of uncertainty as entropy remains too narrow (as well as the quantitative notion of information). More adequate concepts may be necessary in order to advance the analysis of information convergence of algorithms.

Much more difficult and much more interesting problem is to understand what is *algorithmic information content* of an algorithmic (i.e., solvable by an algorithm) problem. An abstract description of algorithm as a set of traces is indispensable to embark on the study of this question. This can be done, but this is only a minor first step necessary to launch an attack on the problem.

References

Adriaans, P. (2013). Information. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Fall 2013 ed.).

Barwise, J., & Seligman, J. (1997). *Information flow: The logic of distributed systems.* Cambridge University Press, New York.

Floridi, L. (2013). Semantic conceptions of information. In E. N. Zalta (Ed.), *The stanford encyclopedia of philosophy* (Spring 2013 ed.).

Kolmogorov, A. N., & Tikhomirov, V. M. (1959). ε-entropy and ε-capacity of sets in function spaces. *Uspekhi Mat. Nauk*, *14*(2(86)), 3–86. (In Russian. English translation in: Selected Works of A.N. Kolmogorov: Volume III: Information Theory and the Theory of Algorithms (Mathematics and its Applications), Kluwer Academic Publishers, 1992.)

Martin, N. F. G., & England, J. W. (1981). *Mathematical theory of entropy* (J. K. Brooks, Ed.). Addison-Wesley.

Slissenko, A. (1991). On measures of information quality of knowledge processing systems. *Information Sciences: An International Journal*, *57–58*, 389–402.

Slissenko, A. (2011). On entropy in computations. In *Proc. of the 8th intern. conf. on computer science and information technology (csit'2011), september 26–30, 2011, yerevan, armenia. organized by national academy of science of armenia* (pp. 25–30). National Academy of Science of Armenia. (ISBN 978-5-8080-0797-0)

Slissenko, A. (2012). Towards analysis of information structure of computations. In *Proc. of international conf. philosophy, mathematics, linguistics: Aspects of interaction (phml'2012) may 22–25, 2012, saint petersburg, russia* (pp. 182–192). International Mathematical Euler Institute, Russian Academy of Sciences, Saint Petersburg, Russia. (ISBN 978-5-9651-0642-4)