in: Hanson Philip (ed), Information, Language & Cognition, OUP, 1990. pp 340-365

Kirsh, D. When is Information Explicitly Represented? The Vancouver Studies in Cognitive Science. (1990) pp. 340-365. Re-issued Oxford University Press. 1992.

A.e

CHAPTER TWELVE

When Is Information Explicitly Represented?

David Kirsh*

INTRODUCTION

Computation is a process of making *explicit*, information that was *im*plicit. In computing 5 as the solution to $\sqrt[3]{125}$, for example, we move from a description that is not explicitly about 5 to one that is. We are drawing out numerical consequences of the description $\sqrt[3]{125}$. We are extracting information implicit in the problem statement. Can we precisely state the difference between information that is implicit in a state, structure or process and information that is explicit?

Most discussions of implicit/explicit belief, knowledge, and representation confidently assume that we know what it is for information to be explicitly encoded; the problematic notion — if any notion is problematic at all — is implicit information.[1] What inspires this confidence is a particular vision of computation.

Let us suppose that to understand a computation it is necessary to track the trajectory of informational states the computing system follows as it winds its way to an explicit answer. If a computer is seen as a mechanism which applies rules to syntactically structured representations, it is natural to view explicit information as an encoding of information in syntactic structures that are interpretable according to a well-behaved theory of content, such as a truth theory. We can then point to a syntactic structure in the system and say 'that form encodes this content.' This, I believe, is our underlying idea behind our intuitions about explicit encodings.

As different kinds of computational mechanisms are discovered and explored — PDP systems, massive cellular automata, analogue relaxation systems — it is becoming increasingly difficult, however, to track the trajectory of informational states these mechanisms generate. There is no doubt that we must find some method of tracking them; otherwise there is no reason to think of them as more than complex causal systems.[2] But it is at present an open question whether the model of rules operating on explicit representations is a perspicuous model of their style of computation.

Once the lid has been raised on new styles of computation we are forced to re-examine our uncritical intuitions about basic notions. We already know that there are many ways information can be implicit in a state, structure or process, and that we are largely ignorant of the full variety of ways that information can be built into architecture, internal dynamics, and environment-system interaction. It seems to follow that one reason it is hard for us to track informational trajectories is that we don't yet know how to determine what information is in a system.

The same problem arises for computational systems that are familiar: we do not know how to determine unambiguously exactly what information is encoded in a system, even explicitly.

For instance, what information is contained in a system that has lost its pointers to one of its data sets? The data is still recorded in the system, in the sense that if the system could regain its obliterated pointers, the full data set could be retrieved. But *ex hypothesis* those pointers are unrecoverable. The states are unusable. Does the system still explicitly encode the data even though they are absolutely inaccessible?

Suppose the pointers are not simple address locations, themselves stored in a look-up table, but rather are calculated by solving a complex function. Or suppose the data is not found in a single location but spills over to many cell locations connected by pointers of the most complex sort. Is the data explicitly encoded?

Again, suppose a set of axioms is represented in a language as expressive as first order predicate calculus. Is the whole deductive closure of the set implicit? Even if that set is infinite, or would require exponential computation to derive its members? What about the $2^{100^{100}}$ digit of π? Is that implicit in the state of an arithmetical engine?

Or suppose that a system has highly ambiguous encodings and must deliberate in order to choose the right interpretation for a given word. What information is encoded explicitly? Is any? Must explicit encodings be non-ambiguous?

That such questions arise is proof that we have unsettled intuitions about the meaning of explicit and implicit information even in familiar programmable symbol manipulating systems. Computer and cognitive scientists talk as if they have a precise idea of these concepts. But they do not.

My intent in what follows is to articulate a particular conception of explicit information that at least may serve as a stable basis for subsequent inquiries into the meaning of implicit information. When I began this inquiry I too assumed that our notion of explicit was unproblematic. No longer. The paper is divided into three sections.

In the first, I show, in greater detail, why the notions of explicit and implicit need elucidation. Our intuitions are not consistent; nor is there any settled view how to resolve them. Yet the concepts are important for both computer and cognitive scientists.

In the second section, I explore efforts to identify explicit information with syntactically and semantically well-defined representations. It is hard to imagine any more natural image of explicit encodings of information than sentences in a natural language. But as we shall see it is not enough that information be present in an encoding; it must be usably present, it must be 'instantly' accessible. This condition of use places a heavy constraint on what sorts of representations can encode information explicitly.

In a brief third section, I mention some of the implications of my view. My approach throughout will be informal.

Our Intuitions About Explicitness Are Inconsistent

Perhaps the simplest and most persistent intuition we have about what explicit means is that information is explicit if it is there, for all to see, much like an unambiguous word in a book. This image of words in a text has four properties which it is tempting to ascribe to explicit representations generally:

(1) locality: they are visible structures with a definite location;
(2) movability: no matter where in a book a word is to be found, or where in a library the book is stored, that word retains its meaning and retains its explicitness;
(3) meaning: words have a definite informational content;
(4) availability: the informational content of a word is directly available to the system reading it; no elaborate translation or interpretation process is necessary to extract the information it represents.

On the surface, these four properties seem to explain some obvious facts. For instance, we believe that the number 3 is explicitly represented as being *in* the set A: {1,5,3,7,4,4} because the information that 3 is a member of the set is *on the surface* of the data structure. The meaning of the numeral '3' is readily understood by any numerically literate agent, so its informational content is directly available. It is not

so context sensitive that the agent must read the entire data structure or the entire contents of current memory to determine that meaning. And it has a specific location in the data structure. We can point to what in the data structure explicitly carries meaning.

By contrast, if we say that an element is a member of a set iff it satisfies a given list of properties, say $\{x \mid x \text{ is an even integer and } 0 \leq x \leq 9\}$ we designate a unique set but in a manner which requires computation on the part of the user. The elements cannot be just read off. It is true that we are stating the property list explicitly as opposed to using the elliptical . . . notation to denote the elements, as in {0,2,4, . . . }, which specifies the properties of the elements implicitly. But both specifications fail to present the elements in a manner that can be directly read off. They do not explicitly encode the elements of the set.

The same difference holds between a table of trigonometric functions where there is a separate entry for each value of sin $n°$, and Euler's equation $e^{ix} = cosx + i \sin x$ for generating trigonometric values. Euler's equation is a *compact* way of describing a trigonometric table. But it does not explicitly represent the table. Taken to a first approximation, and restricted within certain bounds, Euler's equation offers the same information as the table. Yet it is in a different form. The informational content of the table is not directly available in the equation; an elaborate reasoning process is necessary to extract the information explicitly contained in the table.

Another way a representation can carry information inexplicitly is by display. For instance, the number of elements in A: {1,5,3,7,4,4} – A's cardinality — is not explicitly represented by A, even though each digit in A is explicit and ready to be counted. *A displays* its cardinality; it does not explicitly represent or encode it.

Let us say that information is displayed if there is a process in a system which can, in short order, *extract* that information, while it is explicitly represented if there is a process in the system which can *immediately grasp* the information. Information that is displayed lies just beneath the surface. Information that is explicitly represented lies on the surface.

The trouble with using immediate graspability, or better *immediate readability* as the mark of explicitness is that we run into problems as soon as we ask whether to count accessing time as part of the reading process. Are the elements in large sets immediately readable? Suppose we have a matrix 10,000 by 10,000 and we want to know the identity of the element in position (6754,9629). Even if each position in the matrix is marked by two numbers representing row and column, we shall have to expend some computational energy in locating the right position and identifying the symbol found there. The task of finding a position seems no different in principle from determining cardinality, both involve counting, or some other numerical operation. Both require computation. But then if we deny that cardinality is explicitly encoded because it can be determined only through computation, shall we not also have to deny that the value at location (*i,j*) is explicit?

The point at issue here is whether symbols which are *on the surface* in a structural sense may be below the surface in a process sense. I believe they can, and that this difference between structural immediacy and process immediacy lies at the heart of confusions about explicitness.

From a process perspective information is explicit only when it is *ready to be used*. No computation is necessary to bring the content into a usable form. From a structural perspective information is explicit when it has a definite location and a definite meaning. The confusion arises when a representation that seems to be in a usable form when viewed structurally turns out to be in a non-immediately usable form procedurally.

For instance, imagine a reader who wishes to use the suggestions in a book to help him solve a particular engineering problem. The suggestions are there, in a phrase or a line somewhere, but if the book has no index or no obvious ordering, the reader will have to scan an arbitrary amount of the book to find the information he needs. Should we say that the sought-after information is explicit for that reader? Relative to his goal of problem-solving, an indexless book is an inefficient representation of the information he needs. It fails to record the information in an easy to use form.

It will no doubt be objected that there is an important difference between finding information and using it once found. Most everyone will agree that if information is encrypted in a baroque code requiring lengthy decryption before being comprehensible then that information is not explicit. Encrypted information requires preprocessing. The information is present, in some sense, but not present in a usable enough form to be deemed explicit.

But in the case of our imagined engineer, there is no question that once the representation is actually retrieved it is easy to read. The question is whether the accessing process should be viewed as part of the representation's readableness. Is there a relevant difference between spending time and effort to locate information, and spending time and effort to decrypt? Both retrieval and decryption are algorithmic processes; both involve some form of pattern matching or network following.

My own view is that there is not a relevant difference. Explicitness is tied to usability. And usability implies a match up between the procedures available to the agent and the forms the content is encoded in. Granted, these are matters over which we have no fixed intuitions. But we have biases. Are words that are hidden in a tangle of other words any different than encryptions? A standard method of passing secret information is to send a book to one's ally with the unwritten understanding that message words are found in certain spots. Was the secret message encrypted? The question is open to dispute. Suppose the reader must *deliberate* to determine which passages in the book are the relevant ones. Does the book explicitly encode the information he needs?

From a purely computational standpoint there is no fundamental difference between spending time and cycles in finding a datum in space (memory) and spending a similar measure of time and cycles computing that datum in time. It may seem that there is a principled difference here, just as it seems that there is a principled difference between space and time. But we have learned otherwise. Accordingly, just taking computational effort as the measure of explicitness, there is no way of choosing whether to represent a given block of information by a powerful procedure plus-limited data or by a weak procedure plus exhaustive data. Either form may be able to provide the information we want when we want it and in the form we want it. Accordingly, out structural notion of explicitness may run at odds with our procedural notion. Despite out intuitions about what is on the surface we cannot decide what is explicit without knowing in detail how a system works.

Our Intuitions About Implicitness Are Inconsistent

Our intuitions are even more unsettled concerning *implicit* information. It is natural to suppose that information is implicit if it is *mediately* readable; the information is structurally hidden and/or procedurally distant but nonetheless *recoverable* by additional processing. It can be made explicit. Thus, it is often thought that the hallmark of implicit information is that it is not explicit but could be made explicit.

To take the canonical example, in formal logic it is generally assumed that formulas which are not part of a given axiom set are implicit if they lie anywhere in the set's deductive closure. Structurally they are absent but procedurally they are recoverable. Given enough processing they can be brought to the surface and represented explicitly.

This definition, however, runs into problems as soon as we try to say what 'in principle, recoverable' means. Returning to deduction, shall we say that certain formulas are implicit regardless of how much effort is required to recover those formulas? Our intuitions are not decisive here. For any set of premisses, there are going to be certain theorems that are easy to prove — nearby in lemma space — and certain others that are computationally distant.[3] Are all these theorems equally implicit? Perhaps implicitness is a matter of degree? But in that case what shall we say about theorems that are infinitely distant, or infinitely hard to reach? And what shall we say about as yet unproven theorems? Is Fermat's last theorem implicit in Peano's axioms? We know that most interesting representational systems are incomplete. It is possible, then, that Fermat's last theorem is neither provably true nor provably false. Assuming that it is true but not provable is it implicit? On one account it is not, for it does not lie in the deductive closure of the axioms. Yet if the theorem is true (and constructivists are right), then there must exist some non-deductive processes that can extract that information from the axiom set. Shall we say that a given datum of information is implicit in a representation only relative to a set of operations?

To press the point, consider a system able to discover generalizations. For such a system Euler's equation might be discovered by reasoning about a trigonometric table. Euler's equation is potentially implicit in a trigonometric table for that system. Yet whether we think the equation is actually implicit depends on how much other knowledge we believe is necessary to make the discovery. A system which can draw generalizations has a *chance* at discovering Euler's equation. But such a discovery would be remarkable. For one thing, the equation contains more information than the table itself — it applies to any real value of x — so it represents a powerful abstraction. The generalization is more than a simple interpolation of the data. It generalizes to new entities. No one knows what is required for such abstractions. Often they are inspired guesses. This is particularly true where the generalization refers to a fundamental concept, such as e, which is not in the descriptive language of the data. e is like a theoretical entity: supported by observations but not reducible to them. How these new concepts are discovered depends on so many factors that it is impossible, in general, to determine whether a given agent will ever stumble on the correct generalization. But then should we allow that generalizations are implicit in data?

Given these difficulties it is hard to make precise a notion of recoverability which can serve to demarcate the realm of the implicit accurately. I still think it is hard to imagine a more natural criterion of implicitnes that has any chance of working than *that which is not explicit but which could be made so*. But not everyone would agree.

For example, it has become popular in some circles[4] to call information implicit if it is latent in a system even if it is unrecoverable. It is well known that a computation can often be made more efficient by exploiting regularities rather than by explicitly representing them. These regularities are really assumptions about the environment, or about the interaction of the agent with the environment. For instance, a vision module designed to extract a 3-D shape from two stereoscopic images works rapidly if it is equipped with an algorithm which differentiates. Such an algorithm will work if the assumption about the world — in this case, that objects change in shape smoothly and continuously — is true.

Smoothness is a condition of the world that justifies differentiation. It is a *success condition* determining whether the algorithm will work. If the condition is false the algorithm will fail to compute a correct answer. A designer who wishes to determine the algorithm's reliability will need to know how often the assumption is correct. For the truth of the assumption is the theoretical justification of the algorithm. But shall we say that the algorithm implicitly represents the assumption? Or that information about the visual world is implicitly encoded in certain of the states, structures or processes of the visual system? Clearly, this assumption is not recoverable by the system itself because the vocabulary of early vision does not include terms such as smoothness. We find expressions of values for wavelength, physical intensity, zero-crossings, surface contours, depth measures, and so on. But nowhere in this vocabularly does a term for smoothness appear. Nonetheless, it has become fashionable to speak of the system as having an implicit theory of the world. Some would argue that the information is causally effective. Is this just sloppy language?

To take another case, some robots currently under research navigate without maps. Such systems are equipped with a compass, with knowledge of their orientation with respect to an origin, and suitable instructions to find their way from any point in a maze to any other. These robots explicitly represent information of the form *if at position A then to get to B orient* 90° *go 10 steps, turn* 120° *then go 15 steps*. It is easy to prove that the total information contained in such instruction sets is sufficient to define a structural map giving the position of all points and identifying all open corridors. A structural map is, in principle, *recoverable* from the instruction set, though not recoverable by the system itself unless it has certain analytic skills.

Should we say that information about structural relations is implicit in the instruction set? By our condition of explicit recovery we must not. According to the condition of recoverability, a system does not encode information implicitly unless it can recover that information and explicitly encode it. Because the robot lacks the ability to translate from its instructions to structural representations of its environment we are obliged to say the instructions do not contain structural information implicitly. Yet what shall we make of the intuition that it is because the instructions *do* contain structural information implicitly that when they are interpreted correctly they generate the right behaviour? If the instruction set did not conform to the structure of the world what could explain its success? *Prima facie*, the instructions succeed because they implicitly encode information about the structure of their environment. They contain structural information.

What these examples show, I believe, is that we have not yet any settled view about our intuitions about explicit and implicit information. On the one hand, it is reasonable to require that information be *actually* recoverable to be implicit. Yet, on the other, it is reasonable to grant that information can be embedded in a system so that it is causally efficacious despite being unrecoverable. Recoverability is contingent on a host of other arguably incidental processes. This, at any rate, is a position some would like to defend.

Why It Matters Whether Our Intuitions Are Unsettled

Such inconsistencies would be unproblematic if terms like explicit and implicit never appear in psychological and philosophical theories. But they do.

Fodor,[5] for instance, maintains that mental states are functional relations to explicit representations. To know or believe a certain fact is to be in a certain computational relation to a representation which explicitly encodes information about that fact. Explicitness is important to Fodor. Yet he never states what explicitness amounts to short of saying it must satisfy some ill-defined formality condition. This leaves us groping for a workable criterion of formal. We do not know, for example, whether cardinality when displayed rather than represented directly is a formal property. The same applies to the relation of having a location in a matrix, or to the relation of being connected to, or being beside. Some of these properties are usually represented, others are displayed. Are they formal properties? How we answer these questions has deep consequences. For, according to Fodor, it dete.mines what shall count as an episode in our mental life.

Gibson and his followers, too, make strong claims about implicit information. They maintain that information about shape and distance, for example, is implicit in the ambient flux of light energy striking our retinas. The visual system, we are told, does not explicitly represent edges etc., then construct further representations of shape. It picks up the invariants implicit in visual energy fields and puts that information to use in controlling behaviour "directly," without ever representing it. Because such information is put directly to use, the visual system has no facility for explicitly naming invariants; it never represents them explicitly. Visual invariants in Gibson's sense, fail to be implicit by our condition of recoverability. For by that condition a system can implicitly encode information only if it can, in principle, explicitly encode it as well, since to recover information it must be possible to represent that information explicitly. How shall we interpret Gibson's remarks? Can information about visual invariants be implicit in light energy?

More recently, Perry, Smith, and Rosenschein[6] have contended that information can be implicit in a system because that system is embedded in a particular environment. A system well-adapted to its environment contains information about that environment and its momentary relations to that environment, even though that information is built into the design of the system and so is in principle inaccessible. On their account, information need not be amenable to eventual explicit representation to be implicit. The information we decide is present in a sytem is not identical to the sum of information that is explicit plus the information that is recoverable. Once again information can be implicit but unrecoverable. Yet again what are we to make of these claims?

The upshot, it seems to me, is that we know somewhat less about information processing than we suppose. Information processing has always been understood as a process of transforming representations — transforming explicit representations. But owing to the variety of physical mechanisms that are often interpreted to be computing functions, this view is no longer universally held. This does not mean that information processing is *not* explicit representation processing. But until we have an adequate theory of the relation of implicit to explicit information we cannot decide the issue.

Towards a Theory of Explicitness

Because implicit is defined largely negatively as information that is present but not explicitly encoded, any inquiry into implicit information must presuppose a theory of explicit information. If I am right, however, our structural and procedural intuitions about explicitness are so confused that there is no easy theory to offer. We may offer a stipulative theory but it will never satisfy all our intuitions.

I shall argue that the source of confusion lies in the bewitching image of a word printed on a page. Words on a printed page have four properties that make them an attractive model of explicitness: locality, movability, meaningfulness, and immediate readability. When we look closely at each attribute we find each is, in some manner, misleading.

In the remainder of this paper, I will discuss each attribute in an effort to separate truth from fantasy. My own positive account emerges along the way.

Four Conditions on Explicitness

Locality. It is a fact of English and all other natural languages that words are represented by written tokens that are spatially compact and readily separable from their spatial neighbours. It is tempting to suppose that all explicit encodings of information share this property. Explicit information, after all, is encoded in codings. These codings must present the information in a modular, readily surveyable manner; otherwise they could not present the information in a ready to use form.

If locality were true, however, it would eliminate, in one stroke, the possibility of distributed connectionist systems ever having explicit representations. This seems to me a good reason to deny it. The kernal of truth in the locality condition is that a word, however it be represented, must be a *determinate* something. It must have identity conditions. It is pointless talking of a state, structure or process encoding information explicitly unless we can be precise about which state, structure or process does the encoding. But it certainly does not follow that these identity conditions mandate spatial isolation. They don't. They require, of course, that the system using the representation must have operators which can recognize those representations. Humans find spatial boundaries especially easy to use for individuation. But there is no *a priori* reason why symbols must be spatially separate. To demand spatial separation places an unmotivated restriction on the range of recognition devices that non-humans might use in reading and communicating.

For example, we can readily imagine a system which encodes information in the wavelengths of the visible spectrum. Since many wavelengths and intensities can be superposed on the same spatial region but later filtered out, one colour, such as a shade of white or pink, can actually explicitly encode many different pieces of information. A system appropriately endowed with colour filters can read off the information immediately. It "sees" separate colour tokens and reads them directly. To anyone without the filters, however, the information is invisible.

The same principle applies to information encoded in scatter diagrams. We can imagine a fax machine, for instance, which distributes information like buckshot sprayed over a page. Several such pages could be superposed. To a system appropriately set up, each page is separable without loss. But again, only to a system with the appropriate operators to read the buckshot distribution. Humans would find such distributions unreadable because they lack edges and simple spatial forms. The problem would be compounded by superposition. Yet what humans are able to see is irrelevant. There are many codes we cannot read unaided.

The only constraint on explicitness that locality imposes, then, has little to do with spatial forms, spatial cohesiveness or spatial size. It is about identity conditions: a representation that explicitly encodes information must be made up of tokens that are readily separable from their surroundings. This separation process may vary arbitrarily from system to system. To be sure, there are limits. A system must be able to identify tokens quickly without engaging in substantial computation. Later I will state more precisely what this means. But for now the point is that locality does not mean spatial isolation. It means separability by the host system.

This then is the first condition on explicit encoding of information:

> Condition (1): The states, structures, or processes — henceforth symbols — which explicitly encode information must be easily separable from each other.

Movability. The ideal of *operational* identity conditions also lies at the bottom of the movability condition. In its simplest form the movability condition states that a word can occur in more than one location and still carry its meaning. That is, the identity of a word is largely independent of context.

There is a profound justification for this idea. If we grant that there is no fundamental difference between transmitting information across space, and transmitting it across time, then transmission across time is storage, while transmission across space is spatial communication. Words serve as compact vehicles for meaning. They are the carriers of information. It is natural, then, to assume that we can use them both to store and to send information.

This view leads quite naturally to the idea that a word retains its meaning whether on page 1 or page 601. In principle, it could be sent from one page to the other. Either in token or in type. One consequence of this view, however, is that context is largely irrelevant to word content. This follows because if the information content of a word changed once it was transmitted we would have no way of reliably sending information. The very idea of a word is of a physical vehicle that holds its meaning across situations. But then the identity of a word must be largely independent of where and when the word appears in a system.

Now symbols which are totally mobile and which retain their identity whatever their context lie at one extreme of a continuum of symbol systems. Such symbols can never by polysemous; they can never have indexical elements; and they can never be read differently by different operators. In short, an information processing system using totally mobile symbols must be uniform throughout: there can be no regions where the symbol is interpreted differently, and no states which the system can enter into which cause exactly similar symbols to be read differently.

To see just how restrictive this constraint is I shall briefly examine several representational languages. We can then test the reasonableness of movability as a condition on explicit encoding. For if we believe that we *can* encode information explicitly in a language that violates the movability condition we have grounds for rejecting movability as just construed as a condition on explicit encoding.

Which languages satisfy movability? By a language I mean any set of individuatable states, structures or processes which can be paired with meanings. In the simplest cases, the theory which specifies a language consists of two components: a notational component and a meaning component.

The notational component tells us what to accept as allowable variation in the structure of a token. It is the theory of symbol separation. So, for example, although my writing changes from page to page as I change my posture, a good notational component would specify enough variability in tokens to cluster my written words into correct equivalence classes. It gives the identity conditions of atomic symbols.

The meaning component tells us what information is carried by each symbol. So, for example, a meaning theory for the symbols on a map will tell us that '•' means *cities with populations in excess of 50,000.*

Now just how restrictive is a language that allows no ambiguity whatsoever? Restrictive, very restrictive. Such systems can have no syntax, even a simple syntax. For instance, simple languages such as Arabic notation for numbers will have too complex a structure. Perfect notational mobility implies that the '5' in 105 and the '5' in 501 must carry the same information. It implies that identical notational elements encode identical information whatever their context. But of course the meaning of '5' changes with position. In 105, '5' means units, in 501, '5' means 500.

In order to capture this variability of meaning with position, we need to introduce a syntax. The point of syntax is to allow us to determine how to adapt the information content of a symbol to its position. For languages with syntax, the meaning component must factor in the contribution to meaning which the word's syntactic role makes. Since most languages do have syntax, a theory of language will usually contain, in addition to its notational and meaning components, a third component — a syntactic component — which describes the syntactic role symbols play when combined into compound structures.

In the case of Arabic notation the contribution to meaning made by position is trivial. Defining position as location in a string read from right to left, we then interpret '5' as follows:

'5' in position i means $5 \times 10^{i-1}$

The language for counting based on Arabic numerals has a very simple syntax. Yet even this syntax, we have seen, violates the movability condition. I think that few of us would deny that Arabic numerals represent numbers explicitly. So movability in its extreme form is too restrictive. But there are many other natural languages in which position is not all that can affect meaning. The symbols which come before or after may matter. Can these encode information explicitly?

For instance, a standard trick for extending an instruction set beyond the limits set by the individual keys on a keyboard is to use some characters as *switches*. A control character, for example, allows any character that follows it to be interpreted as a command rather than as a typed letter. *A* in the context of *Control-a* does not have its normal meaning. The control character switches *a*.

To accomodate switches we need to increase the complexity of the syntax of languages discussed so far. Up until now a syntactically primitive symbol has had no apparent structure. Our notational theory told us exactly what symbols were syntactically atomic. Now, however, we must treat certain *strings* of characters as syntactically simple. They are to be accorded the same syntactic role as atomic symbols despite their apparent molecular structure. This will have no real effect on our meaning theory. For once a language has syntax, its meaning rules are defined over syntactic elements. Thus, in our meaning theory we will find axioms such as:

'Control-a' means *move cursor to line's beginning*
'Control-e' means *move cursor to line's end*

These revisions extend the representational power of a language by adding more primitive symbols to its list of meaningful expressions. Because these syntactically primitive symbols are not notationally primitive, however, the host system has a slightly more complex recognition task, for it must first recognize the notation for 'control' and the notation for 'a' before it can recognize the presence of 'control-a' as a syntactic primitive.

Is the presence of a set of switches in a language sufficient to prevent it from encoding information explicitly? That depends on how local is the connection between a switch and the notational element it changes. For instance, if a switch may be set at an arbitrarily early point in a sentence, then the user of the language must keep track of which switches have been set. This never gets complicated in a computational sense because the user need only keep a stack of active switches and compare each current element against the stack. In fact, even if a switch once turned on may be switched off by a later switch, the job of tracking which switches are on is still simple. For again, the user need only check the current element against the stack. If that element is a switch, the user removes a member from the stack. If it is a non-switch, the user either changes its meaning (and removes a switch), or the user interprets the element with its standard meaning. Accordingly, the only taxing feature of this language is that one must remember which switches have been set. If there are few switches, and each switch is a distinct notational element such as Meta or Control which cannot itself be switched in meaning, this task should be trivial. If humans do not, in fact, find reading this language trivial, that they may tell us something about how long, and how large is the stack which humans can manage with ease.

Switches are a simple method for expanding the vocabulary of a language while keeping notation concise. But they do not allow us to compact our meaning theory: for every switched symbol there must be a meaning axiom.

In most economical languages, however, the same switch can effect different symbols in similar ways. For instance, the manual describing the commands available for my editor tells me that 'a' and 'e' when following 'Control' and 'Meta' mean *beginning* and *end* respectively, while 'Control' and 'Meta', when preceding 'a' and 'e,' indicate *line* and *sentence* respectively. Thus *'Control-a'* means *move cursor to line's beginning*, while *'Meta-a'* means *move cursor to sentence's beginning*.[7]

Such systematic variation in meaning allows us to compact our meaning theory. We can save memory. But it does not save computation. Once again, a stack will be needed to mark the switches that are on, and once again each notational element will be compared against it. Memory is saved because we can get by with fewer meaning axioms. But now instead of determining the meaning of *Control a* by looking up that entry in a memory intensive meaning theory, we must compute its meaning by looking up *Control* and *a* separately, and combining those meanings according to a general rule of combination. Computation increases.

Should we say such languages explicitly encode information? Again, it seems an open question. As codes go, such languages are no more compact than ordinary switched languages. They do get by with a more memory-compact decoder; but that decoder may take slightly longer to apply than one which just looks up the answer. Everything turns on how much time it takes to determine meaning. If the set of switches is small, or the host's decoder is highly parallel, then meaning may be determined almost instantly.

Let us grant that there is room for doubt whether languages with dedicated switches can be counted on always to encode information explicitly. There can be no doubt, however, that if we allow that *any* symbol may serve as a switch for any other, then information will sometimes be hidden. For now every symbol may, in principle, be ambiguous.

The net effect of unconstrained ambiguity is that syntactic rules may be arbitrarily complex because the disambiguations they must help to perform may be arbitrarily complex. The set of syntactic rules necessary, for example, for deciding which type of 'a' we find in a given expression, may enjoin us to examine many letters, or combinations of letters, before and after *a*. Because each of these letters in turn might be ambiguous we might eventually require a set of syntactic rules that computes a function of staggering complexity. For instance, to decide what *a* means in a certain context we might have to compute a function such as $a = f(b, c, f_1(d), f_2(f_3(f_4(e))), \ldots)$ to determine the particular *a* we are dealing with, and then look up its meaning.

To take an example from English, the sentence *Police police police police police*[8] is in principle grammatical. Read as: Police who are policed by policemen, are themselves policers of policemen, we see that each occurrence of 'police' has a unique syntactic and semantic identity. In some contexts 'Police' means *policeman*, in others it means *to enforce*. Complexity enters because there are so many combinations of meanings to consider. The amount of computation needed just to determine what a single expression means rises exponentially with the length of the sentence. Should we say that each occurrence of 'police' is explicit when it is so hard to identify the symbol?

If there remains any doubt here it is because we have not completely resolved whether to base our judgment of explicitness on the computational cost of recognizing and using a symbol or on the fact that the notational and syntactic elements are well-defined. On the other hand, it is tempting to go with our eyes and say that if we can see there in front of us a well defined symbol we know to be meaningful, it must be explicit. We can see the term *police*, so its meaning must be explicitly encoded. Its identity is well defined, though hard to determine. On the other hand, if efficiency is important then it is not enough that there exists *some* mechanism, however complex, for recognizing the symbol. For by that token, structure hidden arbitrarily deeply in a represention could also be called explicit. Thus, an edge might be said to be explicitly encoded on a retinal intensity matrix because there is an effective procedure for extracting it. This is absurd. Surely there is a substantive difference between information that is explicitly encoded and information that must recovered? But then efficiency does matter; it is the driving condition.

Accordingly, each step away from total movability — total context independence — is a step which increases the complexity of the processes which recognize a symbol and its meaning. At some point these recognition costs become too high and a language becomes unable to encode information explicitly. Such languages are too complex to be read and understood immediately. The truth in the movability condition then is this:

> **Condition (2):** An ambiguous language may explicitly encode information only if it is trivial to identify the syntactic and semantic identity of the symbol.

Immediately readable. I have been arguing that for an expression in a language to encode information explicitly it must be trivial for a user of that language to recognize the notational components of the expression; trivial for that user to recognize the syntactic role of those components, and trivial as well for it to recognize the meaning of both the components and the expression as a whole. In short, I have been arguing that the expression must be immediately readable.

But what does it really mean for a recognitional process to be trivial? Can we say in more precise computational terms what immediately readable amounts to?

The definition I propose is that we call a recognitional process *trivial* if there is a mechanical process that identifies the relevant property in *constant time*. Constant time is a measure of the absolute computational complexity of a process. It means that the number of computational operations needed to solve a problem is a constant independently of the size of the problem instance. For example, the time needed to recognize whether a binary number is even is constant regardless of size because the test for evenness is local, it involves checking the last digit. Similarly, to decide if an encoding is all 1's, all we need to do is add 1 and check to see if the new number overflows, i.e., has a longer encoding length. This too can be done in constant time.

Constant time is the smallest complexity order known. Few computations fall within it. Accordingly, in saying that an expression explicitly encodes information only if it can be parsed and interpreted in constant time, we are placing a strong and precise condition on explicitness. Instead of vague intuitions about structures being *immediately usable* or permitting their information to be *directly read off*, we now have a precise principle for interpreting immediacy.

The criterion of constant time, like all complexity orders, is meaningful only if we know what can be done in a single step. For example, on some machines a piece of information can be retrieved in a few steps regardless of how big the memory storage unit is. On such machines, retrieval is a constant time operation. On other machines, as memory size increases the number of steps needed to find information rises logarithmically. On these machines, memory retrieval is a log time operation.

The one weakness in using constant time as a mark of immediacy is that sometimes we may recognize small inputs immediately, even though the recognitional process for arbitrary inputs takes nonconstant time. Strictly speaking, complexity is not meaningful for finite inputs; for in principle the answers to any finite problem can be stored in a giant look-up table, where the minimal amount of computation required to find an answer would be approximately the same across all problems.

Yet sometimes this is exactly what we believe is the case: patterns are recognized immediately because they are matched in memory.

To accommodate this intuition let us think of operators as having a certain spatial *attention span*. We may think of the attention span of an operator as its input window, the number of basic notational elements that may be in focus at any time. In a sense, it is the measure of parallelism inherent in the operator.

For example, to determine whether a given encoding is symmetric — as in *0110*, where one half is a mirror image of the other — a system with an attention span of 1 will iteratively compare numerals at each end. In our own case, however, if the number of digits is small, we can tell at a glance, without aid of iteration, if the two are mirror images. We gestalt this *global* property of the numeral. And so we can decide symmetry in constant time for any numeral up to some length *n*. Once *n* is reached there are too many digits to fit inside our attentional field. At that point, we too must iteratively scan. And so the property is no longer explicit. We no longer just 'see' it.

Attention span sets an upper bound on what can be immediately gestalted. That means that the net affect of attention span for larger problems is, at best, to reduce by a constant factor the absolute number of steps required.[9] This has the effect that complexity orders are unchanged by attention span. If a problem was order log it remains order log, except for problem instances that fall within the attention span. The upshot is recognition processes that are normally non-constant time, such as recognizing switched symbols, remain so, unless the symbols are, for instance, close together in time or space.

Accordingly:

Condition (3): symbols explicitly encode information if they are either:
- readable in constant time; or
- sufficiently small to fall in the attention span of an operator.

Meaning. The final condition of explicitness is that every expression that contains explicit information must have a definite information content. It is tautological that a symbol can explicitly encode information only if there is some information that it encodes. Although I have been arguing that the question of explicitness really concerns how quickly information can be made available in an encoding, I have yet to explain what I mean by *information*.

Just what it is for a state, structure, or process to express meaning remains the premiere issue of twentieth-century philosophy. My objective here, though, is not to clarify what information means, but to show that whatever theory of meaning one holds, the same concept of immediate apprehension can apply. Accordingly, let us consider some contenders.

The first theory of meaning is the most widely held:

> A system immediately recognizes the meaning of a symbol if it grasps the contribution which the symbol makes to the meaning of the larger expression of which it is a part.

For instance, recognizing the meaning of 'cat' in *the cat in the hat*, on this view, consists in entering a state which contributes to the larger state of knowing the truth conditions of the whole sentence. To discharge the question-begging term 'knowing' we might reformulate the claim in more verificationist terms. Thus, recognizing the meaning of 'cat' in *the cat in the hat* consists in turning on a subset of the abilities involved in recognizing when a cat is in a hat. Associated with 'cat,' then, would be a set of abilities, or dispositional states, some of which are perceptual and motor, which are triggered (in constant time) by the appearance of the symbol 'cat' and which can combine with other abilities triggered by other symbols. The substance of the theory lies in first identifying the relevant abilities, and second explaining how they come to be triggered in just the right way and just the right order to produce appropriate composite abilities.

When understood in its referential version, this theory requires that the agent be able to 'know the referent' of an expression in constant time. If we identify this condition with the agent's being in a certain state, then we can say that a symbol explicitly encodes a certain datum of information for a system *S* only if *S* can enter a state of knowing the truth conditions of the expression in constant time. It is an empirical and conceptual problem to determine what this state is for any given system.

The second theory of meaning also attempts to explain in process terms what understanding consists in. Unlike the first, this theory places most emphasis on reasoning skills.

> A system immediately recognizes the meaning of a symbol if it directly enters a state that rationally constrains the system's future inferences and judgements.

For instance, recognizing the meaning of 'cat' in *the cat in the hat* may consist in entering a state which constrains the class of deductions and inductions that the system might make about cats, hats, being inside, and so on. These possible inferences and judgements are somehow *rationally regimented* by the semantic import of 'cat.' The substance of the theory lies in explaining first, the norms of reasoning and judgment, and second, how a given proposition fares in the cognitive economy of a rational agent. If we accept the second theory of meaning, then, we will say that a symbol explicitly encodes a certain datum of information for S only if S can, in constant time, enter a state which appropriately constrains S's possible trajectories of reasoning and judgment. It is an empirical and conceptual problem to determine what these constraints are for any given system.

The third theory of meaning I shall mention, unlike the other two, does not attempt to provide a full blooded account of meaning that grounds understanding in a set of abilities to recognize truth conditions or to reason rationally. This theory offers no explanation of the abilities an agent must have to understand a concept. It does, however, tie understanding to the activation of symbols. These symbols in turn might activate abilities.

> A system immediately recognizes the meaning of a symbol if it accesses (in constant time) any relevant associated symbols.[10]

For instance, recognizing the meaning of 'cat' in *the cat in the hat* may consist in retrieving certain other symbols. Eventually this process must ground out in basic abilities to act, perceive, and reason. Because these grounding abilities might be slow to activation they cannot be part of the explicit content of a symbol. They may be part of the symbol's total meaning. But they are not explicitly encoded.

Thus to take an example from English once again, the symbol "him" in the sentence 'Then John read *him* his rights.' explicitly encodes only that information that can be directly read from it. The referent is not part of this information, for a parser may have to look arbitrarily back among other sentences to locate it. Sometimes this process of locating the referent of a pronoun can be done by syntactic means. There are binding rules and in certain cases these will suffice to determine reference. In short sentences, though, the referent may be within the attention span of the agent. Hence explicitly encoded. For example, *John shot himself*. In other cases, the parser may have to rely on extra-syntactic knowledge which no agent could have within its attention span. For instance, to determine the referent of 'it' in 'Christine put the candle onto the wooden table, lit a match, and lit *it*,' the interpreter must exploit knowledge about tables and candles. Tables are rarely lit, and especially not simply by applying matches to them. The knowledge that is employed to decide that *it* refers to the candle, is located somewhere in the system, but there is no predetermined place it must be. It could be encoded in the state of the interpreter, in rules for meta-level control, or in a lexical data base. Determining where this state is could take arbitrarily long. Accordingly, its content is not explicitly present.

Sketchy as these accounts of meaning are they provide us with a hint at the fourth condition, for they share a common feature: namely that whatever meaning is, the states, structures, or processes that instantiate apprehension of that meaning must be able to be turned on in constant time. Thus:

> **Condition (4)**: The information which a symbol explicitly encodes is given by the set of associated states, structures, or processes it activates in constant time.

IMPLICATIONS

I have claimed throughout this paper that our intuitions about explicit information are confused. Explicitness really concerns how quickly information can be accessed, retrieved, or in some other manner put to use. It has more to do with what is present in a process sense, than with what is present in a structural sense. Three notable consequences follow.

The first consequence is that not everything which we can assign a meaning to is explicitly encoded. If a sentence takes longer than constant time to interpret, then its meaning is not on the surface. Again this holds whether the sentence is a declarative, such as *Buffalo buffalo buffalo buffalo buffalo*[11] or a procedure. For instance, the sentence (add(square 2) (5th-root 3125)) in one sense means (add 4 5). But not explicitly. On the surface, it means to add the square of 4 to the 5th root of 3125.

On the other hand, if (add 4 5) were a constant time procedure we would be obliged to say that its evaluation *was* explicitly represented. This has the consequence that information may be explicit even if it is not represented by its canonical symbol. Returning to our adding example, (add 1 1) explicitly encodes 2 for any system that can add in constant time or which has memorized the answer and can trivially retrieve it. Normally, we would expect that the difference between 2 and (add 1 1) is precisely that the first explicitly encodes 2 while the second explicitly encodes information about a procedure. Yet, while it is true that (1 + 1) does explicitly encode information about adding, it also explicitly contains information about the evaluation of the procedure. This ambiguity rarely occurs, however, because usually, procedures do not evaluate in constant time. So the information they contain is not explicit. It is only when the procedure is a constant time procedure, or when the performance of the procedure is in the attention span of the host, that the symbols become ambiguous. In such cases, there is no reason not to regard the procedure call as both a call to a certain procedure with certain values, and a special name for its evaluation.

If this seems unreasonable, consider a related case. In binary notation the last digit which ordinarily encodes explicitly the number of units, also encodes whether a number is even or odd. Should we say information about evenness is explicitly present? I think we must. We would not likely find an entry in a Tarski meaning theory stating that '. . .0 means even number.' But if the host system has an understanding of what even or odd is, and it tests for evenness by checking the last digit, then what grounds have we for denying that the last digit carries two meanings explicitly? It is reasonable to suppose that '1' in the units location just is how we represent oddness. To be sure, a system may not always make use of information that is explicit. Being explicit is not the same as being *occurrent*. Occurrent states are occurrent because control has passed to a procedure which interprets the symbol. The control makes occurrent an interpretation. But what a symbol explicitly represents is independent of what is happening in the control structure. The only connection is that if we do not *know* all the constant time procedures that might act on the representation we cannot know all the bits of information the representation might encode explicitly.

The upshot is that an ordinary meaning theory for a language will not specify all the meanings of a symbol unless that theory was constructed with foreknowledge of the constant time procedures present in the system, and foreknowledge of what can be the contents of a span of attention.

This need for foreknowledge is one reason procedural semantics is hard. But by restricting the focus of procedural semantics to the constant time effects a representation can have rather than to arbitrarily long term effects, the project of procedural semantics may become more tractable. This, I take it, is one substantial virtue of the proposed theory of explicitness.

A second consequence of tying explicitness to rate of access is that there is no principled distinction between declarative and procedural representations. It is irrelevant whether 'place 10' means a location in some vector, or it means some compiled procedure which when invoked starts a set of processes for placing object 10 somewhere. Either meaning may be explicitly encoded. What counts is that the link to the vector location or to the compiled procedure be negotiable in constant time.

This focus on process is significant because it encourages us to think about information processing in terms of informational movement. Representations are inert unless coupled with processes which interpret them. This trivial point is often ignored in correlational theories of meaning. Thus we find correlationists looking for static structures to interpret as representations even in essentially dynamic systems such as relaxation networks. The truth, however, is that for such dynamical systems, information content is to be found in the coupling of structure with process. It is the union of structure and process which can explicitly encode information.

The final and most philosophically significant point is that the

Language of Thought as usually conceived is capable of generating representations that are not actually explicit in the proprietary sense I have been discussing. This has the consequence that, at a minimum, one of the following is *false*:

- the language of thought is the best level of analysis to represent perspicuously the episodes in our *mental life;*
- the events in our mental life are identical with operations on explicit representations;
- the language of thought perspicuously describes human information processing.

The primary motive for postulating a language of thought, it will be recalled, is that there are important regularities in thought, belief and action that would otherwise be missed were we to look for explanations at the neural level alone or at the competence level alone. Competence theories are above all structural theories: they tell us what is computed, what knowledge a system must have to be able to perform those computations, and, in the best competence theories, they tell us something about the decompositional structure of the computational process. But the details of the actual computational trajectories they leave undescribed. The opposite is true at the neural level: so many details are available about particular computational trajectories individual people follow that it is hard to find a level of characterization which perspicuously captures the important informational transitions that generalize well. The language of thought is meant to be the right level of abstraction to describe these informational transitions.

Does the language of thought successfully describe this middle level? In most discussions, the language of thought is assumed to be as complex as first order predicate calculus. Yet we know that parsing and assigning a semantic interpretation to first order predicate calculus is a non-constant time process. Something must give. We cannot believe the language of thought to be simultaneously a complete description of human information processing and a complete description of human mental life. If we opt for the language of thought as a complete description of the mental, then we must forsake identifying mental activity with computation on explicit representations.

If any of these three conclusions seems counterintuitive it is because we tend to think of explicitness as a local property of a data structure: something which can be ascertained without studying the system in which it is embedded. But that is a mistake.

ACKNOWLEDGEMENT

My thanks to Eric Saund, Greg Smith, and Bob Stalnaker for many helpful hours of discussion.

NOTES

[*] Support for this work has been provided in part by DARPA under Office of Naval Research contracts N00014-85-K0124, and by the Army Institute for Research in Management, Information and Communication Systems contract number DAKF11-88-C-0045.

[1] See, for instance, Dennett, Three Kinds of Intentional Psychology, reprinted in *The Intentional Stance*, Cambridge, MA: Bradford/MIT Press 1987), 55-6; Hector Levesque, and Ron Brachman, "A Fundamental Tradeoff in Knowledge Representation and Reasoning," reprinted in *Readings of Knowledge Representation*, eds. Brachman and Levesque (1985), 41-70; Rob Cummins, "Inexplicit Information," in *The Representation of Knowledge and Belief*, eds. M. Brand and R.M. Harnish. (Tuscon: University of Arizona Press 1986).

[2] Even if we choose to see them as computers merely because we can interpret them at an input output level as implementing functions, we certainly can never know that they are computing those functions correctly unless we can interpret intermediate states.

[3] The issue here is not whether we can decide *after* a successful proof has been found just how many inferential steps are required to reach a given theorem. The issue is that there is no means of determining during the course of searching for a proof how far away that particular theorem remains. We cannot tell whether we are getting closer or moving farther away from the theorem.

[4] Cf. Noam Chomsky, *Rules and Representations* (Oxford: Blackwell 1980); Stan Rosenschein, "Formal Theories of Knowledge and AI and Robotics." (*SRI* 1985; Hubert Dreyfus, *What Computers Can't Do*, (New York: Harper and Row 1979); and Terry Winograd and Fernando Flores, in *Understanding Computers and Cognition* (New York: Addison-Wesley 1987).

[5] For instance, in *Psychosemantics: The Problem of Meaning in the Philosophy of Mind* (Cambridge, MA: MIT Press 1987), 25, Fodor states: "According to the Representational Theory of Mind, programs — corresponding to the 'laws of thought' — *may* be explicitly represented; but 'data structures' — corresponding to the contents of thoughts — *have to be*" (Fodor's emphasis).

[6] Brian Smith, "On the Threshold of Belief," to appear in *Foundations of Artificial Intelligence*, special edition of *Artificial Intelligence*, ed. D. Kirsh; Rosenschein, "Formal Theories of Knowledge in AI and Robotics."

[7] It is clear that the designers of this editor chose command names they thought reflected some systematicity, in order to make the commands easier to remember. The commands themselves, however, they implemented by a look-up table, thereby treating names as fusions. This design choice was a typical store versus compute choice. It is clearly more efficient to store all the combinations of letter sequences as molecular primitives, if the number of combinations is small, rather than parse each sequence each occasion of use. But as a vocabulary increases it pays to parse. For bigger applications it may be wiser to use compositional axioms, instead of a look-up table.

[8] This example, and the proof that unbounded ambiguity leads to exponential computation is drawn from *Computational Complexity and Natural Language* by Barton, Berwick, and Ristad (Cambridge, MA: MIT Press 1987).

[9] Of course, if attention span could be flexibly expanded, then the complexity profile could be altered. A machine that is able to dedicate the ever-increasing hardware and memory required to enlarge its input window can potentially "see" the answer to ever larger problems. But the overhead in terms of the number of wires and connections needed to sustain an enlarged window of attention will grow at the same rate as the number of steps grows for systems with fixed attention span. So there will always be relatively narrow attention spans.

[10] Cf. Allen Newell, "Physical Symbol Systems," *Cognitive Science* 4 (1980): 135-83; and Allen Newell, "The Knowledge Level," *Artificial Intelligence* 18: 1 (1982): 18-127.

[11] Meaning: Buffalo from Buffalo outwit buffalo from Buffalo.