Title
Enzymatic computation and cognitive modularity

Journal
Mind & Language, 20(3)

ISSN
0268-1064

Author
Barrett, H Clark

Publication Date
2005-06-01

Peer reviewed

Enzymatic Computation and Cognitive Modularity

H. CLARK BARRETT

Abstract: Currently, there is widespread skepticism that higher cognitive processes, given their apparent flexibility and globality, could be carried out by specialized computational devices, or modules. This skepticism is largely due to Fodor's influential definition of modularity. From the rather flexible catalogue of possible modular features that Fodor originally proposed has emerged a widely held notion of modules as rigid, informationally encapsulated devices that accept highly local inputs and whose operations are insensitive to context. It is a mistake, however, to equate such features with computational devices in general and therefore to assume, as Fodor does, that higher cognitive processes must be non-computational. Of the many possible non-Fodorean architectures, one is explored here that offers possible solutions to computational problems faced by conventional modular systems: an 'enzymatic' architecture. Enzymes are computational devices that use lock-and-key template matching to identify relevant information (substrates), which is then operated upon and returned to a common pool for possible processing by other devices. Highly specialized enzymes can operate together in a common pool of information that is not pre-sorted by information type. Moreover, enzymes can use molecular 'tags' to regulate the operations of other devices and to change how particular substrates are construed and operated upon, allowing for highly interactive, context-specific processing. This model shows how specialized, modular processing can occur in an open system, and suggests that skepticism about modularity may largely be due to failure to consider alternatives to the standard model.

1. Introduction

> So we can now (maybe) explain how thinking could be both rational and mechanical. Thinking can be rational because syntactically specified operations can be truth preserving insofar as they reconstruct relations of logical form; thinking can be mechanical because Turing machines are machines.
>
> However things eventually work out for computational nativism in cognitive science, this really is a lovely idea and we should pause a moment to admire it. Rationality is a normative property; that is, it's one that a mental process *ought* to have. This is the first time that there has ever been a remotely

Thanks to Gary Brase, Greg Bryant, Peter Carruthers, Leda Cosmides, Tom Dickins, Ray Gibbs, Rob Kurzban, Karthik Panchanathan, Dan Sperber, Peter Todd, John Tooby, and an anonymous reviewer for useful feedback and discussion of ideas presented here.

Address for correspondence: UCLA Center for Behavior, Evolution, and Culture, Department of Anthropology, Haines Hall 341, Box 951553, Los Angeles, CA, 90095-1553, USA. Tel (310) 267-4260
Email: barrett@anthro.ucla.edu

Mind & Language, Vol. 20 No. 3 June 2005, pp. 259–287.

> plausible mechanical theory of the causal powers of a normative property. The first time ever (Fodor, 2000, p. 19).

Of all the theoretical approaches that have been offered for understanding mental processes, none has been as successful, both in terms of popularity and explanatory power, as the computational theory of mind (CTM). CTM views thought as computation: the use of algorithmic rules to systematically map inputs, i.e., information instantiated in neurochemical patterns, onto outputs, i.e. different patterns of information that have been systematically transformed. What makes the patterns in question information is that they 'stand for' something: that they, in turn, can be mapped onto something in the world or mind. The mapping operations are computations. From Babbage, to Turing, down to modern artificial intelligence and neural network researchers, this has been and remains a beautiful and beguiling idea: it means that 'to the extent that thought consists of applying *any* set of well-specified rules, a machine can be built that, in some sense, thinks' (Pinker, 1997, p. 68).

However, whereas Pinker and many others hold that *all* (or most, or much) of actual human thinking may be computational in the sense described here, Fodor, among others, is skeptical. As he puts it, 'it hadn't occurred to me that anyone could think that it's a very *large* part of the truth; still less that it's within miles of being the whole story about how the mind works' (Fodor, 2000, p. 1).

This has become one of the central debates in the cognitive sciences: the extent to which the processes of thought are carried out by computational devices or procedures, and in particular, specialized ones (see, e.g., Buller and Hardcastle, 2000; Coltheart, 1999; Fodor, 2000; Samuels, 1998; Samuels, Stich and Tremoulet, 1999; Segal, 1996; Sperber, 1994). On one side, it is argued that those aspects of thought that are functionally organized and reliably truth generating could scarcely be carried out by anything *but* such procedures (this has been called the 'Massive Modularity Hypothesis'; Sperber, 1994). On the other side, it is argued that such an architecture could not possibly account for the observed flexibility, context-sensitivity, and globality of thought.

As Sperber (1994) has observed, if the brain contains specialized computational devices, what their properties are is 'a matter of discovery, not stipulation'. But current debates about modularity have been hung up on very specific and narrow assumptions—stipulations, in fact—about how specialized computational procedures must be instantiated. In particular, Fodor's (1983) model has been very influential in this regard, and as he has shown, accepting it appears to entail rejecting the notion that most of what we think of as cognition could be handled by computational devices of any kind (Fodor, 2000). But might there not be other ways of instantiating specialized computational procedures?

My objective in this paper is to show that, while the Fodorean model of modularity has been both important and influential, it has been perhaps *too* influential, because it has foreclosed ways of thinking about modularity other than the very specific model he proposed in his 1983 book. In principle, there are a large number of possible computational architectures that are non-Fodorean

in nature. Here, I elucidate the features of one possible system, an enzyme-like system in which modules all have access to a central blackboard or bulletin board, and in which there are no rigid routing procedures between modules. The enzyme metaphor for modularity was originally proposed by Sperber (1994), and bears resemblance to other, existing models in the cognitive sciences, such as Selfridge's pandemonium model (Selfridge and Neisser, 1960) and the classifier system model developed by Holland and colleagues (Holland, Holyoak, Nisbett and Thagard, 1986). Unlike Fodorean modular systems, enzymatic systems require no 'meta-module', or routing system, to route information, and do not require input restriction or compartmentalization mechanisms to restrict their access to subsets of a global database. These are important details of the Fodorean model which, as Fodor (1983, 2000) shows, render it implausible as a model of central cognitive processes. The mistake, I argue, lies in assuming that Fodor's architecture is the only possible one.

Here I return to the first-principles logic of CTM as a source of predictions about the design of cognitive architecture. CTM was originally invoked to explain the functionally organized, truth-preserving and truth-generating qualities of cognition. Its main assumptions are that cognition is computation, and that cognition works because computational procedures are designed to systematically derive useful information from their inputs. Although it is often left unsaid, this cannot be an accident: to the extent that computational procedures are so designed, it is because of a history of natural selection. Accepting these premises entails searching for the design features that would be required for such a system to be implemented. And this search, in turn, can be informed by consideration of the problems that such a system would face in carrying out its evolved functions.

As Fodor (2000) points out, central cognitive processes face information-processing problems that peripheral systems do not face. Here, I will focus on the adaptive problems faced by systems designed to (1) preserve the systematically true properties of information they process, while (2) routing information to procedures that can generate useful inferences from it. Of all possible architectures, not all solve these problems equally well. This is a source of insight about the design of cognitive architecture. While this paper in no way solves all of the problems faced by central computational systems, it is intended to show how adopting the design stance with respect to information-processing problems can point to possible solutions in ways that defeatist skepticism about the power of computational systems cannot.

2. First Principles of CTM

Fodor (2000), in summarizing CTM, suggests that the rationality of thought depends on the truth-preserving qualities of computational operations, which are truth-preserving in turn by virtue of the fact that they obey the laws of logic. Although the terms *logic, rationality*, and *truth* may approximately capture certain

principles whereby the mind works, they must be qualified. They cannot literally be general principles of cognition, because minds are not, in fact, logic machines, but survival machines. The functional design features of cognitive devices were shaped by feedback from their effects on the survival and reproduction of the organisms that contained them, and nothing else.

Organisms' computational systems are designed to preserve certain properties of information as it passes through their computational procedures because maintaining those properties had important consequences for inference and decision-making that impacted survival and reproduction in ancestral environments. In principle, organisms in the past could have entertained many true, rational, and logical thoughts (as well as many false, irrational, and illogical ones) that had no impact on fitness, and therefore, no impact on the cognitive design of the organisms entertaining them. Of those properties of information that evolved computational systems preserve, many might rightly be regarded as 'true'; some might not be; and other kinds of information that might be perfectly 'true', or have plenty of implications for matters of truth, might be systematically discarded because they had no impact on fitness in past environments. A similar analysis applies to logic and rationality: what is respected by the computations of evolved systems is not logical principles *per se*, but rather, computational principles that reflect enduring structural properties of the world, some of which might correspond to principles of formal logic, and some not. On this view, deviations from rationality are not only possible but likely, if the stable properties of the world on which computational rules depend for their truth-preservingness are systematically altered.

In short, placing our hope for the rationality of thought on the existence of domain-general, abstract principles of logic is misguided because there is no reason to expect that a mind containing such principles ever evolved. However, natural selection *can* engineer computational mechanisms that embody domain-specific, content-specific, context-specific principles of inference and information handling that evolve because they are 'truth-preserving' in a more narrow sense, relying on specific structural properties of the world that have been true in the past (Tooby and Cosmides, 1992). Fodor is right about the beauty of truth-preserving operations, but wrong about the *kinds* of truth that they evolved to preserve.

What do we mean by 'truth preserving'? To a logician, a truth preserving operation is one that reliably generates true propositions as outputs, provided that the inputs are true. For example, a categorical syllogism is truth preserving: *All men are mortals; Socrates is a man; therefore, Socrates is mortal*. Not only that, operations such as this are truth deriving, or truth generating: the output is something we didn't know before. One can imagine other, more content-specific operations that generate reliable inferences from inputs, e.g.: *That animal is a predator; that animal is looking at me; therefore, that animal might chase me*. This kind of property—the ability to infer or predict something that was not previously known on the basis of available inputs—would be a useful property for evolved computational systems to have, even though a given operation might be truth preserving only under restricted conditions, and only for restricted inputs. When

we speak of truth preserving operations, we can speak of them quite legitimately in this sense.

In addition, in a system in which information passes through many stages, selection will favor operations that preserve some elements of the information as it is processed, and alter others. For example, information about a lion that enters the retina will undergo many transformations as it passes from system to system. As it does, it is important that certain key aspects of the information not be lost: for example, the fact that it is *about a lion*, the fact that the lion is over *there*, and so on. In short, we expect processing systems to be designed to preserve information of certain kinds, and to discard others. Subsequently, there must be a way of reliably getting information that has been preserved to those systems designed to use it.

3. Specialized Computational Devices

In order for a computational system to function, it must contain inference procedures, or rules. These cannot be just *any* rules, because many would systematically produce nonsense rather than truth. Moreover, there must be more than one rule, because what counts as truth preserving depends on the kind of operation being performed as well as on the kind of information it is being performed on. There is a 'truth' about whether my index finger will land on the space bar of my keyboard when I reach for it that is not computed by my mind using the rules of syllogistic reasoning. What one rule can do, another one can't; each must be tailored to the kind of operation for which it is responsible. This is the origin of the notion of computational specificity: computational processes require multiple, specialized kinds of algorithms.

Marr (1982), Fodor (1983) and others have suggested that such algorithms could be instantiated in specialized computational devices (SCDs). An SCD is a device that (1) accepts information of a particular kind (i.e., information that meets certain input requirements), (2) performs specified operations on it (computations), and (3) outputs the resulting information in a format useable by other systems. In principle, such devices, and their potential benefits, are not difficult to imagine. For example, one can imagine a 'snake detector', designed to solve the problem of detecting snakes by monitoring visual input and firing, or outputting a representation along the lines of 'there's a snake' whenever it detects a snake-like, sinusoidal shape. To function properly, SCDs must (1) encounter the information that they are designed to process, and (2) encounter it in a format that they can use. This 'routing' problem constitutes a principal focus of current debates over the computational architecture of the mind (Fodor, 2000).

For SCDs in perceptual input systems, it is not particularly difficult to imagine requirements (1) and (2) being satisfied. For example, one could imagine a snake detector simply wired up to monitor retinal output for snake-like shapes. However, we know that in fact, raw retinal output does not simply afford representations of objects or shapes. There must be additional computational

devices, what Fodor (1983) calls sensory transducers, which take retinal output and turn it into a format other devices can use. There must also be devices that 'parse' this output into things like object representations, and pass them to devices such as snake detectors. As Fodor (1983), Marr (1982), Pylyshyn (1986), and others have pointed out, the visual system has a design of this kind: it is composed of many devices or modules, each designed to perform a particular task that others cannot perform.

In order for these mechanisms to carry out the functions for which they were designed, there is a sense in which they must be coevolved: the design of one depends, in some way, on the design of the other. In the case of the object parser and the snake detector, they must be co-designed in such a way that the snake detector can properly use and interpret the information produced by the object parser. In other words, not only must information be in a format that can be passed between the two devices, but there must also be a *convention of interpretation*. Because objects were, by definition, not marked (or parsed) as such in the input to the object parser, however the parser ends up marking them, the snake detector must be designed to exploit/assume this marking convention. This is a crucial step for truth preservation, and it is likely to be a general principle of modular cognitive systems. When one device outputs information, there must be properties in that information—aspects of its representational format—that allow its 'meaning' to be properly interpreted by other devices.

4. Possible Routing Architectures

One way of ensuring that information doesn't get misinterpreted is to have SCDs arranged such that the outputs of some SCDs are matched to the inputs of others in one-to-one fashion, like pipes. The output of one device is simply fed into the input of another device (see Figure 1). This is one way of handling problems of routing, and of ensuring that there is no miscommunication between devices in the interpretation of information formats and content. Information conventions can be extremely local, and the receiving system 'knows', or can be designed to assume, where the information is coming from, and therefore what it is about.

This is the kind of architecture that Fodor (1983) originally proposed for input systems (and other peripheral systems, such as motor systems). Such architectures are composed of 'vertical' systems, or faculties. In vertical systems, information flows one way, in a bottom-up fashion; modules are arranged in layers, such that once information enters a device in one layer, it cannot subsequently enter another device in the same layer; and information is routed from one device to another. Consequently, different modules have access to different pools or sources of information. As Fodor puts it, they are non-overlapping.

But in the imaginary example of the object parser and the snake detector, the object parser does not 'know' whether or not the information it just processed was

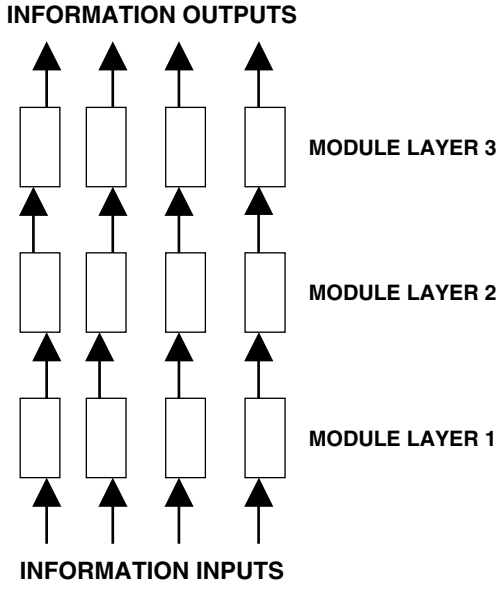

Figure 1 *Vertical pipe architecture*

about a snake, and therefore whether it should be sent directly and only to the snake detector (i.e. to a particular module in the next layer downstream). Rather, the snake detector sits there 'observing' the information coming out of the object parser, and waiting until it detects a match with its snake template. Many other detectors must be doing so as well; the object parser is not simply producing output for the benefit of the snake detector. In design terms, it would be an ineffective design if the snake detector, in making its snake / no snake decision, used up the information so that other devices in the same layer could not use it. Instead, it would be useful for the information to return to or remain in the pool—or at least not be used up—so that other devices, such as face detectors, artifact detectors, etc., could scrutinize it for a match (Kurzban, 1996).

Figure 2 illustrates an architecture that has this design, in which many modular devices have access to the same common pool or bulletin board of information (also sometimes called a blackboard architecture; see Pylyshyn, 1999). In this architecture, all representations can be scrutinized by all devices. When a representation fails to meet the input criteria of a device, it is returned to the pool unaltered (e.g. in Figure 2, representation 2 is returned unaltered by device B). When a representation does meet the input criteria of a device, it is processed and then re-posted to the same bulletin board for further scrutiny by other devices (e.g. in Figure 2, representation 2 is processed by device C and returned, perhaps with altered content, as 2C). It is possible to imagine a design that would leave a copy of the original unaltered in the pool as well.

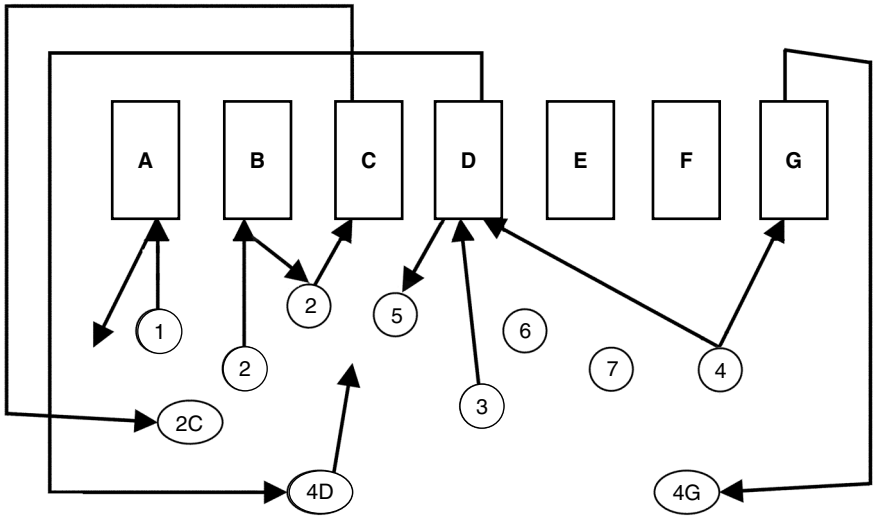

Figure 2 *Bulletin board architecture*
Information packets (labeled 1 through 7) are posted on a 'bulletin board' or public representational space that is visible to all modules (modules are labeled A through G). Modules constantly monitor the bulletin board for representations that they are able to process

Although re-posting violates the Fodorean assumption of one-way or strictly bottom-up, vertical processing, such a feature would be useful if a given representation could, in principle, satisfy the input conditions of more than one device in the system. Otherwise, the representation would get used up, i.e. removed from the pool, by the first device able to process it. One could imagine disastrous consequences if, for example, there were a squiggly line detector, distinct from the snake detector, in the same system; if a snake representation met the squiggly line detector first, no snake alarm would be sounded. Crucially, in the bulletin board design, the output of the object parser is publicly available for many, many SCDs to scrutinize. Presumably, the evolved function of the object parser is to produce object representations for the potential use of every computational subsystem that is designed to take parsed object representations as input.

Fodor himself has proposed that while peripheral systems have the sort of vertical, compartmentalized pipe architecture depicted in Figure 1, 'central' systems—those responsible for most aspects of 'higher' cognition, such as reasoning, inference, and so on—do not. In a sense, his model bears some similarities to the bulletin board model of cognition in that peripheral input systems output their computational products into a common, central pool (see Figure 3).

The existence of a central pool, however, is where the similarity ends, because Fodor (2000) explicitly denies that processing of information in the pool could be handled by specialized devices. He resists the idea that central processes could be

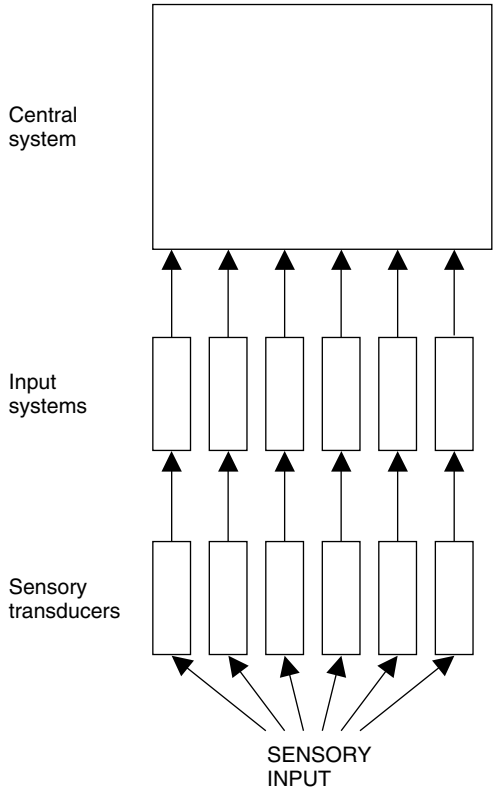

Figure 3 *Fodor's model of the mind*

composed entirely of specialized computational devices in which all devices in the system have access to exactly the same information. Indeed, he argues that because modular systems accept only local inputs, a global processing system composed entirely of modular devices could not be instantiated by any kind of computational system of which we are currently aware. However, not only is it possible for such a computational system to be instantiated, such systems actually exist, and in abundance. Every living cell contains computational systems that use an open, central pool of information, monitored by thousands of different kinds of specialized devices: namely, enzymatic processing systems. These systems show one way that a bulletin board architecture of the kind suggested above could be instantiated.

5. The Enzyme Model

The analogy between enzymes and cognitive modules was first proposed by Sperber (1994). While cognitive modules are not *literally* enzymes, they are

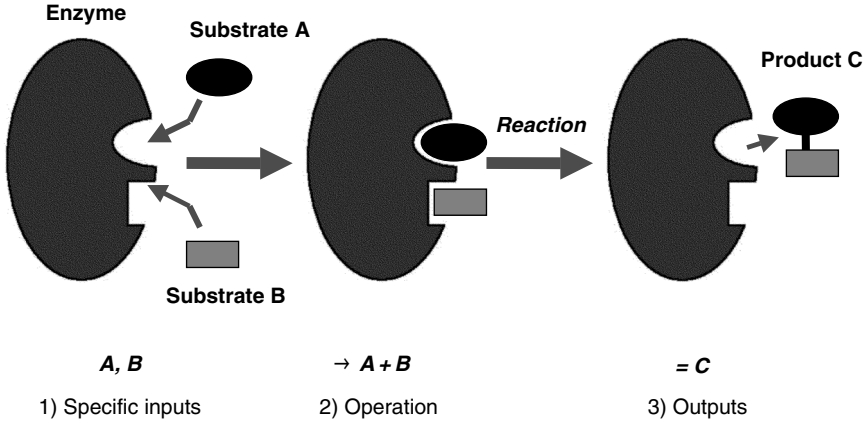

Figure 4 *Enzymatic computation*

computational devices that systematically transform inputs to outputs, without many of the restricting features of Fodorean systems. It is useful to consider the relationship between the design features of real enzymes and the possible design features of cognitive modules that have enzymatic properties, or what might be called 'cogzymes'.[1]

Enzymes catalyze reactions: they systematically combine substrates (molecules) to form new products (other molecules). This can be regarded as a kind of computation (see Figure 4; in fact, Magnasco, 1997, has shown that enzymes are Turing universal). Consider the following description by Coen (1999):

> ... Each cell contains many thousands of different types of proteins, each with a different shape, according to the process it guides. ... Suppose A encounters a large molecule, a protein, that has a shape with a nice little pocket that A fits into very comfortably. We could imagine, for example, that the pocket in the protein matches the shape of the A molecule, like a lock matching a key. ... If the protein has another nearby pocket that matches molecule B, then when B is bumped into, it will also tend to stick ... if they are held in the right way, they will react with each other, joining up to form a new molecule, C. In this way, the shape of the protein, the structure of its pockets and crevices, can facilitate a reaction: A and B coming together to make C (Coen, 1999, p 24).

Like our hypothesized snake detector, enzymes use a template to detect specific substrates; they are passive, monitoring a pool of substrates until a match is found;

[1] Thanks to Tom Dickins for suggesting this term.

and when they encounter a substrate that matches their template, they systematically transform it into something new, in a rule-like fashion. Thus, enzymes have the three basic properties of specialized computational devices:

(1) *Enzymes accept information of a particular kind*, generally in the form of chemical substrates with particular properties that meet the binding specificity criteria of the enzyme in a 'lock and key' fashion.
(2) *Enzymes perform specific operations on the information they admit*, catalyzing reactions that produce reaction products with different properties than the input substrates. While an A + B → C type reaction was used in the example above (see Figure 4), other kinds of operations can be carried out by enzymatic systems as well, including C → A + B, A → B, and so on. In each case, the computation is specific and dependent on the properties of the substrates in interaction with the properties of the enzyme.
(3) *Enyzmes output the resulting information in a format useable by other systems*. In other words, the products of enzymatically catalyzed reactions can then participate in further chemical reactions. There can be enzymatic processing cascades, feedback loops, and so on.

The structural features of enzymes related to these properties are shown in Figure 5. Clearly, cognitive modules are *not* enzymes; enzymes are simply a metaphor. In this metaphor, information processing is catalysis. Enzymes are computational devices that solve problems that cognitive computational systems also face: they achieve functional specificity in an 'open' system. Some aspects of the metaphor are relevant for cognitive systems, and some are not. Relevant aspects include:

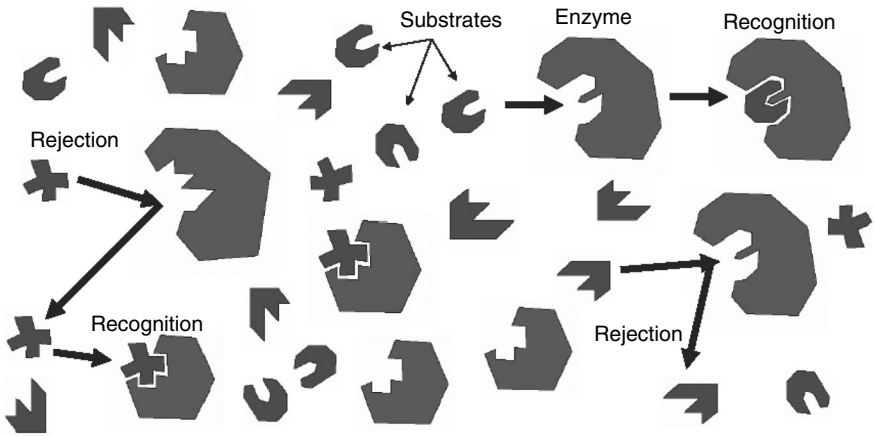

Figure 5 *Binding specificity*

Access generality with processing specificity. One can have an enzymatic system in which all of the enzymes in the system have access to all of the substrates, and in which only the 'correct' reactions are catalyzed. This is because of the lock-and-key nature of molecular recognition processes, which depend on the diffusion of information for their proper functioning (Alberts *et al.*, 1994; although there exist enzymes embedded adjacently in substrates to increase processing efficiency). To take another metaphor, one can put a variety of enzymes and a variety of substrates together in the same bag (i.e. without distinct compartments), shake vigorously, and presto: many reactions have been catalyzed, substrates have been turned into products, and, most importantly, the enzymes have catalyzed only those reactions they were designed to catalyze.

How is the input specificity of enzymes—their binding specificity and catalytic specificity—achieved? It is achieved through a template matching process, in which the fit of a chemical substrate to the binding site of the enzyme (an exposed surface of a three-dimensional macromolecule) is tested. Molecular recognition is achieved by the simultaneous formation of many weak chemical bonds between a substrate molecule and this template, in a lock-and-key fashion. There need be no mechanism that delivers substrates to enzymes; recognition events can occur by chance collision. Enzymes thus depend on diffusion in order for molecular recognition to occur. For the analog of such a system to be instantiated in the brain, there would therefore have to be a neural equivalent of diffusion, such as massively parallel distribution of information.

Multidimensional input criteria and byproduct processing. The reason that binding procedures of enzymes have analog properties is that many individual chemical bonding events contribute to the recognition process. The sum of these determine recognition; there can be better or worse degrees of fit. In this sense (i.e. fuzzy input criteria) the input procedures of enzymes resemble neural networks more than classical symbol-manipulation systems. There can be byproduct reactions, in which enzymes couple with substrates that mimic in some way, either by chance or by design, the substrates that the enzyme evolved to operate on (Alberts *et al.*, 1994). Many synthesized drugs are analogs of this kind, designed to fit the active sites of enzymes not originally evolved to process them. In Sperber's (1994) terms, these analogs would be part of the actual domain of an enzyme, though not part of its proper domain. Low levels of byproduct processing by enzyme-like cognitive devices may be what permit novel combinations and processing of representations beyond, in some sense, what the mechanism was designed to do. They might permit particular kinds of novel inference, such as metaphoric inference or analogy, via structural alignment/overlap. Nevertheless, enzymes evolve so that their recognition criteria are specific enough to function essentially as Boolean yes/no gates that effectively pick out a single kind of molecular substrate from all of the chemicals in the cytoplasm in which they normally find themselves, and reject others.

Class-level processing, carry-through, and tags. Another interesting property of enzymes is that they can be designed to pick out not just one substrate, but to

identify substrates of a particular class—for example, molecules that all have one set of subunits in common, even if subunits in other portions of the molecule vary. In some cases, the non-recognition portions of the molecule will be carried through the catalytic process without alteration (i.e. they will be 'preserved'), and in other cases these non-recognition portions will be operated on by the enzyme. This property of *carry-through* is an important one that not all systems have. For example, parallel distributed processing (PDP) systems have difficulty with operations in which one portion of a representation is systematically altered while the rest is unchanged (Marcus, 2001). In contrast, enzymes can easily perform operations such as, for example, *for all* X, $X \rightarrow X + Y$.

One way in which class-level processing is achieved in actual enzymatic systems is through the use of molecular tags: chemical substrates added by one enzymatic process, which are then used by subsequent processes for recognition purposes. Classes of substrates can be given a common tag, which can admit them to or restrict them from certain processes. Biochemical examples of such tagging processes include methylation and phosphorylation, which can enhance or block transcription of tagged DNA sequences.

Horizontal and top-down control. In real enzymatic systems, tags added to substrates can influence how they are processed by other devices they may later encounter. This allows for horizontal and 'top down' control, in which the outputs of devices can influence other devices at the same horizontal level, a kind of feedback which is not typical of Fodorean modular systems (Fodor, 1983, p. 64). In addition, enzymatic devices can emit tags or signals that are unattached to particular representations, but that have widespread effects throughout a system, turning many devices on or off, or modulating their operations (see Figures 6 & 7). In cognitive

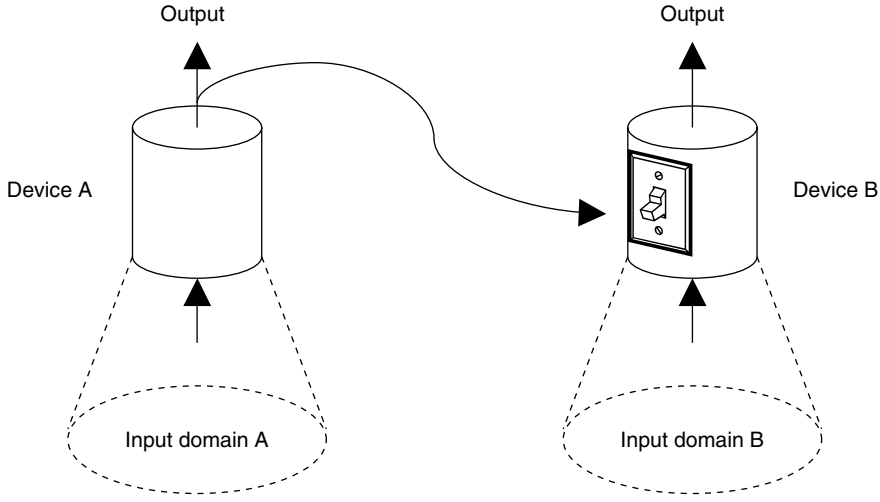

Figure 6 *Horizontal control*

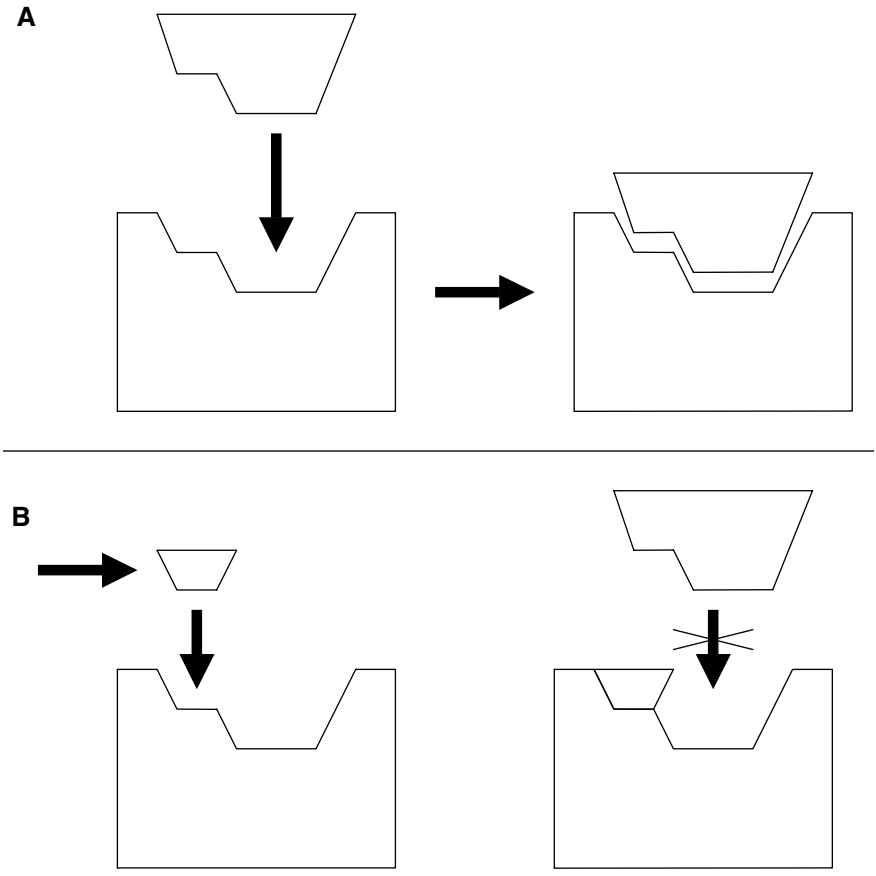

Figure 7 *An enzymatic switch allowing horizontal control*
In absence of a secondary compound, the enzyme is 'on' (A); presence of the secondary compound turns the enzyme 'off' (B)

systems, one could imagine signals such as a danger signal emitted by one device that turned on or off a host of other devices, or altered their processing thresholds.

6. Encapsulation and Domain Specificity

The hardware level biochemical details of enzyme functioning, such as weak covalent bonding, methylation, and so on, are not the central point here. Rather, the enzyme model is important because it points to potential computational solutions to problems that standard versions of modular architecture are said to face. Based on principles of computational design, Fodor (2000) has argued that

modular systems are unable to perform the kinds of tasks and account for the kinds of phenomena that are routinely observed in everyday thought, such as sensitivity to the context in which information is presented, the global revision of beliefs based on a single new piece of information, and so on. However, these problems depend critically on particular features of Fodorean systems that enzymatic systems, despite the fact that they are computational, highly specialized, and modular, do not have.

In *The Modularity of Mind* (1983), Fodor offered a list of properties that he suggested were typical of modules, including: Domain-specificity, informational encapsulation, obligatory firing, shallow output, rapid speed, inaccessibility to consciousness, characteristic ontogenetic course, dedicated neural architecture, and characteristic patterns of breakdown (see Fodor, 1983, for a definition and description of each of these properties). It is interesting to note that while Fodor explicitly pointed out that none of these criteria should be regarded as *necessary* or *defining* of modules, they have been widely interpreted as such (Coltheart, 1999). It is often assumed that a process that lacks one of these properties cannot, by definition, be 'modular'.

Each of these potential characteristics of modules is invoked and discussed to some degree in the literature on modularity. But increasingly, encapsulation has come to be seen as *the* defining feature of modules (Coltheart, 1999; Fodor, 2000; Samuels, 1998; Samuels *et al*., 1999; Segal, 1996; Sperber, 1994). As Fodor put it recently, 'A module *sans phrase* is an informationally encapsulated cognitive mechanism, and is presumed innate barring explicit notice to the contrary' (Fodor, 2000, p. 58). He does agree that one could speak of modularity without encapsulation, i.e. to define modules as 'functionally individuated cognitive mechanisms' (see Coltheart, 1999, for a similar definition in terms of domain specificity), but suggests, and rightly so, that 'Probably everybody who thinks that mental states have any sort of structure that's specifiable in functional terms qualifies as a modularity theorist in this diluted sense' (Fodor, 2000, p. 56).

What is encapsulation, then? Encapsulation refers to the fact that a module's operations are not sensitive to information outside of its proper inputs (in a certain sense, a non-encapsulated algorithm is therefore logically impossible). As Fodor (1983, p. 69) puts it, 'the claim that input systems are informationally encapsulated is equivalent to the claim that the data that can bear on the confirmation of perceptual hypotheses includes ... considerably less than the organism may know'. For example, if one nudges one's eyeball with one's finger, the world appears to move despite the clear and explicit high-level knowledge that it is the eyeball, not the world, which is moving. Presumably, this is because the visual mechanisms involved are not designed to take into account externally caused movement of the eyeball, or knowledge thereof. Fodor calls this 'modularity with a vengeance' (1983, p. 67). A second and related property is domain specificity, which refers to constraints on the range of information a module can access (Fodor, 1983, p. 47). Visual and auditory input analyzers do not have access to, or

use, the same information, and are therefore domain specific, at least with respect to each other (their inputs are non-overlapping).

While domain specificity and encapsulation are conceptually distinct properties, they are related in that they both concern constraints or boundaries on the information that a module uses. In Figure 6, the encapsulation of device B is violated by the fact that output from A can influence its operations (this output from A could include, for example, contextual signals that would alter how B interprets / processes its input). The breadth of 'domain B,' on the other hand, presumably determines how domain specific device B is (note that there is a sense in which information in domain A influences B's operations, through device A). In the case of enzymes, because their operations are causally linked to the shape of their input domain (active site), the properties of domain specificity and encapsulation are not necessarily separable; the exact same properties of the device's active site can determine both its input conditions (and, therefore, domain specificity) and its operations (and, therefore, encapsulation). In the case of real enzymes, it is interesting to note that substrates emitted by another enzyme can interact with an enzyme's active site and thereby alter its operations. The presence of a secondary compound can, among other things, block processing of a substrate by interfering with the active site, turning it 'off' (Figure 7), or be required to turn it 'on' by altering the shape of the active site so as to accept the substrate (e.g. a molecular tag can be required before the substrate is processed). Whether or not one wishes to consider these regulatory substrates part of the device's 'proper domain' or not seems to be largely a semantic issue that is made moot by precise specification of design. More importantly, the presence of horizontal control, and control by potentially contextual signals, does not seem to render the device any less modular.

Informational encapsulation and domain specificity are both properties related to the input specificity of a device, so they will be considered together here under the rubric of input specificity, or input restriction. Regarding input specificity, two distinct issues are important: first, the distinction between *access specificity*, i.e. the breadth of information to which a device has access, and *processing specificity*, i.e. the breadth of information that a device actually processes; and second, the degree to which modules can be controlled or influenced horizontally or top-down by the outputs of other devices. Enzymes differ from Fodorean modules as conventionally construed because (a) they are access-general while processing-specific and (b) they are subject to rich horizontal control by other devices, and therefore sensitive to factors of context, etc., outside of their principal input domains.

7. Access Specificity vs. Processing Specificity

One reason why modular devices are widely considered to be poor candidates for central cognitive processes is because of the 'globality' of central processes. It is widely believed (though never proven) that central cognitive processes, in general, have access to *all* of the information in central knowledge stores. When trying to

figure out how to fix my leaking drain, for example, it feels like I can bring virtually any bit of knowledge I have to bear on the problem. A glance at Figure 1 shows why Fodorean modules appear to be poor candidates for central processes: their inputs are highly local. Clearly, devices that solve the routing problem by restricting their input pool will be unable to access all of the representations within an information store, a major problem for globality. The problem arises, however, not from the functional specificity of the devices, but how information is routed to them.

The enzyme model shows how processing specificity can be achieved even with complete access generality. Consider the snake detector and the face detector mentioned above. These devices have completely overlapping access specificity, because they both monitor the exact same pool of outputs of the object parser. This is not to say, however, that they are not functionally specific devices, nor that they admit the same inputs for processing. If one imagines each device to admit input on the basis of matching to a particular 'template', then neither actually processes information that the other processes. From a functional perspective, this is a very useful design: both have access, at least in principle, to the same information, even if each is selective about which information it actually processes.

As the example of the snake detector showed, for many kinds of information-processing systems with computational division of labor, i.e. for many systems composed of multiple SCDs, the best design for the system will be for the components of the system *not* to be access-restricted with respect to each other's inputs. Publicly accessible bulletin boards, with information in commonly processable formats to which many mechanisms have access (e.g. in this case, snake detectors, face detectors, artifact detectors, and so on), have design advantages that natural selection may have favored in many cases. The reason is that this design allows for the benefits of the computational division of labor to be reaped: the object parser does its job once, and many other systems can then exploit the output.

While not all computational systems need have a bulletin board design, many might. Access restriction can be unnecessary and / or disadvantageous when SCDs have precise input criteria, and are therefore able to 'decide for themselves' which information they process (they thus do not require 'meta-modules' in order to route information, a problem raised by Fodor in his critique of massive modularity). This reasoning fits very well with many of the arguments that Fodor (2000) raises about why central systems can't be modular. We know that thinking is 'flexible' or 'holistic' in that it can use information of diverse kinds and sources in reaching conclusions; from an evolutionary perspective, we have reason to believe that the ability to bring to bear many different kinds of information on a given problem would have been favored by selection.

One could imagine access-general enzymatic systems that were still hierarchical, i.e. that still had one-way flow: there could be, for example, layers of modular devices each of which scrutinized a common pool of outputs from the layer just below it. A central bulletin board design, on the other hand, has just a single pool,

from which inputs are drawn and into which outputs are deposited. This is access generality with a vengeance. This design allows for multiple processing (reprocessing) of the same representation (Figure 2), which is not possible in a vertical system in which processing is separated into horizontally segregated, one way processing streams (Figure 1). This is a very useful feature, because it allows for computational devices to leverage the work done by other devices within the same system. It also allows for the possibility of horizontal regulation of modules by other modules, either by altering representations in some way to which other devices are sensitive, or by emitting a system-wide signal. Each of these has distinct advantages. The open pool design allows for many kinds of context effects and interactive leveraging of computational power.

The advantages enjoyed by access-general systems come at a cost. In particular, when an information transfer system is entirely access-general, it can no longer use 'pipes' to guide information from one device to another. In the interstices between devices, information is in danger of getting lost, unless a solution can be found for coupling information of specific kinds to the devices capable of processing it, and ensuring that information is properly interpreted as it passes from device to device. Again, the enzyme model suggests a solution, in the form of tags that regulate how and when substrates are processed.

8. Semantic Tags, Scope Restriction, and Processing Specificity

From an adaptationist point of view, various properties of information might be expected to be preserved by cognitive operations, other properties altered, and others simply destroyed or left behind. One of the most fundamental properties of information that one might expect to be preserved is *reference*, or *aboutness*. Continuing with the example above, when a lion is encountered, information 'about' the lion will enter the brain through sensory input systems. It is important that, as this information passes through various computational procedures, it retains some identifier, some way for each subsequent processing subsystem that it encounters to 'know' that it is 'about a lion'. How might this be achieved by an enzymatic computational system?

Suppose that the object parser outputs an object representation, a package of information about size, shape, texture, distance, etc., all bound together as the representation of a single object, which then, as a substrate, floats around the common pool, colliding with a variety of recognition enzymes. As it contacts each of these enzymes, fit with the active site of each recognition enzyme is tested. When a positive match is found, processing occurs. For example, suppose that something in the object representation satisfies the input criteria of the lion recognition mechanism. This enzymatic processing system then catalyzes a particular reaction: it adds a 'tag' to the object substrate that identifies it as a lion—a LION tag (see Figure 8). This can be thought of as a *semantic tag*.

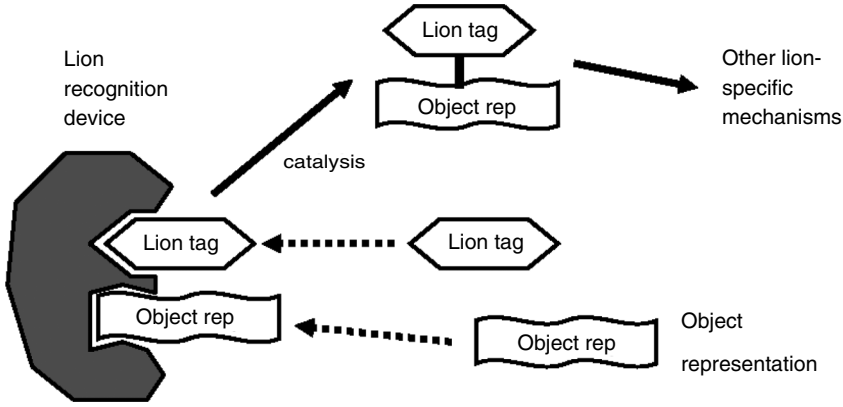

Figure 8 *Catalysis of a tagging reaction*

This design is useful in the context of truth preservation because it provides a solution to the 'preservation of aboutness' problem described above. Because of the importance of preserving aboutness, we expect semantic tags, once added, to be carried through many, if not all, computational processes that operate on information to which the tag is attached (although tags could also be removed in some cases; note also that tags need not be literally tags, just any consistent, conventionally recognized alteration of some portion of the representation).

What is the functional value of preserving aboutness information? Because the truth-preserving properties of computational procedures apply only to restricted classes of information, there must be a way of coupling information with procedures that will be truth preserving with respect to *it*. Semantic tags permit information to be coupled to computational procedures that are appropriate to it. A 'LION' tag, in effect, carries the information, 'Attention all procedures that can generate true inferences from information about lions: here's something for you'. A one-to-one input-output piping system is one, rather inelegant and inflexible way of solving the routing problem; such systems face the problem that mechanisms outputting lion information must 'know' where to send it. Semantic tags, however, bypass this problem. The tagged information is posted on the bulletin board for other mechanisms to make use of; the outputting device does not need to know in advance where to send the information.

An implication of the logic of semantic tags is that they should have causal properties: representations should be admitted to or excluded from processes on the basis of the tags they carry (rendering them 'syntactic' under Fodor's (2000) definition). The function that such tags serve has been called 'scope restriction' by Cosmides and Tooby (2000, p. 60): they 'regulate migration of information among subcomponents of the human cognitive architecture.' Not only can such tags *permit* information to be processed by procedures appropriate to it, they can *prevent* information from being processed by inappropriate procedures, i.e. procedures

that will not be truth preserving with respect to the class of information in question (the class that carries the tag). They are part of what Cosmides and Tooby call a scope syntax: semantic tags 'bound the scope' of processes to which tagged representations can be admitted. Without such devices, it would be difficult for evolved computational procedures to consistently maintain their truth preserving character.

In biochemical systems, scope restriction is achieved in similar ways, with tags, such as methyl groups, that permit or restrict the admission of substrates (e.g. DNA sequences) to certain enzymatic operations (e.g. transcription / translation). Tags can thereby provide a kind of *functional* input restriction, even in the absence of hard-wired piping. Like a child safety lock, a single tag can render a representation 'out of bounds' for a host of computational processes without establishing a wall or physical partition between the representation and those processes. This functional input restriction can, in turn, be instantly removed by removing the tag. This is not possible with a vertical Fodorean routing system of the kind depicted in Figure 1, in which the processing pathways that information can follow are pre-set. Fodor has suggested that modularity may be regarded as a property defined by the database that a device has access to. Tag-mediated input restriction suggests that the input pool of a device could in principle be changed on the fly, granting or denying access to a database simply by changing the set of representations to which a particular tag is attached, without altering the design of the device in any way.

9. Processing Cascades and the Computation of Higher-order Semantic Properties

If a tagging system is to function properly, the tags added by one set of computational devices should be used as input criteria by other computational devices (perhaps in combination with other properties of the representation to which they are attached). By adding a tag, a device helps to couple information to other procedures appropriate to information of that kind.

Evolutionary considerations suggest the possible existence of computational procedures specialized to operate on kinds of information relevant to semantic categories that seem rather abstract, such as predators, kin, mental states, social exchanges, and many more 'content' domains as well (Tooby and Cosmides, 1992). If such specialized computational procedures exist, natural selection must have found a way to couple them with the appropriate inputs. However, it seems unlikely that such coupling could be achieved by a one-step template matching system. As Fodor (2000) has pointed out, one cannot use simple cue detection to achieve such coupling for many higher-level semantic categories: for example, it seems unlikely that a single template could be engineered that could identify social exchanges, or mental states, solely from raw perceptual cues.

It is, however, possible to compute higher-level semantic categories through the serial operation of multiple computational processes, each of which computes a

semantic primitive (adds a semantic tag) that can then be used by other computational processes as input. If each computational device returns its output to the public representational pool, higher-order semantic categories can be computed.

Consider, for an example, how the semantic property PREDATOR might be computed from perceptual input, using an imaginary example. We could envision a three-step process. First, information from a lion passes through the object parsing system, and is deposited in the central pool. Next, this representation is matched with a perceptual template that adds a LION tag and returns it to the public representation pool. Finally, a third mechanism takes as input the representation with the LION tag—perhaps using something akin to a lookup table of animal tags that satisfy its input criteria—and adds a PREDATOR tag (see Figure 9).

Now this representation contains at least two tags, LION and PREDATOR (which are now attached to all of the information in the representation, such as where the object is, whether it is approaching, and so on). A multi-step process was required to compute the higher-order semantic category; addition of the PREDATOR tag would not have occurred without the prior addition of a LION tag. Once the PREDATOR tag is added to the representation, this tag can then be used to admit this information to various predator-specific computational procedures. A 'higher-order' or 'abstract' semantic category has been computed from raw sensory inputs. In principle, there is no reason a particular representation could not carry many, many tags. This is part of the benefit of bulletin-board style, unencapsulated representational systems. Note that not all tags used by an evolved computational system, nor the information used to compute them, need be 'innately specified'. For example, one can imagine semantic tags that are evolutionarily novel in their *specific* content (e.g. LION), but which are still tokens of a pre-existing semantic category of PREDATOR.

The use of processing chains to compute higher-order, abstract semantic categories—thereby coupling content-specific computational processes with appropriate input—probably occurs in many evolved computational systems. For example, evolutionary logic suggests that degree of relatedness through descent from a common ancestor—kinship—might be expected to be taken into account when making various decisions. Yet, degree of relatedness with an individual cannot be computed from a simple perceptual template. Rather, there must be multiple mechanisms that use a host of cues, such as co-residence during childhood, the way other people treat the individual in question, etc., to compute relatedness (Lieberman, Cosmides and Tooby, 2003).

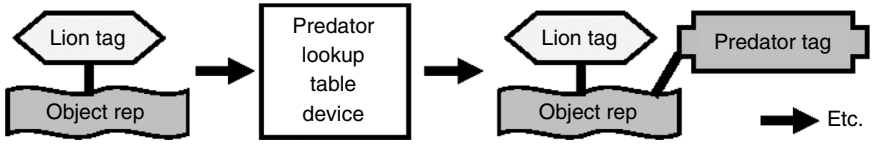

Figure 9 *Catalysis of a secondary tagging reaction*

This perspective leads one to expect richly tagged representations, which can be admitted to a variety of computational processes on the basis of having specific tags or even specific combinations of tags. One could also imagine multiple copies of a representation being spawned, with different combinations of tags (for example, it might make sense to preserve uncorrupted copies of representations using 'raw data' tags that block them from processing; Cosmides and Tooby, 2000). A given representation may have many tags, which admit it to computational procedures specialized for semantic categories of different kinds. For example, a given representation could simultaneously carry tags such as ABOUT-LION, ABOUT-OBJECT, ABOUT-INTENTIONAL-AGENT, etc. Each tag would admit the representation in question to some computational processes, and restrict it from others. Such tags might also be used to control the features of computational processes themselves, by acting as secondary substrates that alter the 'shape'/computational properties of an enzyme, thus controlling the computations being performed on the representation itself.

10. The Syntax / Semantics Problem

The notion that a system of tags might be used to couple information of specific content (semantic) types with particular content-specific computational procedures is different in many respects from the more traditional view of computational procedures as analogous to content-free, logical operations, which are specifically designed *not* to be sensitive to the content of the representations they operate on. Indeed, the observation that real-life thought processes do appear to be strongly sensitive to the content of representations has been used to argue that such processes could not be computational in the conventional sense (Fodor, 2000).

In Fodor's critique of the Massive Modularity Hypothesis, what he regards as perhaps the most substantial problem for computational systems of any kind is that they are sensitive only to the 'syntactic properties' of the representations they operate on (Fodor, 2000, p. 24). Turing machines are indeed insensitive to the content or meaning of the symbols they operate on, as are all of the 'truth preserving' operations of formal logic (modus ponens, etc.). As long as the input representations satisfy the 'well-formedness' (syntactic) rules of any computational system, they will be processed according to the rules of the system (for example, 'p' in a representation admitted to a modus ponens algorithm needs to be in the form of a proposition; p can't be, for example, a pixel bitmap or a 21/2 D sketch). Consequently, a property of the algorithms of formal logic is that the semantic properties of the representations they operate on are invisible to them. Although a syllogism machine that is fed false premises may output false conclusions, as far as the machine is concerned, 'truth' has been preserved, in the sense that it has obeyed the truth-preserving rules that it was designed to obey.

The problem is that cognitive processes *are*, in fact, sensitive to the meaning of the representations they process—notoriously so. The results of such sensitivities

are known in cognitive psychology as 'content effects' (Cosmides, 1989; Griggs and Cox, 1982). A classic example is the Wason selection task, which was originally designed to test just the kind of content-independent computational abilities that one might expect people to have if logical principles were instantiated in the laws of thought. The evidence suggests that they aren't.

In the Wason task, subjects are given a rule of the form *if P, then Q*, a series of cards showing on one side whether P is true and on the other side whether Q is true, and asked to turn over whatever cards are necessary to determine if the rule is broken. In contrast to what one would predict if subjects used the laws of formal logic, the pattern of cards turned over depends heavily on the content of P and Q. For example, if the rule is 'if there is a vowel on one side of the card [P], then there is an even number on the other side [Q]', the pattern of cards subjects turn over, i.e. which of the cards *P, not P, Q*, and *not Q* they select, tends to be different than if the rule is 'if you are drinking beer [P], then you must be over 21 [Q]' (Cosmides, 1989; Griggs and Cox, 1982).

Why is this a problem? For Fodor, it is a problem because the rule that is used to determine which cards to turn over should be sensitive only to the *syntactic* properties of the Ps and Qs it is operating on. Recall that one test for whether a property is syntactic or not is substitutability: can you swap out one P for a different P, and still have a well-formed representation? If one considers two well-formed statements of the form *if P, then Q*, one can see that this is, in principle, possible:

(A) If *there is a vowel on one side of the card*, then *there is an odd number on the other side of the card*.
(B) If *you give me ten dollars*, then *I will give you my watch*.

One can, in fact, mix and match the Ps and Qs in some ways and still have an intelligible if-then statement, for example: If *there is a vowel on one side of the card*, then *I will give you my watch*. The intelligibility of this suggests that there are at least some mechanisms for which *there is an odd number on the other side of the card* and *I will give you my watch* are substitutable, and, therefore, are syntactically identical. But the substitutability test can be a test in the other direction as well. In fact, we can state this as a kind of deduction rule:

> When replacing P_1 with P_2 in a representation causes the representation to be processed differently by a particular procedure, P_1 and P_2 are not syntactically substitutable/identical with respect to *that* procedure.

In other words, while sentences (A) and (B) above may appear to have the same syntactic form to some procedure (e.g. a procedure for detecting *grammatical* well-formedness), this does not mean that they necessarily have the same form for *all* procedures. This is what Cosmides (1989) proposes: that *I will give you my watch* is a kind of representational element that is admitted to a cheater detection algorithm, whereas *there is an odd number on the other side of the card* is not. For formal logic, P

and Q are syntactic categories that can take any propositions as input, but for whatever logic the mind is using, this is not true—the tags that P and Q carry influence how they are processed.

While Fodor is aware of Cosmides' claim, he is skeptical that a representation such as *I will give you my watch* could be 'marked' in some way that places it in a different syntactic category, for some mechanisms, than *there is an odd number on the other side of the card*. This is because he assumes that the inputs to a cheater detection mechanism would have to be purely perceptual (this is entailed by his equation of modular devices with input systems), and therefore that the device would need to rely on some kind of perceptual cue to 'decide whether what it's looking at *is* a social exchange' (Fodor, 2000, p. 75). Because there are unlikely to be simple perceptual cues that could reliably pick out social exchanges and only social exchanges, he cannot envision such a device. As he puts it, even if social exchanges in some ancestral environment were 'orange with gray stripes', they are not likely to be now. 'So the massive modularity thesis can't be true unless there is, inter alia, a module that detects the relevant Subtle Cues and infers from them that a social exchange is going on' (Fodor, 2000, p. 76).

But modules need not accept *only* perceptual cues as data; they can leverage the prior operations of other devices, and combine perceptual with contextual information. The use of semantic tags can in principle solve the problem of how 'orange color' is added to a representation: just as Fodor says, there may be, 'inter alia, a module that detects the relevant Subtle Cues and infers from them that a social exchange is going on'. Or, more likely, there might be many modules, each of which *adds* a tag to a representation on the basis of certain properties. Among the requirements for admission to the cheater detection mechanism, the representation must be tagged as being a social transaction, and as involving costs and benefits; the final set of tags could be assembled by devices each of which computes only a part of the necessary information (Figure 10). That such prior tagging is crucial is suggested by the experimental observation that the contextual story in which the rule is presented can cause subjects to

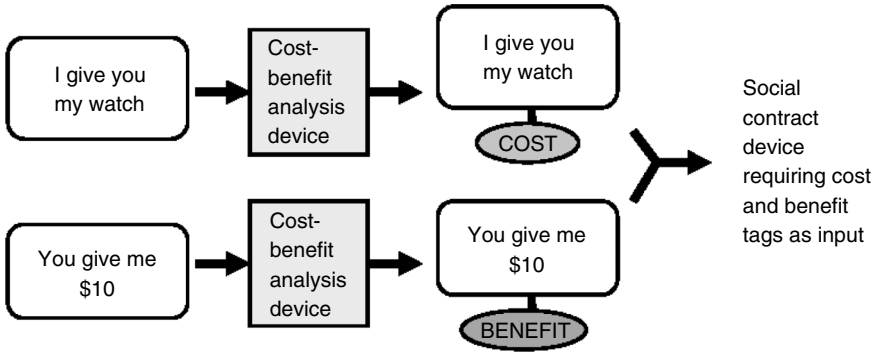

Figure 10 *Enzymatic computation of social contract primitives*

interpret the *same* rule as either a social contract or a 'descriptive' (non-social-contract) rule. In real life, I might hand you my watch as an offer to sell it, or to hold it while I swim; the interpretation of the same percept will depend on context and prior knowledge. This makes sense from a design perspective. The social contract system is designed to pick out events that are defined by abstract social properties, not perceptual ones.

11. Globality, Abduction, and Catalytic Equilibrium

We are now in the realm of 'higher' cognition, which, unlike perceptual processes, seems quite flexible, and unlike the 'computational reflexes' that are said to result in such things as optical illusions (Fodor, 1983; Pylyshyn, 1986). There is widespread resistance to the notion that the kinds of semantic processes described above could be handled by modular systems. For example, the fact that a given sentence has so many shades of meaning and leads to so many possible inferences is taken by many as proof itself that no modular system could be responsible for these effects. Fodorean systems do indeed seem incapable of accounting for the global and flexible properties of higher cognition, given their narrowly restricted input pools and the impossibility of horizontal and top-down information transfer. In particular, Fodor (2000) suggests that the phenomenon known as abductive inference, or 'inference to the best explanation', cannot be accounted for by *any* computational system, because a wide and diverse array of facts might bear on what the best explanation is. Facts must percolate through the entire system, and there is no way of deciding in advance which facts are relevant. The problem, as Fodor sees it, is that modular systems are 'computationally local', whereas phenomena like abduction are 'computationally global'. Information does not spread throughout Fodorean systems, but remains only in the relevant processing stream.

Although specific solutions to problems such as abduction are not proposed here, several features of enzymatic modular systems may render them better candidates than Fodorean systems for solving problems like abduction, global belief revision, and context effects on inference. First, percolation (diffusion, and diffusion with catalysis) is possible: there is nothing to prevent the same representation from passing through many different computational procedures, and a given representation may be compared to many beliefs (i.e. other representations in the system) simultaneously (in this sense, enzymatic systems are parallel processing systems). Second, unlike Fodorean systems, enzymatic systems allow for the possibility of feedback, including both positive feedback (e.g. amplification) and negative feedback (e.g. inhibition). For example, the bucket brigade algorithm of Holland *et al.* (1986) suggests a means of reinforcing hypotheses consistent with evidence. Third, reprocessing means that tagging of a representation by one procedure can affect how it is then processed by other procedures; the same piece of information can lead to different inferences depending on how it has been previously tagged (which in turn can depend on the context in which it is

presented). Searle (2002), for example, suggests that no rule-based (computational) system could account for the difference in how the verb 'cut' is understood in 'cut the grass' and 'cut the cake', nor why the sentences 'cut the sun' and 'cut the mountain' violate our intuitions. But one can imagine a system for tagging different substance kinds that would lead to 'Cut the grass' and 'Cut the cake' being admitted to different inferential processes, and 'Cut the sun' and 'Cut the mountain' being blocked because of lack of substance tags.

Unlike Fodorean systems, in enzymatic computational systems every device in the system has access, in principle, to every representation in the system, and therefore, can in principle leverage the inferential power of every other device in the system. In a system where information is allowed to propagate through all relevant inference devices, all inferences that the system is capable of generating, given its current set of rules and its complete knowledge database, *will* be generated. One might think of this kind of global process—representations seeping through the system by diffusion and generating all possible inferences as they go—as the system going to catalytic equilibrium. Abduction is, in a sense, the reverse of this process—taking a diverse set of facts and back-generating the single fact which, when passed through many inference systems, would produce these diverse inferences. It seems at least intuitively plausible that these processes could be something akin to reaching catalytic equilibrium in a massively parallel inference system. The scope of propagation of new facts can in turn be governed by tags that regulate which pieces of information are allowed to interact. For example, something like a 'pretense' tag could prevent counterfactual information that is being entertained from corrupting true beliefs (Cosmides and Tooby, 2000). While the details remain to be worked out, catalytic processes appear to be more promising candidates for handling phenomena such as belief revision than are Fodorean ones.

12. Conclusion

Many features of the enzyme model presented here are not new in cognitive science. For example, Selfridge's (Selfridge and Neisser, 1960) pandemonium model, originally intended to be a model of perceptual processes, involves many independent devices or 'demons' that scrutinize a common database, or blackboard, of information, and report in parallel, and competitively, to higher-level demons that make decisions on the basis of their collective output. The classifier system model (Holland *et al.*, 1986) also uses a bulletin board design, a system of tags, and further innovations such as the bucket brigade algorithm as a means of inferential feedback. Nevertheless, widespread skepticism about the possibility of higher cognitive processes being handled by specialized devices persists. One major reason for this, I have argued here, is that the success of Fodor's model of modularity has impaired peoples' ability to imagine other possible architectures. Add to this the widespread belief that endorsing modularity implies endorsing large numbers of complicated devices specified in every detail by large numbers of gene loci (which it doesn't), and people are prone to accept out of hand Fodor's claim that central processes can't be modular, and can't even be computational.

Fodor's argument is a case where the conclusion reached—that thinking can't be computational even in principle—should make us highly suspicious that at least one of the premises is wrong. What is likely to be incorrect is the premise that all systems composed of specialized devices must be Fodorean modular systems. Evolutionary logic alone makes this assumption suspect. Because Fodorean modules have poor design features for solving problems faced by central cognitive processes, we should expect that the devices that handle central cognition *don't* have those features. Moreover, we can use adaptationist design logic to generate hypotheses about computational designs that could solve those problems. In particular, central cognitive systems face problems of information routing, and the integration of information from diverse sources, that are not faced by peripheral systems that operate only on highly localized inputs. At the same time, just as perceptual mechanisms gain inferential power by instantiating specialized routines tailored to the kinds of information they are designed to process, so might central inference mechanisms. Putting these two observations together can be a powerful source of insight about the possible design features of central systems. Unfortunately, many prefer to be persuaded by Fodor's argument that it is better to throw up one's hands and declare central processes a mystery.

The enzyme model is offered not only as an existence proof, i.e. that non-Fodorean modular systems are possible and exist, but also, as a step towards applying the logic of adaptationism to understanding information handling within the mind. Just as engineers of the world-wide web have developed systems of tags and routing for the control of information in vastly parallel, open systems, and search engines that perform abduction-like global inferences quickly, so natural selection might have designed similar systems for the handling of information in the brain. There is nothing about the problem that implies that the devices to which the information is being routed are not highly specialized.

Finally, it is worth noting that analogies to real biological systems like enzymes might offer different insights than analogies to artificial systems such as the CPU architectures of modern computers. In the long run, natural selection is much more innovative at finding solutions to problems than are people. To date, the devices imagined in cognitive science have been constrained by the properties of computers that we ourselves have built. But we may not have begun to imagine the devices that natural selection has created. They are bound to be more ingenious and stranger than anything we have yet invented ourselves.

Center for Behavior, Evolution, and Culture
Department of Anthropology, UCLA

References

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J. D. 1994: *Molecular Biology of the Cell, 3rd Ed.* New York: Garland.

Buller, D. J. and Hardcastle, V. G. 2000: Evolutionary psychology, meet developmental neurobiology: Against promiscuous modularity. *Brain and Mind*, 1, 307–325.

Coen, E. 1999: *The Art of Genes: How Organisms Make Themselves*. New York: Oxford University Press.

Coltheart, M. 1999: Modularity and cognition. *Trends in Cognitive Sciences*, 3, 115–120.

Cosmides, L. 1989: The logic of social exchange: Has natural selection shaped how humans reason? Studies with the Wason selection task. *Cognition*, 31, 187–276.

Cosmides, L. and Tooby, J. 2000: Consider the source: The evolution of mechanisms for decoupling and metarepresentation. In D. Sperber (ed.), *Metarepresentation*. New York: Oxford University Press.

Fodor, J. 1983: *The Modularity of Mind*. Cambridge: MIT Press.

Fodor, J. 2000: *The Mind Doesn't Work That Way: The Scope and Limits of Computational Psychology*. Cambridge: MIT Press.

Griggs, R.A. and Cox, J.R. 1982: The elusive thematic-materials effect in Wason's selection task. *British Journal of Psychology*, 73, 407–420.

Holland J. H., Holyoak K. J., Nisbett R. E. and Thagard P. A. 1986: *Induction: Processes of Inference, Learning, and Discovery*. Cambridge: MIT Press.

Kurzban, R. 1996: Ontological filters: How evolution solves the chicken and egg problem. Unpublished Master's Thesis, University of California Santa Barbara.

Lieberman, D., Tooby, J. and Cosmides, L. 2003: Does morality have a biological basis? An empirical test of the factors governing moral sentiments relating to incest. *Proceedings of the Royal Society of London, Series B*, 270, 819–826.

Magnasco, M. O. 1997: Chemical kinetics is Turing universal. *Physical Review Letters*, 78, 1190–1193.

Marcus, G. F. 2001: *The Algebraic Mind: Integrating Connectionism and Cognitive Science*. Cambridge: MIT Press.

Marr, D. 1982: *Vision*. New York: H. Freeman and Co.

Pinker, S. 1997: *How the Mind Works*. New York: Norton.

Pylyshyn, Z. 1986: *Computation and Cognition*. Cambridge: MIT Press.

Pylyshyn, Z.W. 1999: Is vision continuous with cognition? The case for cognitive impenetrability of visual perception. *Behavioral and Brain Sciences*, 22, 341–423.

Samuels, R. 1998: Evolutionary psychology and the massive modularity hypothesis. *British Journal for the Philosophy of Science*, 49, 575–602.

Samuels, R., Stich, S. and Tremoulet, P. D. 1999: Rethinking rationality: From bleak implications to Darwinian modules. In E. LePore and Z. Pylyshyn (eds.), *Rutgers University Invitation to Cognitive Science*. New York: Blackwell.

Searle, J R. 2002: 'Sneaked' or 'snuck'? *New York Review of Books*, March 14, 2002.

Segal, G. 1996: The modularity of theory of mind. In P. Carruthers and P. Smith (eds.), *Theories of Theories of Mind*. New York: Cambridge University Press.

Selfridge, O. G. and Neisser, U. 1960: Pattern recognition by machine. *Scientific American*, 203, 60–68.

Sperber, D. 1994: The modularity of thought and the epidemiology of representations. In L. A. Hirschfeld and S. A. Gelman (eds.), *Mapping the Mind: Domain Specificity in Cognition and Culture*. New York: Cambridge University Press.

Tooby, J. and Cosmides, L. 1992: The psychological foundations of culture. In J H. Barkow, L. Cosmides and J. Tooby (eds.), *The Adapted Mind: Evolutionary Psychology and the Generation of Culture*. New York: Oxford University Press.