

Creativity and computers*

Margaret A. Boden

School of Computing and Cognitive Sciences, University of Sussex, Falmer, Brighton BN1 9QH, UK

What is creativity? One new idea may be *creative*, while another is merely *new*: what's the difference? And how is creativity possible? Artists and scientists rarely know how their original ideas come about. They mention intuition, but cannot say how it works. What's more, many people assume that there will never be a scientific theory of creativity—for how could science possibly explain fundamental novelties? The very notion seems to be a contradiction in terms. But computational ideas are helping us to understand how human originality is possible. We can now say something specific about how intuition works and how the human mind can seem to surpass itself.

The problem

Creativity is a puzzle, a paradox, some say a mystery. Artists and scientists rarely know how their original ideas come about. They mention intuition, but cannot say how it works. Most psychologists cannot tell us much about, it, either. What's more, many people assume that there will never be a scientific theory of creativity—for how could science possibly explain fundamental novelties?

As for computers, these are commonly believed to lie right at the opposite end of the philosophical spectrum. Surely, they can have nothing whatever to do with creativity? This opinion was first expressed over a hundred years ago, by Ada, Lady Lovelace (the friend and collaborator of Charles Babbage).

Lady Lovelace realized that Babbage's 'Analytical Engine'—in essence, a design for a digital computer—could, in principle, 'compose elaborate and scientific pieces of music of any degree of complexity or extent' (quoted in ref. 1, see also ref. 2). But she insisted that the creativity involved in any elaborate pieces of music emanating from the Analytical Engine would have to be credited not to the engine, but to the engineer. As she put it: 'The Analytical Engine has no pretensions whatever to originate anything. It can do [only] whatever we know how to order it to perform'.

If Lady Lovelace's remark means merely that a computer can do only what its program enables it to do, it is correct—and, from the point of view of theoretical

psychology, important. It means, for instance, that if a program manages to play a Chopin waltz expressively, or to improvise modern jazz, then the musical structures and procedures in that program *must* be capable of producing those examples of musical expression or improvisation. It does not follow that human musicians do it in the same way: perhaps there is reason to suspect that they do not. But the program specifies, in detail, *one* way in which such things can be done. Alternative theories, involving different musical structures or psychological processes, should ideally be expressed at a comparable level of detail.

But if Lady Lovelace's remark is intended as an argument denying any interesting link between computers and creativity, it is too quick and too simple. We must distinguish four different questions, which are often confused with each other. I call them Lovelace-questions, because many people would respond to them (with a dismissive 'No!') by using the argument cited above.

The first Lovelace-question is whether computational concepts can help us understand how *human* creativity is possible. The second is whether computers (now or in the future) could ever do things which at least *appear to be* creative. The third is whether a computer could ever appear to *recognize* creativity—in poems written by human poets, for instance, or in its own novel ideas about science or mathematics. And the fourth is whether computers themselves could ever *really* be creative (as opposed to merely producing apparently creative performance whose originality is wholly due to the human programmer).

The psychologist's interest is mainly in the first Lovelace-question, which focusses on creativity in *people*. The next two Lovelace-questions are psychologically interesting insofar as they throw light on the first.

The answers I shall propose to these three questions are, respectively: *Yes, definitely*; *Yes, up to a point*; and *Yes, necessarily (for any program which appears to be creative)*. In short, computational ideas can help us to understand how human creativity is possible. This does not mean that creativity is predictable, nor even that an original idea can be explained in every detail after it has appeared. But we can draw on computational ideas in understanding in scientific terms how 'intuition' works.

For psychological purposes, the fourth Lovelace-question is less important than the other three. It is not

*Text based on a Friday Discourse delivered at the Royal Institution of Great Britain.

a scientific question, as they are, but—as we shall see—a disguised request for a moral-political decision.

Defining creativity

Why does creativity seem so mysterious? To be sure, artists and scientists typically have their creative ideas unexpectedly, with little if any conscious awareness of how they arose. But the same applies to much of our vision, language, and common-sense reasoning. Psychology, including computational psychology (which uses theoretical concepts drawn from artificial intelligence, or AI), includes many theories about unconscious processes. Creativity is mysterious for another reason: the very concept is seemingly paradoxical.

If we take seriously the dictionary-definition of creation, 'to bring into being or form out of nothing', creativity seems to be not only beyond any scientific understanding, but even impossible. It is hardly surprising, then, that some people have 'explained' it in terms of divine inspiration, and many others in terms of some romantic intuition, or insight. From the psychologist's point of view, however, 'intuition' is the name not of an answer, but of a question. How does intuition work?

People of a scientific cast of mind generally try to define creativity in terms of 'novel combinations of old ideas', where the surprise caused by a 'creative' idea is due to the improbability of the combination.

Admittedly, the novel combinations have to be valuable in some way, because to call an idea creative is to say that it is not only new, but interesting. Combination-theorists typically omit value from their definition of creativity, perhaps because they (mistakenly) take it for granted that unusual combinations are always interesting. Also, they often fail to explain how it was possible for the novel combination to come about. They take it for granted, for instance, that we can associate similar ideas or recognize more distant analogies, without asking just how such feats are possible.

These cavils aside, what is wrong with the combination-theory? Many ideas—concepts, theories, paintings, poems, music—which we regard as creative are indeed based on unusual combinations. For instance, part of the appeal of the Lennon-McCartney arrangement of *Yesterday* was their use of a cello, an instrument normally associated with music of a very different kind; and poets often delight us by juxtaposing seemingly unrelated concepts. Many creative ideas, however, are surprising in a deeper way. They concern novel ideas which not only *did* not happen before, but which—in a sense that must be made clear—*could* not have happened before.

Before considering just what this 'could not' means,

we must distinguish two senses of *creativity*. One is psychological (let us call it P-creativity), the other historical (H-creativity). An idea is P-creative if the person in whose mind it arises could not have had it before; it does not matter how many times other people have already had the same idea. By contrast, an idea is H-creative if it is P-creative *and* no-one else has ever had it before.

H-creativity is something about which we are often mistaken. Historians of science and art are constantly discovering cases where other people, even in other periods, have had an idea popularly attributed to some individual hero. Whether an idea survives, whether it is lost for a while and resurfaces later, and whether historians at a given point in time happen to know about it, depend on a wide variety of unrelated factors. These include fashion, rivalries, illness, trade-patterns, economics, war, flood, and fire. It follows that there can be no systematic explanation of H-creativity.

Certainly, there can be no *psychological* explanation of this (historical) category. But all H-creative ideas, by definition, are P-creative too. So a psychological explanation of P-creativity would include H-creative ideas as well.

What does it mean to say that an idea 'could not' have arisen before? Unless we know that, we cannot make sense of P-creativity (or H-creativity either), for we cannot distinguish radical novelties from mere 'first-time' newness.

An example of a novelty which clearly *could* have happened before is a newly-generated sentence, such as 'The pineapples are in the bathroom-cabinet, next to the oil-paints that belonged to Savonarola'. I have never thought of that sentence before, and almost certainly no-one else has, either.

The linguist Noam Chomsky remarked on this capacity of language-speakers to generate first-time novelties endlessly, and he called language 'creative' accordingly. His stress on the infinite fecundity of language was correct, and highly relevant to our topic. But the word 'creative' was ill-chosen. Novel though the sentence about Savonarola's oil-paints is, there is a clear sense in which it *could* have occurred before. For it can be generated by the same rules that can generate other English sentences. Any competent speaker of English could have produced that sentence long ago—and so could a computer, provided with English vocabulary and grammatical rules. To come up with a new sentence, in general, is not to do something P-creative.

The 'coulds' in the previous paragraph are *computational* 'coulds'. In other words, they concern the set of structures (in this case, English sentences) described and/or produced by one and the same set of generative rules (in this case, English grammar).

There are many sorts of generative system: English

grammar is like a mathematical equation, a rhyming-schema for sonnets, the rules of chess or tonal harmony, or a computer program. Each of these can (timelessly) describe a certain set of structures. And each might be used, at one time or another, in actually producing those structures.

Sometimes, we want to know whether a particular structure could, in principle, be described by a specific schema, or set of abstract rules.—Is '49' a square number? Is 3,591,471 a prime? Is this a sonnet, and is that a sonata? Is that painting in the Impressionist style? Could that geometrical theorem be proved by Euclid's methods? Is that word-string a sentence? Is a benzene-ring a molecular structure that is describable by early nineteenth-century chemistry (before Friedrich von Kekule's famous fireside daydream of 1865)?—To ask *whether an idea is creative or not* (as opposed to how it came about) is to ask this sort of question.

But whenever a particular structure is produced in practice, we can also ask what generative processes actually went on in the computational system concerned.—Did a human geometer (or a program) prove a particular theorem in this way, or in that? Was the sonata composed by following a textbook on sonata-form? Did Kekule rely on the then-familiar principles of chemistry to generate his seminal idea of the benzene-ring, and if not how did he come up with it?—To ask how an idea (creative or otherwise) *actually arose*, is to ask this type of question.

We can now distinguish first-time novelty from radical originality. A merely novel idea is one which can be described and/or produced by the same set of generative rules as are other, familiar, ideas. A genuinely original, or creative, idea is one which cannot. So *constraints*, far from being opposed to creativity, make creativity possible. To throw away all constraints would be to destroy the capacity for creative thinking. Random processes alone can produce only first-time curiosities, not radical surprises (although randomness can sometimes contribute to creativity).

To justify calling an idea creative, then, one must specify the particular set of generative principles with respect to which it is impossible. Accordingly, psychologists can learn from literary critics, musicologists, and historians of art and science. But their (largely tacit) knowledge of the relevant structures must be made as explicit as possible.

The psychology of creativity can benefit from AI and computer science precisely because—as Lady Lovelace pointed out—a computer can do only what its program enables it to do. On the one hand, computational concepts help us to specify generative principles clearly. On the other hand, computer-modelling helps us to see, in practice, what a particular generative system can and cannot do. The results may

be surprising, for the generative potential of a program is not always obvious: the computer may do things we did not know we had 'ordered it' to perform.

It follows from all this that, with respect to the usual mental processing in the relevant domain (chemistry, poetry, music, ...), a creative idea is not just improbable, but *impossible*. How did it arise, then, if not by magic? And how can one impossible idea be more surprising, more creative, than another? If the act of creation is not mere combination, or what Arthur Koestler called 'the bisociation of unrelated matrices'³, what is it? How can creativity possibly happen?

Exploring and transforming conceptual spaces

A generative system defines a certain range of possibilities: chess-moves, for example, or jazz-melodies. These structures are located in a conceptual space (what computer scientists would call a search-space) whose limits, contours, and pathways can be mapped in various ways. Mental maps, or representations, of conceptual spaces can be used (not necessarily consciously) to explore the spaces concerned.

When Dickens described Scrooge as 'a squeezing, wrenching, grasping, scraping, clutching, covetous old sinner', he was exploring the space of English grammar. He was reminding us (and himself) that the rules of grammar allow us to use any number of adjectives before a noun. Usually, we use only two or three; but we may, if we wish, use seven (or more). That possibility already existed, although its existence may not have been realized by us.

A more interesting, more complex, example of exploration can be found in the development of post-Renaissance Western music. This music is based on the generative system known as tonal harmony.

Each piece of tonal music has a 'home key', from which it starts, from which (at first) it did not stray, and in which it must finish. Reminders and reinforcements of the home key were provided, for instance, by fragments of scales decorating the melody, or by chords and arpeggios within the accompaniment. As time passed, the range of possible home keys became increasingly well-defined. Johann Sebastian Bach's 'Forty-eight', for example, was a set of preludes and fugues specifically designed to explore—and clarify—the tonal range of the well-tempered keys.

But travelling along the path of the home key alone became insufficiently challenging. Modulations between keys were then allowed, within the body of the composition. At first, only a small number of modulations (perhaps only one, followed by its 'cancellation') were tolerated, between strictly limited pairs of harmonically related keys. Over the years, however, the modulations became increasingly daring, and increas-

ingly frequent until in the late nineteenth century there might be many modulations within a single bar, not one of which would have appeared in early tonal music. The range of harmonic relations implicit in the system of tonality gradually became apparent. Harmonies which would have been unacceptable to the early musicians, who focussed on the most central or obvious dimensions of the conceptual space, became commonplace.

Moreover, the notion of the home key was undermined. With so many, and so daring, modulations within the piece, a 'home key' could be identified not from the body of the piece, but only from its beginning and end. Inevitably, someone (it happened to be Arnold Schoenberg) eventually suggested that the convention of the home key be dropped altogether, since it no longer made sense in terms of constraining the composition as a whole. (Significantly, Schoenberg suggested various new constraints to structure his music-making: using every note in the chromatic scale, for instance.)

Another example of extended exploration was the scientific activity spawned by Mendeleyev's Periodic Table. This table, produced in the 1860s for an introductory chemistry textbook, arranged the elements in rows and columns according to their observable properties and behaviour. All the elements within a given column were in this sense 'similar'. But Mendeleyev left gaps in the table, predicting that unknown elements would eventually be found with the properties appropriate to these gaps (no known element being appropriate).

Sure enough, in 1879 a new element (scandium) was discovered whose properties were what Mendeleyev had predicted. Later, more elements were discovered to fill the other gaps in the table. And later still, the table (based on observable properties) was found to map onto a classification in terms of atomic number. This classification explained why the elements behaved in the systematic ways noted by Mendeleyev.

However, exploring a conceptual space is one thing; transforming it is another. What is it to transform such a space?

One example has been mentioned already: Schoenberg's dropping the home-key constraint to create the space of atonal music. Dropping a constraint is a general heuristic, or method, for transforming conceptual spaces.

Non-Euclidean geometry, for instance, resulted from dropping Euclid's fifth axiom, about parallel lines meeting at infinity. (One of the mathematicians responsible was Lobachevsky, immortalized not only in encyclopaedias of mathematics but also in the songs of Tom Lehrer.) This transformation was made 'playfully', as a prelude to exploring a geometrical space somewhat different from Euclid's. Only much later did it turn out

to be useful in physics.

Another very general way of transforming conceptual spaces is to 'consider the negative': that is, to negate a constraint. One well-known instance concerns Kekule's discovery of the benzene-ring. He described it like this (quoted in ref. 4, p 39):

I turned my chair to the fire and dozed. Again the atoms were gambolling before my eyes. ... (My mental eye) could distinguish larger structures, of manifold conformation; long rows, sometimes more closely fitted together, all twining and twisting in snakelike motion. But look! What was that? One of the snakes had seized hold of its own tail, and the form whirled mockingly before my eyes. As if by a flash of lightning I awoke.

This vision was the origin of his hunch that the benzene-molecule might be a ring, a hunch which turned out to be correct.

Prior to this experience, Kekule had assumed that all organic molecules are based on strings of carbon atoms (he himself had produced the string-theory some years earlier). But for benzene, the valencies of the constituent atoms did not fit.

We can understand how it was possible for him to pass from strings to rings, as plausible chemical structures, if we assume three things. First, that snakes and molecules were already associated in his thinking. Second, that the topological distinction between open and closed curves was present in his mind. And third, that the 'consider the negative' heuristic was present also. Taken together, these three factors could transform 'string' into 'ring'.

(Kekule tells us himself that the first of these assumptions is correct; and recent computational work on 'connectionist' systems, or 'neural nets', is helping us to understand how such associations are possible. As for the other two assumptions, there is independent psychological evidence that they are true of human minds in general.)

A string-molecule is what topologists call an open curve. Topology is a form of geometry which studies not size or shape, but neighbour-relations. An open curve has at least one end-point (with a neighbour on only one side), whereas a closed curve does not. An ant crawling along an open curve can never visit the same point twice, but on a closed curve it will eventually return to its starting-point. These curves need not be curvy in shape. A circle, a triangle, and a hexagon are all closed curves; a straight line, an arc, and a sine-wave are all open curves.

If one considers the negative of an open curve, one gets a closed curve. Moreover, a snake biting its tail is *a closed curve which one had expected to be an open one*. For that reason, it is surprising, even arresting ('But look! What was that?'). Kekule might have had a similar reaction if he had been out on a country walk and

happened to see a snake with its tail in its mouth. But there is no reason to think that he would have been stopped in his tracks by seeing a Victorian child's hoop. A hoop is a hoop, is a hoop: no topological surprises there.

Finally, the change from open curves to closed ones is a topological change, which by definition will alter neighbour-relations. And Kekule was an expert chemist, who knew very well that the behaviour of a molecule depends not only on what the constituent atoms are, but also on how they are juxtaposed. A change in atomic neighbour-relations is very likely to have some chemical significance. So it is understandable that he had a hunch that this tail-biting snake-molecule might contain the answer to his problem.

Computer programs and the arts

Many artists use computers as tools, to help them create things they could not have created otherwise. So-called 'computer music', for instance, may use sounds which no orchestra could produce. A visual artist may get ideas from computer graphics. And even the humble word-processor plays a part in many literary efforts. But these examples are of little interest here.

We are concerned with those programs which produce aesthetically interesting creations themselves, or which (in their attempts to do so) throw light on the psychological processes underlying human art. There are a number of programs which explore artistically interesting spaces, and a few which produce aesthetically acceptable results. As yet, however, there is no 'artistic' program which transforms its space in significant ways.

For example, Harold Cohen (already a well-known painter when he started working with computers) has written a series of programs which produce pleasing, and unpredictable, line-drawings⁵. (I have one in my office, and on several occasions a visitor has spontaneously remarked 'I like that drawing! Who did it?') These have been exhibited at the Tate and other major art-galleries around the world, and not just for their curiosity-value.

Each of Cohen's programs explores a certain style of line-drawing and a certain subject-matter. The program may draw acrobats with large beach-balls, for instance, or human figures in the profuse vegetation of a jungle. (As yet, he has not written a colouring-program which satisfies him; meanwhile, he sometimes colours his programs' drawings by hand.)

Much as human artists have to know about the things they are depicting, so each of Cohen's programs needs an internal model of its subject-matter. This model is not a physical object, like the articulated wooden dolls found in artists' studios, but a generative system. It is a set of abstract rules which specify, for

instance, not only the anatomy of the human body (two arms, two legs), but also how the various body-parts appear from various points of view. An acrobat's arm pointing at the viewer will be foreshortened; a flexed arm will have a bulging biceps; and an arm lying behind another acrobat's body will be invisible.

The program can *draw* acrobats with only one arm visible (because of occlusion), but it cannot draw one-armed acrobats. Its model of the human body does not allow for the possibility of there being one-armed people. If it were capable of 'dropping' one of the limbs (as a geometer may drop Euclid's fifth axiom), it could then draw one-armed, and one-legged, figures. But the resulting pictures might not be so plausible, nor so pleasing. The reason is that its current world-model contains rules dealing with human stability and picture-balance, some of which may implicitly or explicitly assume that all people have four limbs. If so, a three-limbed person (assuming one limb were 'dropped') might be drawn in an impossible bodily attitude. Human artists drawing a one-armed person would not do this, unless they were deliberately contravening the laws of gravity (as in a Chagall dreamscape).

The psychological interest of Cohen's work is that the constraints—*anatomical, physical, and aesthetic*—written into his programs are perhaps a subset of those which human artists respect when drawing in comparable styles. A host of questions arise about just what those constraints may be. And a host of issues can be explored by building additional or alternative rules into Cohen's programs, and examining the range of structures that result.

However, Cohen's programs are like hack-artists, who can draw only in a given style. The style may be rich enough (the generative system powerful enough) to make their drawings individually unpredictable. But the style itself is easily recognized.

At present, only Cohen can change the constraints built into the program, so enabling it to draw pictures of a type which it could not have drawn before. But some programs, perhaps including some yet to be written by Cohen, could do so for themselves.

To be able to transform its style, a program would need (among other things) a meta representation of the lower-level constraints it uses. For instance, if it had an explicit representation of the fact that it normally draws four-limbed people, and if it were given very general 'transformation heuristics' (like 'drop a constraint' and 'consider the negative'), it might sometimes omit one or more limbs. Recent evidence from developmental psychology^{6,7} suggests that this sort of explicit representation of a lower-level drawing-skill is required if a young child is to be able to draw a one-armed man, or a seven-legged dog. (Comparable evidence suggests that flexibility in other skills, including language, also



Figure 1



Figure 2



Figure 3



Figure 4



Figure 5



Figure 6



Figure 7

Figure 1. 'Two friends', a painting (computer-generated drawing), oil on canvas. Collection: Hendel. © Becky Cohen. **Figure 2.** Hand-colored computer-generated drawing. **Figure 3.** Hand-colored computer-generated drawing, oil on canvas. Collection: Hendel. **Figure 4.** Hand-colored computer-generated drawing. **Figure 5.** 'Herb with wall hanging', a painting (computer-generated drawing), oil on canvas. © Becky Cohen. **Figure 6.** 'Meeting on Gauguin's beach', oil on canvas. Collection: Gwen and Gordon Bell. **Figure 7.** 'Two men on the edge', oil on canvas. *Note.* In all the figures, the original drawing is by the AARON program, written by Harold Cohen. Photographical acknowledgements are to Becky Cohen.

requires the development of generative systems which explicitly represent lower-level systems.)

A second example of an 'artistic' program is the jazz-improviser written by Philip Johnson-Laird^{8,9}. This has appeared in no concert-halls, and at first hearing seems much less impressive than Cohen's programs (Johnson-Laird reports that it performs at the level of a moderately competent beginner). However, it raises some highly specific questions—and provides some suggestive answers—about the nature of the complex conceptual space involved, and about how human minds are able to explore it.

A jazz-musician starts with a chord-sequence, such as a twelve-bar blues. (The performance will be an improvisation based on a fixed number of repetitions of the chord-sequence.) Often, the chord-sequence has already been written by someone else. For writing such sequences, unless they are kept boringly simple, typically requires a great deal of time and effort. They are complex hierarchical structures, with sub-sections 'nested' at several different levels, and with complex harmonic constraints linking sometimes far-separated chords. They could not be improvised 'on the fly' (where no backtracking is possible), but require careful thought and self-correction.

To take an analogy from language, consider this sentence: *The potato that the rat that the cat that the flea bit chased around the block on the first fine Tuesday in May nibbled is rotting*. You probably cannot understand this multiply-nested sentence without pencilling-in the phrase-boundaries, or at least pointing to them. If someone were to read it aloud, without a very exaggerated intonation, it would be unintelligible. Moreover, you would find it difficult, perhaps impossible, to invent such a sentence without writing it down. For you cannot select the word *is* without remembering *potato*, twenty-two words before. (If you had started with *The potatoes ...* you would have needed *are* instead.)

Similarly, jazz-composers cannot improvise complicated chord-sequences. Indeed, they have developed a special written notation to help them to keep the various harmonic constraints in mind while composing such sequences.

The jazz-musician's task, in playing a chord-sequence, is more difficult than yours in reading a sentence. For he is improvising, rather than merely reading. The 'chords' in the chord-sequence are actually classes of chords, and the player must decide, as he goes along, just how to play each chord. He must also decide how to pass to the next chord, how to produce a melody, how to harmonize the melody with the chords, how to produce a bass-line accompaniment, and how to keep the melody in step with the metre.

Johnson-Laird argues that, because of the limited storage-capacity of human short-term memory, the

rules (or musical 'grammar') used for generating these features of the performance must be much less powerful than the hierarchical grammar used to produce chord-sequences. Accordingly, his program consists of two parts.

One part generates a simple, harmonically sensible, chord-sequence (compare 'The potato is rotting'), and then complicates it in various ways to produce a nested hierarchical structure (comparable to a grammatically complex sentence). The second part takes that chord-sequence as its input, and uses less powerful computational rules to improvise a performance in real time. What counts as an acceptable 'melody', for instance, is determined by very simple rules which consider only a few previous notes; and the harmonies are chosen by reference only to the immediately preceding chord.

When more than one choice is allowed by the rules, the program chooses at random. A human musician might do the same. Or he might choose according to some idiosyncratic preference for certain intervals or tones, thus giving his playing an 'individual' style. (The same obviously applies for literature and painting.) This is one of the ways in which chance, or randomness, can contribute to creativity. But it is the constraints—governing harmony, melody, and tempo—which make the jazz-performance possible in the first place. Without them, we would have a mere random cacophony.

Besides harmony, melody, and tempo, there are other structures which inform music. Piano-music, for example, is composed to be played expressively (composers often put expression-marks in the score), and human musicians can play it with expression. Indeed, they have to: without expression, a piano-composition sounds musically dead, even absurd. In rendering the notes in the score, pianists add such features as *legato*, *staccato*, *piano*, *forte*, *sforzando*, *crescendo*, *diminuendo*, *rallentando*, *accelerando*, *ritenuto*, and *rubato* (not to mention the two pedals).

But how? Can we express this musical sensibility precisely? That is, can we specify the relevant conceptual space? Just what is a *crescendo*? What is a *rallentando*? And just how sudden is a *sforzando*?

These questions have been asked by Christopher Longuet-Higgins¹⁰ (whose earlier work on the conceptual space of tonal harmony was used within Johnson-Laird's jazz-program). Using a computational method, he has tried to specify the nature of the musical skills concerned. Working with Chopin's *Minute Waltz* and *Fantaisie Impromptu in C Sharp Minor*, Longuet-Higgins has discovered some counterintuitive facts about the conceptual space concerned. For example, a *crescendo* is not uniform, but exponential (a uniform *crescendo* does not sound like a *crescendo* at all, but like someone turning-up the volume-knob on a wireless); similarly, a *rallentando* must be exponentially

graded (in relation to the number of bars in the relevant section) if it is to sound 'right'. Where *sforzandi* are concerned, the mind is highly sensitive: as little as a centisecond makes a difference between acceptable and clumsy performance. By contrast, our appreciation of *piano* and *forte* is less sensitive than one might expect, for (with respect to these two compositions, at least) only five levels of loudness are needed to produce an acceptable performance. More facts such as these, often demonstrable to a very high level of detail, have been discovered by Longuet-Higgins' computational experiments. As he points out, many interesting questions concern the extent to which they are relevant to a wide range of music, as opposed to a particular musical style.

Strictly speaking, this work is not a study of creativity. It is not even a study of the exploration of a conceptual space, never mind its transformation. But it is highly relevant to creativity (as is Longuet-Higgins' earlier computational work on harmony). For we have seen that creativity can be identified only with respect to a particular generative system, or conceptual space. The more clearly we can identify this space, the more confidently we can identify and ask questions about the creativity involved in negotiating it. A pianist whose playing-style sounds 'original' may be exploring and transforming the space of expressive skills which Longuet-Higgins has studied.

Of course, we can recognize this originality 'intuitively', and enjoy—or reject—the pianist's novel style accordingly. Likewise, we can enjoy—or reject—drawings done by human artists or by computer programs. But understanding, in rigorous terms, *just how these creative activities are possible* is another matter. If that is our aim, computational concepts and computer modelling can help.

What of literature? There are many different conceptual spaces involved here. One of these concerns human motivation, the various psychological structures that are possible—and intelligible—within human action and interaction. Most novels and short-stories are less concerned with transforming this space than with exploring it in illuminating ways.

Current computer programs that write stories are woefully inadequate compared with human storytellers. But the best of them get what strength they possess from their internal models of very general aspects of motivation. Consider this example, written by a program asked to write a story with the moral 'Never trust flatterers':

The fox and the crow

Once upon a time, there was a dishonest fox named Henry who lived in a cave, and a vain and trusting crow named Joe who lived in an elm-tree. Joe had gotten a piece of cheese and

was holding it in his mouth. One day, Henry walked from his cave, across the meadow to the elm-tree. He saw Joe Crow and the cheese and became hungry. He decided that he might get the cheese if Joe Crow spoke, so he told Joe that he liked his singing very much and wanted to hear him sing. Joe was very pleased with Henry and began to sing. The cheese fell out of his mouth, down to the ground. Henry picked up the cheese and told Joe Crow that he was stupid. Joe was angry, and didn't trust Henry any more. Henry returned to his cave. THE END.

Exciting, this little tale is not. But it has a clear structure and a satisfactory end. The characters have goals, and can set up subgoals to achieve them. They can cooperate in each other's plans, and trick each other so as to get what they want. They can recognize obstacles, and sometimes overcome them. They can ask, inform, reason, bargain, persuade, and threaten. They can even adjust their personal relationships according to the treatment they get, rewarding rescue with loyalty or deception with mistrust. And there are no loose ends left dangling to frustrate us.

The reason is that this program can construct hierarchical plans, ascribing them to the individual characters according to the sorts of motivation (food-preferences, for example) one would expect them to have. It can think up cooperative and competitive episodes, since it can give one character a role (either helpful or obstructive) in another's plan. These roles need not be allocated randomly, but can depend on background interpersonal relations (such as competition, dominance, and familiarity). And it can represent different sorts of communication between the characters (such as asking, or bargaining), which constrain what follows in different ways. All these matters (like the body-models in Cohen's programs) are represented as abstract schemata, or generative systems, which are used to produce the story-structure.

A story-writer equipped not only to do planning, but also to juggle with psychological schemata such as escape, ambition, embarrassment, or betrayal could come up with better stories still. To design such a program would be no small feat. Every psychological concept involved in the plots of its stories, whether explicitly named in the text or not, would need to be defined—much as 'stability' had to be defined for the acrobat-drawing program, and 'melody' for the jazz-improviser.

The complexities are so great (and the background knowledge of the world so extensive) that it is unrealistic to expect there to be a computerized story-writer that can perform at better than a hack-level—if that. But our interest here is not in getting computers to do our creative acts for us, but in using the computational approach to help us understand what is involved when we do them. The complexity of the mind that is able to read *Hamlet* with understanding is

staggering, never mind the complexities involved in writing it.

Hamlet—or *Macbeth*. You may remember Macbeth's description of sleep:

Sleep that knits up the ravelled sleeve of care,
The death of each day's life, sore labour's bath,
Balm of hurt minds, great nature's second course,
Chief nourisher in life's feast.

This passage works because Shakespeare's readers, like him, know about such worldly things as knitting, night and day, and the soothing effects of a hot bath. In addition, they are able to understand analogies, even highly unusual or 'creative' analogies, such as comparing sleep with a knitter. But how can this be? A knitter is an animate agent, but sleep is not. How can the human mind map 'sleep' onto 'knitter' so as to realize the link: that both can repair the ravages of the previous day? Similarly, how can we understand Socrates' remark (in Plato's *Theaetetus*) that the philosopher is 'a midwife of ideas'? A philosopher is not (usually!) a midwife. And while a new idea is indeed new, vulnerable, and perhaps flawed—like a baby—it is nevertheless very different from a baby. Like sleep, ideas are not even animate. How, then, can someone create, or creatively interpret, such a strange comparison?

A recent computational model of analogy has successfully interpreted the philosopher-midwife analogy, mapping 'idea' onto 'baby' as required. This program is an example of a connectionist system. Such systems can help us to understand the psychology of pattern-recognition and analogy, and of 'creative' associations such as poetic imagery or Kekule's bringing-together of molecules and snakes.

A connectionist system involves many units, each coding one (often very simple) feature. The units are linked by excitatory and inhibitory connections (as are neurons in the brain). Units coding for mutually consistent features will tend to excite each other's activity, whereas mutually inconsistent units will inhibit each other. For instance, a unit coding for 'white' may excite both 'cream' and (less strongly) 'yellow', but it will inhibit 'blue', 'red', and (above all) 'black'.

These AI-models can take many different constraints into account at the same time, where no constraint is necessary but a large number are sufficient for making the judgment concerned. It follows that they are tolerant of 'noise' (missing or spurious information). Not surprisingly, then, connectionist systems are better than traditional AI-programs at modelling pattern-association. And since they can associate similar but non-identical patterns, they offer a promising basis for studying analogy.

The analogy-program let loose on Socrates' analogy

is based on a semantic network containing over thirty thousand words. The words are linked to one another more in the way of a thesaurus than a dictionary, for they bear links not only to synonyms and defining properties, but to less closely related words as well. (They also bear links to their opposites, as a thesaurus-entry does too; so this network could readily support many different uses of 'consider the negative'.) These semantic links are there not just because they are useful in getting the computer to do interesting things, but also because there is extensive psychological evidence that concepts in human minds are stored in broadly similar ways.

Using an analogy-mapper that compares concepts in terms of structural similarity, semantic centrality, and pragmatic (contextual) importance, this connectionist program interpreted Socrates' analogy successfully. That is, asked to come up with the 'best' equivalent of 'baby', it came up with 'idea'—*even though* it recognized the inconvenient fact that a central feature of a baby (its being alive) does not hold of an idea. After this mapping has been effected, of course, one can then describe ideas as alive, or as more or less worthy of survival. In other words, one can use the very feature which is missing in one pole of the analogy to explore that pole itself.

The analogy-interpreter has a 'sister-program' which comes up with analogies, as opposed to interpreting ready-made analogies input to it. In its current form, it would not spontaneously generate the idea-baby or sleep-knitter comparisons, because it looks for the 'best'—that is, the *closest*—analogy it can find. Even if it were told to ignore the twenty best comparisons, it would not come up with either of these notions. The reason is that the designers were primarily interested in the use of analogy in science, not in rhetoric or poetry. In poetry, *distance* between the two poles of the analogy is often preferred.

One could allow the program to generate highly distant analogies, but this would risk the production of 'wild' or 'crazy' comparisons. The context of a poem provides additional constraints which keep the wildness within the bounds of intelligibility. In the four-line fragment of Macbeth's speech, for instance, there is a succession of images for sleep each of which (even 'death') suggests some alleviation of previous troubles. The wildness of each individual analogy is thus tempered by the mutually reinforcing semantic associations set up by all the others. Even human minds cannot always use contextual associations fruitfully, nor even recognize them (a parliamentary candidate recently sent out leaflets showing her photographed in a neglected graveyard). Giving a program the grasp of semantic space needed to generate wild-yet-persuasive analogies, instead of close ones, would be a major exercise.

Programs and scientific discovery

Several 'inductive' programs have come up with useful (in some cases, H-novel) scientific ideas. For instance, a suite of programs designed to find simple mathematical and classificatory relations has 'rediscovered' many important physical and chemical laws¹¹. And an expert system (dealing with a strictly limited area of stereochemistry) has drawn chemists' attention to molecules they had not previously thought of^{12,13}. Like the systems discussed in the previous section, however, these programs are exploratory rather than transformational.

Programs capable of *transforming* their own conceptual space are still few and far between. One such is the 'Automatic Mathematician' (AM)^{14,15}. This system does not produce proofs, nor solve mathematical problems. Rather, it generates and explores mathematical ideas, coming up with new concepts and hypotheses to think about.

AM starts out with 100 very primitive mathematical concepts drawn from set-theory (including sets, lists, equality, and operations). These concepts are so basic that they do not even include the ideas of elementary arithmetic. To begin with, the program does not know what an integer is, still less addition, subtraction, multiplication, and division.

Also, AM is provided with about 300 heuristics. These can examine, combine, and transform AM's concepts—including any compound concepts built up by it. Some are very general, others specific to set-theory, and they enable AM to explore the space potentially defined by the primitive concepts. This exploration involves conceptual change, by means of various combinations and transformations.

For example, AM can generate the *inverse* of a function. This heuristic (a mathematical version of 'consider the negative') enables the program to define multiplication having already defined division, or to define square-roots having already defined squares. Another transformation generalizes a concept by changing an 'and' into an 'or' (compare relaxing the membership-rules of a club, from 'Anyone who plays bridge and canasta' to 'Anyone who plays bridge or canasta').

However, AM does not consider *every* negative, nor change *every* 'and' into an 'or'. Time and memory do not allow this. Like all creative thinkers, AM needs hunches to guide it along some paths rather than others. And it must evaluate its hunches, if it is to appreciate its own creativity. Accordingly, some of AM's heuristics suggest which sorts of concept are likely to be the most interesting. If it decides that a concept is interesting, AM concentrates on exploring that concept. For example, it takes note if it finds that the union of two sets has a simply expressible property

that is not possessed by either of them. This is a mathematical version of the familiar notion that *emergent* properties are interesting. In general, we are interested if the combination of two things has a property which neither constituent has.

AM's hunches, like human hunches, are sometimes wrong. Nevertheless, it has come up with some extremely powerful notions. It produced many arithmetical concepts, including *integer*, *prime*, *square root*, *addition*, and *multiplication*. It generated, though did not prove, the fundamental theorem of number theory: that every number can be uniquely factorized into primes. And it suggested the interesting idea (Goldbach's conjecture) that every even number greater than two is the sum of two different primes. It defined several concepts of number theory by following unusual paths—in two cases, inspiring human mathematicians to produce much shorter proofs than were previously known. It has even originated one minor theorem which no-one had ever thought of before (concerning 'maximally-divisible' numbers, which AM's programmer knew nothing about). In short, AM appears to be significantly P-creative, and slightly H-creative too.

Some critics have suggested that this appearance is deceptive, that some of the heuristics were specifically included to make certain mathematical discoveries possible. In reply, AM's programmer insists that the heuristics are fairly general ones, not special-purpose tricks. On average, he reports, each heuristic was used in making two dozen different discoveries, and each discovery involved two dozen heuristics. Even so, a given heuristic may have been used only once, in making an especially significant discovery. (A detailed trace of the actual running of the program would be needed to find this out.) The question would then arise whether it had been put in for that specific purpose, or for exploring mathematical space in a more general way. The precise extent of AM's creativity, then, is unclear. But we do have some specific ideas about what sorts of questions are relevant.

Whereas AM has heuristics for altering concepts, a successor-program (EURISKO)¹⁵⁻¹⁷ possesses heuristics for changing heuristics. As a result, EURISKO can explore and transform not only its stock of concepts, but its own processing-style.

For example, one heuristic asks whether a rule has ever led to any interesting result. If it has not (given that it has been used several times), it is marked as less valuable—which makes it less likely to be used in future. What if the rule has occasionally been helpful, though usually worthless? Another heuristic, on noticing this, suggests that the rule be specialized. The new heuristic will have a narrower range of application than the old one, so will be tried less often (thus saving effort). But it will be more likely to be useful in those cases where it is tried.

Moreover, the 'specializing-heuristic' can be applied to itself. Because it is sometimes useful and sometimes not, EURISKO can consider specializing it in some way. The program distinguishes several sorts of specialization, and has heuristics for all of them. Each is plausible, for each is often (though not always) helpful. And each is useful in many different domains. One form of specialization requires that the rule being considered has been useful at least three times. Another demands that the rule has been very useful, at least once. Yet another insists that the newly specialized rule must be capable of producing all the past successes of the *unspecialized* rule. And a fourth heuristic specializes the rule by taking it to an extreme.

Other heuristics work not by specializing rules, but by generalizing them. Generalization, too, can take many forms. Still other heuristics can create new rules by analogy with old ones. Again, various types of analogy can be considered.

With the help of various packets of specialist heuristics to complement these general ones, EURISKO has been applied in several different areas. It has come up with some H-novel ideas, concerning genetic engineering and computer-chip (VLSI) design. Some of its ideas have even been granted a US patent (the US patent-law insists that the new idea must not be 'obvious to a person skilled in the art').

A third example of a self-transforming program uses IF-THEN rules to regulate the transmission of oil through a pipeline in an economical way. It receives hourly measurements of the oil-inflow, oil-outflow, inlet-pressure, outlet-pressure, rate of pressure-change, season, time of day, time of year, and temperature. Using these data, it alters the inlet-pressure to allow for variations in demand, infers the existence of accidental leaks, and adjusts the inflow accordingly.

But the program was not told which rules to use for adjusting inflow, or for detecting accidental leaks. It discovered them for itself. It started from a set of randomly generated rules, which it repeatedly transformed in part-random, part-systematic, ways. To do this, it used heuristics called genetic algorithms¹⁸. These enable a system to make changes that are both plausible and unexpected, for they produce novel recombinations of the most useful parts of existing rules.

As the name suggests, these heuristics are inspired by biological ideas. Some genetic changes are isolated mutations in single genes. But others involve entire chromosomes. For example, two chromosomes may swap their left-hand sides, or their midsections (the point at which they break is largely due to chance). If a chromosome contained only six genes, then the strings ABCDEF and PQRSTU might give ABRSTU and PQCDEF, or ABRSEF and PQCDTU. Such transformations can happen repeatedly, in successive genera-

tions. The strings that eventually result are unexpected combinations of genes drawn from many different sources. Genetic algorithms in computer programs produce novel structures by similar sorts of transformation.

In general, the plausibility of the new structures produced by this sort of exploratory transformation is increased if the swapped sections are coherent mini-sequences. However, there is a catch—or rather, several. The first is that a self-adapting system must somehow identify the most useful 'coherent mini-sequences'. But these never function in isolation: both genes and ideas express their influence by acting in concert with many others. The second is that coherent mini-sequences are not always sequences. Co-adapted genes (which code for biologically related functions) tend to occur on the same chromosome, but they may be scattered over various points within it. Similarly, potentially related ideas are not always located close to each other in conceptual space. Finally, a single unit may enter more than one group: a gene can be part of different co-adaptive groups, and an idea may be relevant to several kinds of problems.

Programs based on genetic algorithms help to explain how plausible combinations of far-distant units can nevertheless happen. They can identify the useful parts of individual rules, even though these parts never exist in isolation. They can identify the significant interactions between rule-parts (their mutual coherence), even though the number of possible combinations is astronomical. And they can do this despite the fact that a given part may occur within several rules. Their initial IF-THEN rules are randomly generated (from task-relevant units, such as *pressure*, *increase*, and *inflow*), but they can end up with self-adapted rules rivalling the expertise of human beings.

The role of natural selection is modelled by assigning a 'strength' to each rule, which is continually adjusted according to its success (in controlling the pipeline, for instance). The relevant heuristic is able, over time, to identify the most useful rules, even though they act in concert with many others—including some that are useless, or even counter-productive. The strength-measure enables the rules to compete, the weak ones gradually dropping out of the system. As the average rule-strength rises, the system becomes better adapted to the task-environment.

The role of variation is modelled by heuristics (genetic operators) that transform the rules by swapping and inserting parts in ways like those outlined above. For instance, the 'crossover' operator swaps a randomly selected segment between each of two rules. Each segment may initially be in a rule's IF-section or its THEN-section. In other words, the crossover heuristic can change either the conditions that result in a certain

action, or the action to be taken in certain conditions, or both.

One promising strategy would be to combine the effective components of several high-strength rules. Accordingly, the genetic operators pick only rules of relatively high strength. But the effective components must be identified (a rule may include several conditions in its IF-side and several actions in its THEN-side). The program regards a component as effective if it occurs in a large number of successful rules. A 'component' need not be a sequence of juxtaposed units. It may be, for instance, two sets of three (specified) neighbouring units, separated by an indefinite number of unspecified units. The huge number of possible combinations do not have to be separately defined, nor considered in strict sequence. In effect, the system considers them all in parallel (taking into account its estimate of various probabilities in the environment concerned).

Some philosophical puzzles

What of the fourth Lovelace-question? Someone who agrees that a computer could mimic human creativity to a very high degree may nevertheless refuse to credit any computer with *real* creativity. No matter how impressive the performance, it must (on this view) be mere empty mimicry. If so, then the answer to the fourth Lovelace-question is a resounding 'No!'

The belief that no computer could *really* be creative can be defended in several different ways. Let us consider just two of them: the brain-stuff argument and the non-human argument.

The brain-stuff argument relies on a *factual hypothesis*: that whereas neuroprotein is a kind of stuff which can support intelligence, metal and silicon are not.

The hypothesis driving this argument may, conceivably, be true. Possibly, computers are made of a sort of material stuff which is incapable of supporting creative intelligence. Indeed, neuroprotein may be the *only* substance in the universe which has this power. Then again, it may not: there may be thinking creatures on Mars with alien chemicals filling their heads. Science does not tell us this is impossible. But nor does science give us any good reason, at present, to think that metal and silicon are essentially incapable of embodying the many stable yet adaptive structures involved in creative thought.

Some people regard it as intuitively obvious that these materials cannot support intelligence, whereas neuroprotein can. But this is not obvious at all. Certainly, neuroprotein does support intelligence, meaning, and creativity. But we understand almost nothing of how it does so, *qua* neuroprotein—as opposed to some other chemical stuff.

Indeed, insofar as we do understand this, we focus on the neuro-chemistry of certain basic *computational functions* embodied in neurones: message-passing, facilitation, inhibition, and the like. Neurophysiologists have discovered the 'sodium-pump', for instance. This is the electrochemical process, occurring at the cell-membrane, which enables an electrical signal to pass (without losing strength) from one end of a neurone to the other. But this mechanism is psychologically interesting only to the extent that it helps us to understand psychologically relevant functions. Any other chemical process would do, provided that it allowed a neurone to propagate a message from one end to the other.

The fact that we cannot see how metal and silicon could possibly support 'real' intelligence is irrelevant. For, intuitively speaking, we cannot see how neuroprotein—that gray mushy stuff inside our skulls—can do so either. No mind-matter dependencies are intuitively plausible. Nobody who was puzzled about intelligence (as opposed to electrical activity in neurones) ever exclaimed '*Sodium*—of course!'. Sodium-pumps are no less 'obviously' absurd than silicon chips, electrical polarities no less 'obviously' irrelevant than clanking metal. Even though the mind-matter roles of sodium-pumps and electrical polarities are scientifically compelling, they are not intuitively intelligible. On the contrary, they are highly counter-intuitive.

Our intuitions will doubtless change, as science advances. Future generations may come to see neuroprotein—and perhaps silicon, too—as 'obviously' capable of embodying mind, much as we now see biochemical substances as obviously capable of producing other such substances (a fact regarded as intuitively absurd, even by most chemists, before the synthesis of urea in the nineteenth century). As yet, however, our intuitions have nothing useful to say about the material basis of intelligence.

In sum, the brain-stuff argument is inconclusive. It reminds us that computers made of non-biological materials may be incapable of real creativity. But it gives us no reason whatever to believe that this is actually so.

The non-human argument holds that to regard computers as *really* creative is not a mere factual mistake, but a moral absurdity. To answer 'Yes' to the fourth Lovelace-question, on this view, would be to grant certain rights to computers which should be granted only to people.

Each of us has aims, fears, and beliefs, all of which—unless the contrary can be specifically shown—deserve to be respected. Everyone has a right to be heard, a right to try to persuade others, and a right to further their interests. These rights are fundamental to human society. Supporters of the non-human argument insist that we should forever refuse to allow computers any social roles like those enjoyed by people.

To remove all the scare-quotes from psychological words when describing computer programs, to regard them as *literally* intelligent and creative, would be to admit them into our moral universe. We should then have to respect their interests, even—on occasion—above our own. (So someone late for an appointment might excuse himself not by saying 'I had to take the dog for his walk' but 'I had to find a reference for my computer, which wanted to finish the *Times* crossword before starting on its maths problems'.) Similarly, to regard computer-systems as *really* intelligent would mean that they could be deceived and that, all things being equal, we should not deceive them. It would mean, too, that they could *really* know the things they were apparently saying, so we could *really* trust them.

The last example has already come up in the English law-courts. A man was accused of stealing banknotes, and the prosecution submitted a list of banknote-numbers, some of which matched notes found in his possession. In law, documents accepted as evidence must be produced by someone 'having knowledge of their contents. But the crucial list had been produced by the bank's computer. Because a computer (so the judge said) cannot have any knowledge of anything, the accused was acquitted.

Presumably, you regard this as an unsatisfactory outcome. But whatever you think the lawyers should have said in this case, the crucial point is that the decision to remove all scare-quotes when describing programs in psychological terms would carry significant moral overtones. So, like moral decisions in general, it cannot be forced upon us by the facts alone.

Finally, we must consider the common fear that a scientific explanation of creativity (whether computational or not) would not merely demystify creativity, but destroy it as well. Many people regard science as dehumanizing, in the sense that it ignores—or even denies—the existence of purpose, freedom, and subjectivity. Accordingly, they assume that a scientific psychology would at best devalue creativity, and at worst deny it.

The natural sciences have indeed been insidiously dehumanizing. For they have had no concepts capable of expressing subjectivity, no way of describing—still less explaining—how human minds can construct their personal worlds in culturally diverse and idiosyncratic ways. Their silence on matters of the mind, given their many successes and high social status, have (as William Blake foresaw) de-emphasized the phenomena which humanists value most. 'Tough-minded' psychologists (behaviourists, for instance) have been the worst offenders, often explicitly denying the reality or scientific interest of the matters intuitively discussed by 'tender-minded' humanists.

But now, at last, we have a scientifically-grounded vocabulary in which to express such matters precisely.

The central concept of computational psychology, and of artificial intelligence too, is *representation*. This concept enables us to ask questions about the meanings embodied in psychological systems, and the ways in which they mediate thought and action. Certainly, many more computational concepts (and advances in neuroscience, too) will be required before we have an adequate scientific account of the human mind. But to dismiss computational psychology as philosophically bogus because of its many current shortcomings would be like a seventeenth-century philosopher rejecting Galileo's suggestion that 'mathematics is the language of God' because—having no differential equations—he could not explain fluid dynamics.

Sometimes, to be sure, a scientific explanation of behaviour destroys our previous valuation of it. For example, someone might praise the 'intelligence' of the hoverfly, which is able to meet its mate in mid-air. They might assume that it can decide where it wants to get to, and purposefully weave and duck on its way, much as a person can intercept a friend in a crowded square. But this would be a sentimental illusion.

It turns out that the explanation of the fly's behaviour is a mechanism hardwired in the brain, which connects a specific visual signal with a specific muscular response. The flight-path depends strictly on the approach-angle subtended by the target-fly, and it allows of no variation once it has begun. On discovering this simple trigger-effect, the sentimentalist will be cruelly disillusioned. The hoverfly's 'intelligence' has been demystified with a vengeance.

This loss of respect for the hoverfly's intellectual powers is due to the discovery that the fly's 'mind' is much less complex than had been thought. But computational research shows that the human mind is much *more* rich than psychologists previously believed. Even Freud, whose writings were informed by a subtle 'literary' intuition, did not fully realize its complexity.

In sum, a computational science of creativity is not dehumanizing. It does not threaten our self-respect by showing us to be mere machines, for some machines are much less 'mere' than others. It can allow that creativity is a marvel, despite denying that it is a mystery.*

*The arguments in this paper are more fully explored in M. A. Boden, *The Creative Mind: Myths and Mechanisms*, Weidenfeld & Nicolson, London, 1990; paperback (expanded): Cardinal, 1992.

-
1. Lovelace, A., Notes on Manabrea's Sketch of the Analytical Engine Invented by Charles Babbage, in *Faster Than Thought* (ed. Bowden, B. V.), Pitman, London, 1953, pp. 362–408.
 2. Hyman, A., *Charles Babbage: Pioneer of the Computer*, Oxford University Press, Oxford, 1982.
 3. Koestler, A., *The Act of Creation*, Picador, London, 1975. (First Published 1964.)

4. Findlay, A., *A Hundred Years of Chemistry* (ed. Williams, T. I.), 3rd edn., Duckworth, London, 1965.
 5. Cohen, H., *On the Modelling of Creative Behavior*, Rand Corporation, Santa Monica, California, 1981, Rand Paper p. 6681.
 6. Karmiloff-Smith, A., From Meta-processes to Conscious Access: Evidence from Children's Metalinguistic and Repair Data, *Cognition*, 1986, 23, 95-147.
 7. Karmiloff-Smith, A., Constraints on Representational Change: Evidence from Children's Drawing, *Cognition*, 1990, 34, 57-83.
 8. Johnson-Laird, P. N., Freedom and Constraint in Creativity, in *The Nature of Creativity: Contemporary Psychological Perspectives* (ed. Sternberg, R. J.), Cambridge University Press, 1988, pp. 202-219.
 9. Johnson-Laird, P. N., Jazz Improvisation: A Theory at the Computational Level, unpublished working-paper, MRC Applied Psychology Unit, Cambridge, 1989.
 10. Longuet-Higgins, H. C., *Mental Processes: Studies in Cognitive Science*, MIT Press, Cambridge, Mass., 1987.
 11. Langley, P., Simon, H. A., Bradshaw, G. L. and Zytkow, J. M., *Scientific Discovery: Computational Explorations of the Creative Process*, MIT Press, Cambridge, Mass, 1987.
 12. Buchanan, B. G., Smith, D. H., White, W. C., Gritter, R., Feigenbaum, E. A., Lederberg, J. and Djerassi, C., Applications of Artificial Intelligence for Chemical Inference: XXII Automatic Rule Formation in Mass Spectrometry by Means of the Meta-Dendral Program, *J. Am. Chem. Soc.*, 1976, 98, 6168-6178.
 13. Lindsay, R., Buchanan, B. G., Feigenbaum, E. A. and Lederberg, J., *DENDRAL*, McGraw-Hill, New York, 1980.
 14. Lenat, D. B., The Ubiquity of Discovery, *Artif. Intell.*, 1977, 9, 257-286.
 15. Lenat, D. B., The Role of Heuristics in Learning by Discovery; Three Case Studies, in *Machine Learning: An Artificial Intelligence Approach* (eds. Michalski, R. S., Carbonell, J. G. and Mitchell, T. M.), Tioga, Palo Alto, Calif, 1983.
 16. Lenat, D. B. and Seely Brown, J., Why AM and EURISKO Appear to Work, *Artif. Intell.*, 1984, 23, 269-294.
 17. Ritchie, G. D. and Hanna, F. K., AM: A Case Study in AI Methodology, *Artif. Intell.*, 1984, 23, 249-268.
 18. Holland, J. H., Holyoak, K. J., Nisbett, R. E. and Thagard, P. R., *Induction: Processes of Inference, Learning and Discovery*, MIT Press, Cambridge, Mass., 1986.
-