Syntax, and Something of Wildness

HOW TO WRITE POEMS WITH A COMPUTER

By JOHN MORRIS

The first poem that our computer wrote went like this:

> Still midnight, silent,
> Still waters, still frozen,
> Battle dusk, and far.

With a little practice, it was turning out seventeen-syllable Japanese-style haikus at the rate of two per second. The poems have rather a somber tone, a tendency to repetition, and a preoccupation with scarecrows. Here is another one, one of our scarecrow poems:

> The savage, savage
> Scarecrow, down in silent dusk,
> Frozen, well frozen.

The program was written, officially, to test out a new language on Michigan State University's computer. The language was designed to deal with lists, and it seemed natural to use it to write poems. It seemed to me that you could think of a poem (if you wanted to) as a kind of list, made up of sublists for its stanzas.

A little Peter Pauper collection of haikus (*Japanese Haiku,* Peter Beilenson, translator, Peter Pauper Press, Mount Vernon, N.Y., 1956) happened to be on the shelf, left over from Christmas. It furnished both the form and the vocabulary for my poetry-writing computer-program. The requirements were simple. A haiku is a three-line poem. Its first and third lines have five syllables; the second line has seven. There are no absolute rules about the subject matter, but a haiku is much too short for more than a quick glance at a falling leaf, the flutter of a fan, the play of light on water. For such a brief, perhaps ironic, image, it is perfect.

Using the anthology as a guide, I put together a list of haiku-like words, which form a sort of dictionary for the computer to consult. From these words, it selects one at a time, filling each line with the proper number of syllables, until it has three together. Then it prints the poem that it has composed. Sometimes the result is rather pleasant:

> Distance, I listen:
> Far weird savage frozen spring,
> Old song, echo still.

By the time it had gone through a dozen trial runs, the computer had produced some four thousand haikus. For one glorious summer month, I was the world's most prolific poet.

By autumn, unfortunately, the pages were overflowing my office. They badly needed an audience. I was writing poems more rapidly than even I could read them. As a desperate expedient, I retyped a dozen of the best of them and mailed them over to Ann Arbor, to *Generation,* the University of Michigan's student literary magazine. I think that the title I gave the sheaf of them was "At Random," a title that was, I must confess, painfully appropriate.

It would have made a delicious hoax. I looked forward to the newspaper stories, psychological comments on my strange stream-of-consciousness, angry arguments in

MR. MORRIS (A.B. in English '49; A.M. in Math, '51; B.D., the Starr King School in Berkeley, '54), a Unitarian minister for nine years, is now Chief Programmer for the Computer Institute for Social Science Research at Michigan State University, where he is also working on a Ph.D. in philosophy (with no contradiction in terms).

defense of modern poetry, the horror of the critics at the great denoument.

Unhappily, the poems weren't accepted. *Generation* didn't even send a rejection slip. Perhaps the poems are still sitting, after a year and more, in some editor's file, waiting for imminent publication, and perhaps I've spoiled the whole thing by admitting it here, but I doubt it. The poems are awful.

Why in the world, if the computer follows all the rules for haikus, does it turn out such terrible poetry? Does computer art *have* to be bad?

As a kind of answer to this question, I want to describe the way to write a really effective poetry-writing program for the computer. Such a program will need two basic ingredients. One of these is algorithmic; the other is random.

An *algorithm* is named for one Abu Ja'far Mohammed ben Musā Khowārezmī, a ninth-century Arabic arithmetician who helped make zero popular in Europe. Besides introducing Arabic numerals, he seems to have given simple, step-by-step methods for doing division, multiplication, or what-have-you. The computer people, abetted by the New Math people, have captured the word to describe any effective step-by-step procedure or set of instructions. The definition is broad enough to include a recipe in a cookbook. But the prime example of an algorithm is a computer program, a set of instructions for the machine. Without some such set of instructions, the computer doesn't know what to do. It *must* have an algorithm.

Our poetry-writing algorithm will have to have at least three main sections, to choose the words, to test them, and to put them together in the right order. This latter sort of operation is *syntax,* or, in a very general sense, grammar. If you're going to write in English, you have to put the words together in the proper order.

We'll want to compose a long list of the rules of English, and feed these into the computer:

1. Prepositions are bad things to end sentences with.
2. Don't end a sentence with a the.
3. Don't end a sentence with a conjunction, either. . . .

And so on, through all the grammar-book rules.

Unfortunately, rules like these don't help much with poetry, especially with haikus, where the compression of the form means that syntax has to be left somewhat loose. Take one of the best of our computer haikus:

> Glittering midnight:
> Our hollow well, glittering,
> Silent, savage, weird.

Most of the standard rules of syntax would have forbidden this particular combination of words. Although we need syntax, we also seem to need something random, something of wildness, built into the syntax, which will let it break its own rules, some of the time. For producing poetic syntax, we seem to need both an algorithm and an anti-algorithm in our program.

In addition to syntax, we will have to do something about *semantics*. Semantic rules tell us what it is that we're supposed to be writing about. If we're going to write nature haikus, we'll have to know something about nature. Our computer will need rather a large number of rules, telling it that snow is not likely to fall during the summer, that the word "spring" has several rather different meanings from one sentence to the next, and that frogs hibernate during the winter. The semantic algorithm will have to be something about the size of an encyclopedia, which makes the job difficult, but not overwhelming. One computer already has committed the Golden Encyclopedia to memory, and our modest haikus could get along on less.

The real difficulty is in finding some way to break the semantic taboos. One of the major happy effects of poetry generally, and of haikus in particular, is *surprise*. For the sake of surprise, we might want to be able to have snow falling in the spring (with an ambiguous reference to the petals of cherry trees), or to have an old man thinking nostalgically of love, during his (real or figurative) autumn. The list of semantic rules,

then, would have to include some kind of procedure for random by-passing that would let *some* of the lines break the rules, to mix metaphors, or to put discordant ideas ironically together.

In addition to working on syntax and semantics, we have to consider the *texture* of the poem, meaning by this all its sounds and surface features. This is something that the computer linguists have tended to neglect. But they have concentrated their attention on prose translations; poetry must have the proper texture.

We'll want to use whatever resources the English language gives us. For some forms of poetry, we could tell the computer to count syllables, take account of rhythms, and throw out words that don't rhyme. There are no problems here. Our haikus, though, require something a little freer. The computer must pay attention to rhythm and sound, and it must somehow link texture with semantics to make each one complement the other—all without becoming obnoxiously evident in its task. It must grow banal when speaking of banalities, cool or crisp for the displeased mistress, hot and languid for a summer shower. At times it must play with the sheer sound of words (Whitman's "Weapon shapely, naked, wan.") But the use of any of these devices would grow tedious, like a Swinburne epic. We would want a long list of textural algorithms to draw upon, from which we could select, at random, those that would be used in a poem.

Those are the three algorithms that we want, for syntax, for semantics, and for texture. Each of them must have some random method of overriding the rules for the sake of interest and surprise.

Randomness, then, must balance the algorithmic in our computer poems. Randomness, in the sense that the outcome of a flipped coin is random. (In actual practice, the task of getting numbers that behave like random numbers has engaged the best minds of our generation for more than a decade. We are quite happy to let Argonne National Laboratory work on this problem for us, even though their random numbers might not be *quite* as random as those of, say, the Rand Corporation.)

There is something going on in a poem that we try to catch by reaching out at random. We grope in the dark, and we write down whatever we happen to catch. This, as the poets have pointed out, is an analogy to the act of the poet, but it is no more than an analogy. Philosophic poets, like Plato and Lucretius, have never been satisfied with a *purely* algorithmic, world, one in which each line of a poem, each petal of a flower, was completely determined. Both rejected the wholly-determinate world of the atomists. Plato refused even to mention it; Lucretius and Epicurus believed that there was in the world a random factor, a *clinamen* or swerve, a kind of untamed quality about the atoms, which led them to leave their orbits at random. Lucretius thought that he had found human freedom in this randomness.

I am not at all happy about Lucretius's way of solving the problem. There seems, somehow, to be something other than randomness in human genius. Whatever this third element is (and we can get as gushy and mystical about it as you please), it is neither algorithmic nor random, neither determined nor chance; and it is peculiarly present in poetry.

Compare the human with the machine when they write nonsense. Our poetry program has no trouble in turning out nonsense. We've written thousands of nonsense verses, the machine and I. But compare them with this bit of traditional nonsense from Mother Goose:

> Three children sliding on the ice
> Upon a summer's day;
> As it fell out, they all fell in—
> The rest, they ran away.

Our semantic algorithm, if it were *very* good, might just be able to do the first two lines of this. It would know that there is no ice in summer, and its randomness function would decide to invert the meaning. The last two lines might be worked in the same way. Calling in a logical algorithm, the computer would reason: "All the children have fallen in. Nothing that falls in can run away.

Therefore, none of the children can run away." At this point, the randomness function would reverse the logic, permitting some of the children to escape in defiance of the logic.

But what in the world will ever give us an algorithm to handle the delicious nonsense of the third line? Here, even the mightiest computer trembles. Its encyclopedia must know that "as it fell out" means "as it happened" in somewhat archaic English. No great difficulty here. But then it must join this phrase to another with a parallel structure, in which the meaning is completely different, which contributes to the overall meaning of the poem. The two kinds of meanings clash and resolve themselves. You read them, catch your breath, and then read them again, laughing at the trick the poet has played on you. Writing an algorithm to pull of this kind of trick will be something of a job.

Although the job, as I've outlined it, is a big one, it is not overwhelming. Not yet. There is, though, something else going on in a poem that we don't seem to have captured.

Here is a haiku that my little girl wrote one day last autumn:

The wind is blowing—
Gently touching the fall trees
Of red and yellow.

I like this poem very much. It recalls a number of things: A walk that our family took through the Arboretum. The red and yellow leaves with which my little girl decorated the table for me one evening. The packet of colored leaves that she once mailed to me at my office. All of these are part of my delight in the poem. As a poem, as a list of words, it has its own merits, its image of the wind "gently touching" the trees, or the falling cadences of its first and last lines. But, for me, it gets its major effect because it is a communication from a particular human being. And this is precisely what the computer is not.

Perhaps the difficulty of writing poems with a computer may discourage all computer programmers from ever trying seriously to do it. If it does, then poems may remain, as they are now, one last refuge for human beings, for their personal communcations (not to be confused with what the schools call "communication science"), for the strange, non-random phenomenon that we call love.

WOODCUT: WINTER

The snow suspended like gnats
(more persistent even than they
if less wilfully personal),
is it falling up or down?

The frozen roses in the dooryard
and the red berries in bunches
that even the birds forego
each with its crazy snow cap

erasing with white the stems;
roses and berries float
in seen and unseen snow
with no earthly connection.

And snow floats in air
with no earthly connection
tickling the underside of eaves.
How manifest! How hidden!

ALBERT HOWARD CARTER