Darwin, a Game of Survival of the Fittest among Programs

Darwin, a competition between self-reproducing programs, was conceived and played forty years ago at Bell Laboratories. The following pages were transcribed from a 1971 letter on which was based the first public description of Darwin, in the Computer Recreations column of *Software—Practice and Experience*, Volume 2 (1972) pages 91-96, written by ‘‘Aleph-Null’’. A decade further on, A. K. Dewdney’s *Scientific American* column for May, 1984, described an updated version of the game, rechristened as Core Wars.

The letter includes the original 1961 rules of the game, descendants of which are alive and well today; see for example http://www.corewars.org.

The transcript was made by OCR with manual postediting. Aside from uncaught OCR errors, the text is original, but pagination, lineation, fonts, and clearly identified editorial comments are not. It should be noted that Aleph-Null and C. A. Lang are not the same person.

Bell Laboratories
600 Mountain Avenue
Murray Hill, New Jersey
07974
Phone (201) 582-3000
June 29, 1971

MR. C. A. LANG, Editor
Software Practice and Experience
Computer Laboratory
Corn Exchange Street
Cambridge CB2 3QG
England

Dear [hand-inked Aleph-Null]:

Ten years ago Vyssotsky invented an unusual game among computer programs *qua* programs, quite different from the usual games where programs are merely expressions of algorithms, necessary perhaps for speed, but not intrinsically dependent upon the computer. He propounded it one day in August 1961; McIlroy coded the umpire that night; a Bell Laboratories comp center job number, the abstract of which became the enclosed flyer ‘‘Darwin: A Game of Survival and (Hopefully) Evolution’’ was filed next morning; several rounds of the game were played until a couple of weeks later Morris invented the ultimately lethal competitor.

The rules given in the flyer should be augmented by the specific calls that a player makes on the umpire

```
hisno,bot,top = probe(loc,myno)
kill(loc,myno)
claim(loc,myno)
```

where

loc = a memory address in the arena
myno = my species number
hisno = 0 if loc is unoccupied else the number of the species occupying loc
bot,top = limits (first and last + 1) of the largest contiguous free space surrounding loc including space taken up by any occupant of loc

For any particular ‘‘loc’’, ‘‘probe’’ must precede ‘‘kill’’ or ‘‘claim.’’ Of course control vanished without a trace when ‘‘probe’’ landed on a protected cell. ‘‘Kill’’ would do in any individual regardless of species number. ‘‘Claim’’ specified an origin (necessarily between bot and top) for a newly reproduced individual. Any unreasonable request to the umpire resulted in extermination of the offending species.

The smallest creature capable of looping through memory and performing all three functions was about 30 [IBM] 7090 instructions long. The calling sequences were very efficient so most of the code in fact was involved with searching and reproduction — the latter being a nontrivial task since the code of an individual could not simply be copied but had to be correctly relocated. Vyssotsky invented a move-and-relocate loop of five instructions that instantly became standard.

McIlroy had a tiny creature only 15 cells long that could probe and kill, for which we had to cut down the limit of 20 protected cells lest it be immortal. Before decent search strategies evolved this simple hard-shelled ‘‘virus’’ actually won a few rounds.

Morris’s lethal beast that brought an end to the game occupied 44 memory cells and incorporated an ingenious adaptive search. Once an individual of his species had successfully located the top of a free space, and thereby the bottom of some other (unknown) individual it would pick a small integer and poke that far up into the newly located creature. When it successfully found an enemy it would use the same increment on the next individual in memory and so on. If on the other hand it lost control, next time the creature got control it would change its poking increment. Newly reproduced individuals were initialized with a successful increment. By this absurdly simply strategem Morris had made a species each individual of which adapted itself to be lethal against the pattern of protection of some other species. The more thickly memory was inhabited by the enemy, the more virulent was Morris’s attack.

One experiment in ‘‘bisexualism’’ was a fbp. Species had two kinds of individuals, one to search and destroy and another to reproduce either kind. The two species had to work so hard to find each other that each individual occupied almost as much space as an individual of a unisex species, so the division of labor gained nothing in viability. Worse yet, the species could be done in by wiping out just the members of one sex instead of the whole population.

Some rounds of the game, especially when two or more differently protected species of Morris’s were pitted against each other, showed an oscillation in population. If the population of species A swelled at the expense of B it would become vulnerable to C and so on in rotation.

We usually used an arena of 10,000 locations. The games always went fast, under a minute, thanks to the fact that the individual programs were executed rather than interpreted. Interpretation would have guaranteed that the rules were obeyed but the honesty rule that all programs must be circulated among all players at the end of each round was just as satisfactory and probably made the game go 20 or 30 times faster.

Besides the amusing exercise of writing self-reproducing programs, the game posed another unusual implementation puzzle: How conveniently to use existing system features to get the game started? We had to load a large number of copies of each of a handful of very small programs at random origins. Needless to say, our regular loader had not been intended for this application. Here was a case where we wanted the ‘‘scatter’’ of ‘‘scatter loading’’ to mean what it does in ‘‘scatter storage.’’* We actually resorted to ginning up decks of loader origin control cards and creating corresponding tables for the umpire in a semiautomatic fashion. We would sandwich all this stuff by hand between a few system control cards to make it into a job. In those bad old days the computing system was not very good at feeding itself (and to this day the manufacturers seem to think this is a just and proper state of affairs!).

Though Morris cleaned up under the particular rules, Darwin would again become interesting if the umpire introduced a little randomness in any of a number of ways. One

* "Scatter loading’’ referred to binary punched cards that told on a per-card basis where the data belonged in memory; typically successive cards were contiguous. ‘‘Scatter storage’’ is what is now called a hash table.

can then imagine trying more elaborate pattern-matching searches, to be countered by cryptography, then ... Machine language buffs will get a special kick out of Darwin, for here cleverness is all; and cleanness counts for nought.

M. D. McIlroy

R. MORRIS

V. A. VYSSOTSKY

Attachment

DARWIN: A GAME OF SURVIVAL AND (HOPEFULLY) EVOLUTION

by
V. A. Vyssotsky et al.
Bell Telephone Laboratories, Incorporated
Murray Hill, New Jersey

INTRODUCTION

Darwin is a game of survival and (hopefully) evolution played by two or more people, using the 7090. Each player devises a program or programs; the programs of any one player constitute a species. All the players' programs are simultaneously in a section of core known as the arena. Whichever program is currently in control tries

a) to replicate itself in an unoccupied part of the arena

b) as a means to this end, to destroy members of other species by attacking their vulnerable locations.

If the program in control attacks a ''protected'' location of another program, control passes to the program which was attacked.

Outside the arena there is an umpire, which keeps track of who occupies what parts of the playing area, and which cells are protected. The umpire informs the organisms in the arena about the status of any cell in the arena, and switches control if a question is asked about protected storage.

Each game continues until either time runs out or only one species remains. If the members of one species manage to destroy all members of all other species, the player who coded the surviving species wins.

At the conclusion of each game, all programs used by all players are made available for study, to allow the players to alter their strategies in an intelligent fashion for the next game.

The detailed rules of Darwin are such that it can be played on high speed express runs.

RULES FOR DARWIN

A *species* consists of a species number N, a size S(N), an origin O(N), and twenty protected locations P1(N), ..., P20(N). Each player may choose S(N), O(N), P1(N), ..., P20(N) arbitrarily, subject only to the constraints that each of O(N), P1(N), ..., P20(N) shall be less than S(N), and S(N) shall be less than 2000.

An organism of species N is a program which satisfies the following rules:

1) It occupies S(N) locations, L, ..., L+S(N)-1.

2) It has a single entry point L + O(N).

3) It may not look inside the umpire, or elsewhere outside the arena.

4) It may not alter storage locations outside its own block without first notifying the umpire.

5) Whenever it calls the umpire for any reason, the organism identifies itself by its species number, N.

6) The organism may not make use of input-output, or of any traps or trapping modes.

An organism may be any program whatsoever which obeys these restrictions. It may employ any desired computational procedures, may look at any cell in the arena, including cells occupied by members of other species. A player is entitled to make the members of his species communicate or cooperate in any desired fashion, or he may make them ignore each other. Different members of a species need not be the same; there may be as many different subspecies as desired, having in common only N, S(N), O(N), P1(N), ..., P20(N).

At the beginning of play each player is entitled to have in the arena the largest number of organisms which will fit into the lesser of 2000 or A/2M locations, where A is the size of the arena and M is the number of players. The umpire will assign actual storage locations to the organisms by a pseudo-random algorithm, and will transfer control initially to the first organism of minimal size.

Players are expected to debug their species before, and separately from, the actual game. A wild transfer, a program loop, or the illegal destruction of the contents of cells outside the organism currently in control shall lose the game for the responsible player.