Physica 10D (1984) 75–80
North-Holland, Amsterdam

EXPLOITING REGULARITIES IN LARGE CELLULAR SPACES

R.Wm. GOSPER
Symbolics Inc, 845 Page Mill Road, Palo Alto, California 94304, and Lawrence Livermore National Laboratory, Livermore, California 94550, USA

We informally describe a computer algorithm for simulating deterministic cellular automata. It grows "smarter" by selectively recording intermediate computations while operating on a compressed representation of the cellular space-time. No information is lost because configurations do not actually evolve. Instead, the simulator accomodates random exploration of place-times in the future of initial configurations embedded in an effectively unbounded void.

We use Conway's Life automaton as a model, but the idea extends easily to other geometries, dimensions, and number of states.

1. Motivation

In 1970, J.H. Conway announced a particularly interesting cellular automaton which he dubbed "Life" [1–3]. Life is a two state, nine-neighborhood rule applied on an ordinary, two-dimensional grid [4]. Early manual and computer simulations suggested that initially finite configurations of *on* cells remained finite. This led Conway to conjecture the non-existence of such things as "puffer trains", his name for configurations which would travel indefinitely through empty space leaving permanent debris.

Fig. 1. shows four stages of what seems to be a puffer train. Every ten time steps, the topmost three components advance five cells vertically, and left-for-right reflect themselves, while the center component emits a puff of debris (indicated by an arrow). Thus, in ten more time steps, the configuration will try to emit a mirror image puff five cells forward of the previous puff. This would obviously be a puffer train, were the puffs not themselves vigorously interacting, threatening to spawn a disturbance which overwhelms the puffer from behind.

Is fig. 1 a puffer train?

There is probably no way to settle such questions except by experiment. Run the simulation until either the puffing stops, or the debris exhibits enough regularity for determinism to furnish a proof of unbounded growth. (The proven Turing-universality of Life on infinite grids introduces a third, fortunately remote possibility: the configuration may neither regularize nor self-distruct, indefinitely concealing its ultimate behavior.)

Fig. 2 shows that fig. 1 is indeed a puffer train, by virtue of the debris plume achieving both temporal and spatial periodicity, but only after some 2000 steps and a sevenfold multiplication of the original puff period. Subsequent experiments on similar puffer train candidates have shown period doubling to be far likelier than septupling. A few configurations, however, have puffed for several hundred steps, only to disintegrate without establishing an overall period, thereby underscoring the necessity of experimental proof of apparent unbounded growth.

And should one turn from naturalist to engineer, building large configurations of modules which construct or compute things [3], one should still experiment to verify that the "hardware" works.

In either case, the experiments will usually require a large and unpredictable amount of "real estate" and time, yet most of this real estate will repeat itself most of the time.

The following algorithm will let us freely explore the future space-times of large initial con-

0167-2789/84/$03.00

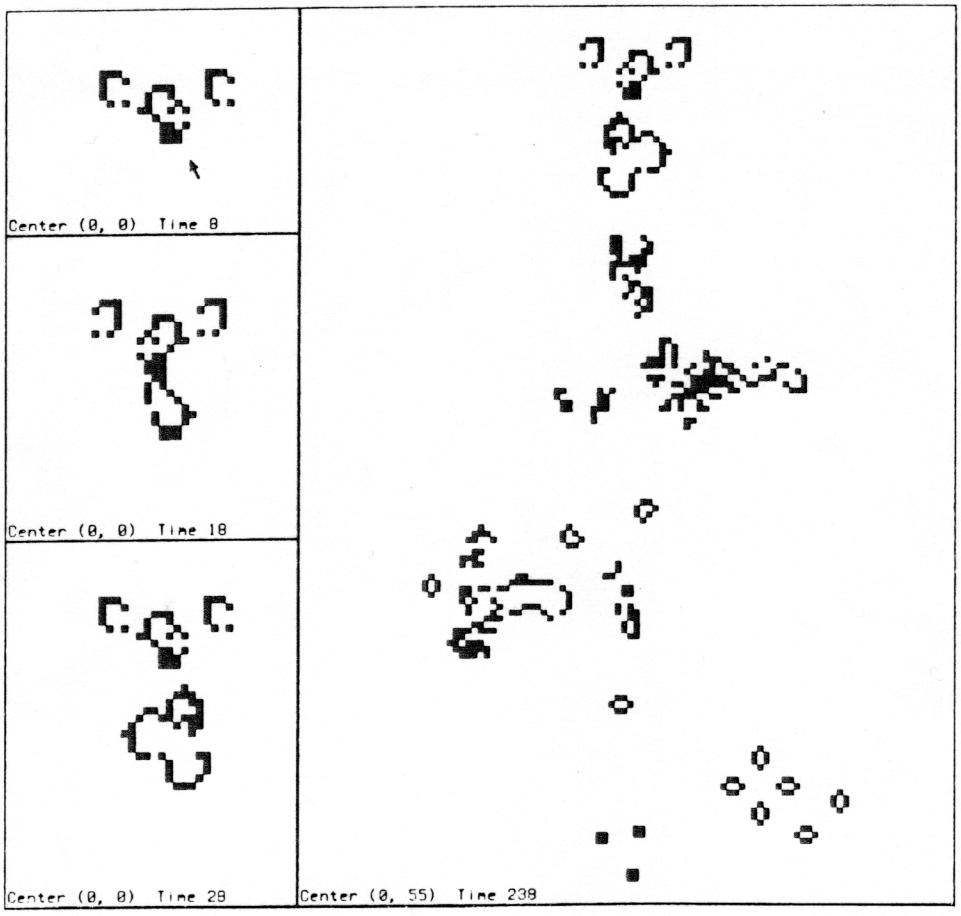

Fig. 1. Four stages in the evolution of a probable puffer train.

figurations, provided that they are sufficiently repetitious, both structurally and behaviorally. It (and enough computational hardware to support it) could have spared Conway one of his few unlucky conjectures.

2. The algorithm

There are two key components – a hash mechanism and macro-cells. The hash mechanism prevents the recomputation of indistinguishable scenarios. Macro-cells mechanize the information-compression of the spacetime behavior of configurations. A macro-cell represents a 2^n by 2^n block of automaton cells, where n is any non-negative integer. Each macro-cell seeks to determine its RESULT, namely the concentric 2^{n-1} by 2^{n-1} macro-cell which the parent macro-cell exclusively determines after 2^{n-2} time steps. But instead of occupying 2^{2n} units of storage, a macro-cell of size 2^n ($n > 0$) requires just five units. These hold (pointers to) the four macro-cells of size 2^{n-1} which comprise the four quadrants, and, if we are lucky (and $n > 1$), the RESULT, also of quadrant size.

The entire structure and evolution of an initial configuration will be encoded in the interlinkings of macro-cells, which are computed as we probe its future. This will usually require fewer macro-cells than you might think, due to two restrictions on when a macro-cell can be created. First, a macro-cell is never created if one having the same quad-

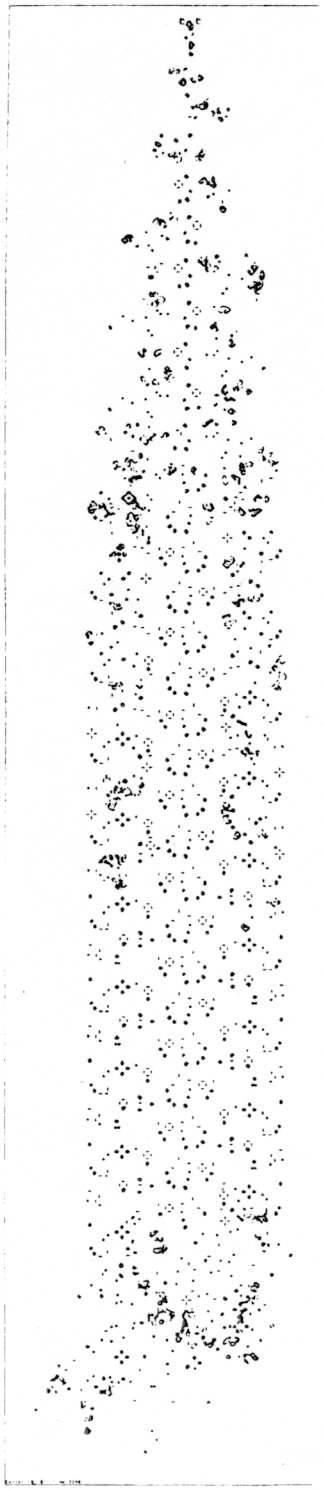

Fig. 2. Above the still-smoldering initial explosion (bottom), there finally appears a segment of temporally and spatially periodic exhaust, proving that above this segment operates a true puffer train [5].

rants already exists. This applies recursively to the quadrants. At the bottom of the recursion are the 2^0 by 2^0 (i.e. 1 by 1) cells, of which there are at most two, since Life is a two state automaton. Likewise, at most $2^{2^2} = 16$ two by two cells can be created, etc. When the algorithm tries to group four quadrants to form a pre-existing macro-cell, the hash mechanism notices the coincidence and returns the old cell instead of a new one. Most importantly, this old cell may already know its RESULT, while the new one cannot. (Ironically, the "old" cell may well have been created later in *simulated* time than the "new" one.)

The second restriction on macro-cell creation is implicit in the algorithm: a macro-cell of size 2^n, $(n \geqslant 2)$ can only be created when its x, y, and time coordinates (relative to all its parent macro-cells) are multiples of 2^{n-2}. Thus, proliferations of cells is limited by indistinguishability when they are small, and by infrequency of creation when they are large.

As an example, an empty universe of diameter 2^{32} is represented by a macro-cell whose four quadrants are all the same, a macro-cell of diameter 2^{31}, etc. Thus, the whole thing requires only 33 macro-cells. Since the representations of the "big" ones are no larger than the "small" ones, one can be casual about creating new universes, as long as one reuses a lot of lower-level components. Thus if one wishes to "edit" an existing 2^{32} by 2^{32} universe by altering a single 1 by 1 cell, one must create a whole new universe. But this can entail no more than 33 new cells (at most one of each size), no matter how extensive the original configuration.

Similarly, simulation rates can increase exponentially in regions of repetitive behavior, because of the reuse of many RESULTs in the construction of larger RESULTs (which represent larger time-steps).

The smallest macro-cell which can have a RESULT is 4 by 4. These and all larger cells produce RESULTS in response to messages, which usually issue from larger macro-cells in pursuit of their own RESULTs. In order to minimize storage requirements of individual cells, message handling code is associated with size classes of macro-cells,

rather than with the cells themselves. If the queried macro-cell already knows its RESULT (from having computed it previously), it just returns it. If a 4 by 4 cell doesn't know its RESULT, it computes it by brute force, i.e. by applying the Life rule [4] to the nine-neighborhoods of each of its four central cells (bits). Larger cells determine their RESULTs by a somewhat inelegant-looking recursion which involves combining a total of thirteen separate RESULTs of quadrants, and other quadrant-sized macro-cells formed by grouping RESULTS and regrouping quarants of quadrants.

Here is a mental picture of the recursion. A macro-cell at time 0 is the top stratum of a patch of earth. Successively deeper strata hold future time-slices. We are trying to excavate down to the RESULT stratum, which is the flat bottom of a hole with sides of slope 1, a frustum of the foreseeable future cone with the initial macro-cell as its base. (Actually, the future cone is the exterior of the light cone of the exterior of the initial macro-cell.) We daren't excavate any steeper, lest we unearth the uncertainty that seeps sideways from the edges at slope 1. The depth of the hole is $\frac{1}{4}$ of its width, and half the width of the bottom. To scoop out such a hole, we recursively scoop out holes of the next smaller size, i.e. one half the desired diameter and depth. By taking the RESULTs of the four quadrants, we reach half of the desired depth, but there remain dikes covering 5/9 of this halfway bottom (see fig. 3). To excavate these dikes, five artificial, shifted "quadrants" must be constructed from quadrants' quadrants, and then RESULTed. This will involve the reexcavation of some thin air, but at little cost, since the reexcavated cells will remember their RESULTs. (If they didn't, the recursion would actually recompute the same results a number of times proportional to the diameter of the outermost macro-cell!) We are now on the halfway bottom, composed of nine subresults, which are then grouped in fours to form four overlapping squares. The grouping of the RESULTs of these four squares is the grand RESULT.

Thirteen scoopings is not too inefficient, given that each scoop is only 1/8 the volume of the desired excavation.

Note that the algorithm is indifferent to the x, y, and time coordinates and even the sizes of the macro-cells. Their spatial coordinates are implicit in the quadrant structure of their owners, and their time coordinates are implicit in the RESULT structure. Nothing is even checking whether a RESULT is to be computed by counting (4 by 4), or by recursion (>4 by 4), since the knowledge of how to RESULT is in the cell classes themselves. The 16 by 16s send the same RESULT message to their 8 by 8s as the 8 by 8s send to their 4 by 4s, without any knowledge of their own size or the size of their quadrants. There are 4 by 4s in the quadrants of the 8 by 8s by construction, and a properly constructed universe will stay that way.

"Digital physicists" i.e. seekers of cellular models of the real Universe, may be interested in this algorithm's decoupling of simulated time from simulator time, thus freeing their models from the irritating requirement of "pan-synchronicity", and giving time a rather more topological interpretation. But this very virtue becomes a drawback when the simulator desires to view the progress of the simulation as if it were evolving flatly and parallel to simulator time. One approach is to artificially maintain x and y coordinates throughout the recursion, and display RESULTs in their proper position whenever they are obtained. This is especially tempting, in that it might have the additional virtue of skipping over scenarios that have been played before. Alas, the result, at least if viewed flat, is incomprehensible. Time not only runs at different speeds in different places, it even appears to run backwards in fixed places. This comes from the reexcavation of "thin air" described earlier, wherein overlapping events are explored several times. A possible fix might be to "ratchet" time by maintaining, for each I by I cell position, the largest time value reached. But this would prevent the simulation of those very large but repetitive configurations for which this algorithm is well suited. What's worse, the ratcheting leaves an unflat time surface which still results in

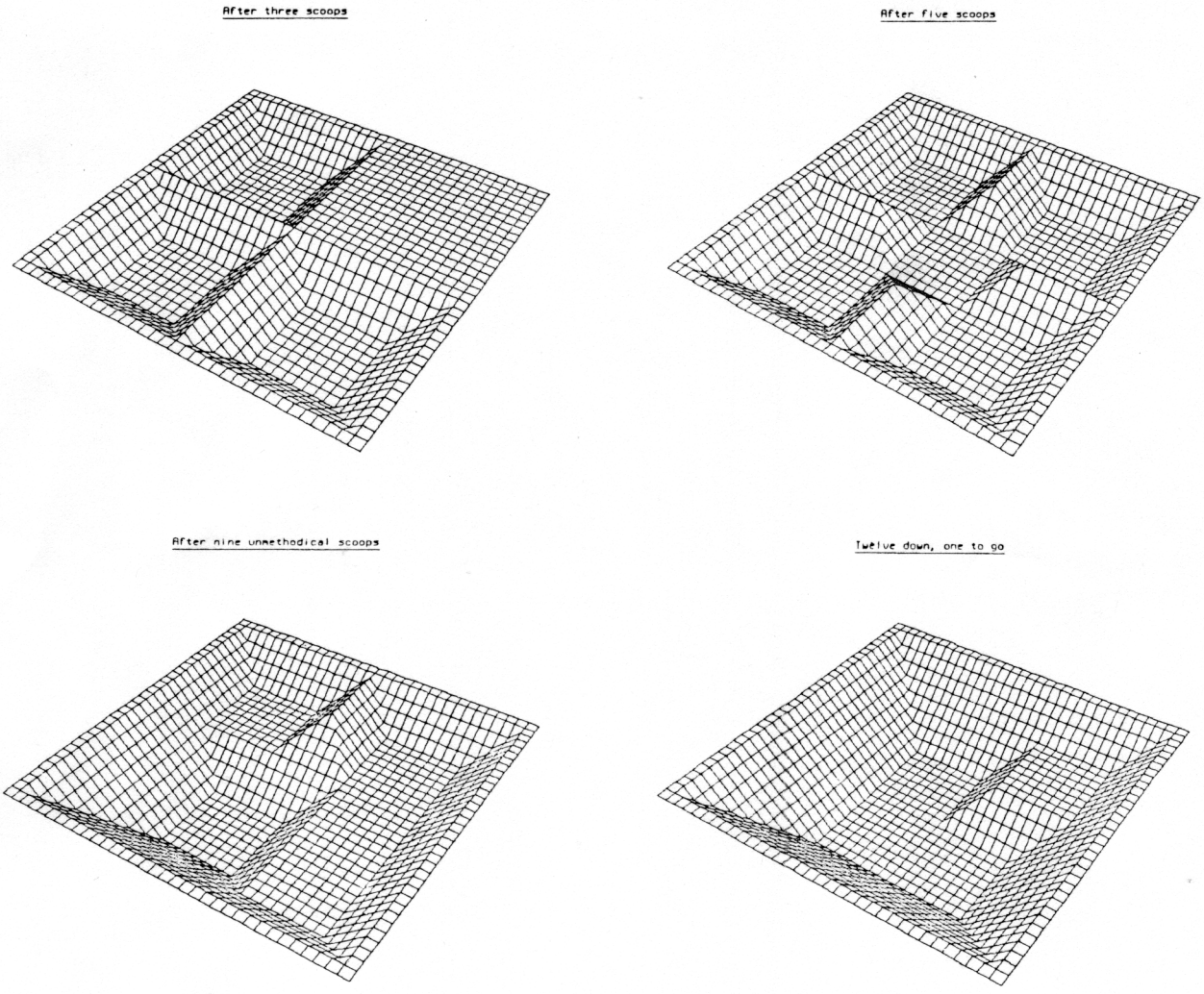

Fig. 3. The recursion step: digging a hole in spacetime with half-size scoops. Greater depths represent greater time. The thin, circumferential flange indicating time stratum 0 extends into neighboring macro-cells.

confusing displays. For example, a traveling oscillator may simply disappear for an extended time, only to reappear simultaneously in several places, due to multiple intersections of the oscillator's straight track with the terraced walls of the partial excavation.

The resolution of this drawback is to let the observation drive the computing, instead of vice versa. In general, one wishes to have one or more "windows" into the spacetime, that is, (probably) rectangular slabs of 1 by 1 cells, one time unit thick. To SHOW the intersection of such a slab with the spacetime, teach the macro-cell classes to check whether their future cone intersects the slab, and if so, propagate the SHOW message, along with appropriate x, y, and time offsets, to the quadrants and RESULTs, which are computed as necessary. Then teach the 1 by 1s to signal the querying window if they get a SHOW message with time and space coordinates all 0. Although significantly more complicated than the RESULT algorithm, this SHOW algorithm is more efficient than it may sound. If no RESULTs need to be computed, SHOW is only logarithmic in the time coordinate.

Furthermore, most future cones (and, by the geometry, the cones of all their components, recursively,) will not intersect non-gigantic windows.

However, the outermost SHOW method ensures that the configuration being probed is surrounded by enough (iteratively doubled) vacuum so that its future cone entirely contains the probe window, no matter how large or remote in space or time. Thus, there are never any edge effects.

Another virtue of the macro-cell approach is that a large SHOW or RESULT computation which is aborted before completion is not wasted, for all of the intermediate RESULTs have been permanently recorded in their owners. Thus, a restarted computation will very quickly regain the state where it broke off. Likewise, independent experiments on the sequence of similar configurations may accelerate as the hash-table accumulates "smart" macro-cells.

Direct generalizations of the RESULT mechanism enable the geometric transformation, aging, and merging of large configurations, as well as extensive replications in arbitrary grids.

3. Conclusion

Even in simulating such an unpredictable and irreversible automaton as Life, considerable economies are possible. By attributing similar thriftiness to whatever implements our own reality, our (simulated) imaginations may be stimulated.

Acknowledgements

I found the RESULT algorithm while at the XEROX Palo Alto Research Center. Its only implementation so far has been in the exceptionally powerful Flavor Lisp mechanism devised by Howard Cannon (of Symbolics, Inc.). This is also true of the subsequently discovered display, geometric transformation, aging, merging, and grid replication algorithms. John Lamping of Stanford University saved me much work by suggesting a binary quadrant structure in place of the ternary structure in my original plan. An early model Symbolics 3600 ran the puffer train of figs. 1 and 2 several million steps, at a rate which doubled every two or three minutes, once the initial explosion settled into oscillation.

Fig. 3 was plotted by MIT Macsyma.

References

[1] M. Gardner, Sci. Amer. 223, No. 4 (1970) 120; 224, No. 2 (1971) 112.

[2] Berlekamp, Conway, & Guy, Winning Ways, Academic Press, Vol. 2, 817–849. Contains design of a Turing machine in Life.

[3] M. Gardner, Wheels, Life and Other Mathematical Amusements, (W.H. Freeman, San Francisco, 1983).

[4] The Life rule: If two of your eight nearest neighbors are *on*, don't change. If three are *on*, turn *on*. Otherwise, turn *off*.

[5] The initial explosion, actually just a puff reacting against the vacuum instead of previous puffs, eventually decays to a harmless cloud of oscillators (of periods one, two and three) at step number 5533.