Finding life in the shadows

David H. Ackley[1] and Elena S. Ackley[2]

[1]University of New Mexico, Albuquerque, NM 87131
[2]Ackleyshack LLC, Placitas, NM 87043
ackley@cs.unm.edu

Abstract

Artificial chemistries hold promise for demonstrating and studying open-ended evolution (OEE), in part, because they are typically engineered 'bottom-up'—emphasizing primitive types that "interact well" in multiple contexts, rather than directly implementing some single given function. A price of that generality is that the living 'agents' or 'components' typically involve substantial configurations of primitives, and indeed it may be unclear when, where, or at what level(s) life or evolution is even happening. A basic OEE research challenge, therefore, is to develop methods for finding living and evolving components by observing a system's chemical dynamics "from the outside". As a case study, this paper demonstrates analytic techniques for identifying and tracking living components, using a version of the "*C21*" digital protocell which has been crudely modified to display autonomous growth, fission and fusion, and death. In this model those operations are randomly triggered, leaving the system (we believe!) free of significant evolutionary dynamics, so we can explore it as a neutral model or "shadow" as proposed by Bedau et al. (1998). Using image files captured during the artificial chemistry simulations, we recover selected atomic types and positions, then derive protocell sizes and positions, then infer key "life events" such as cellular fission and fusion, allowing us to infer components, component movements over time, and "family trees". Even in this simple case of all-but accidental life, without evolutionary activity, we find rich dynamics within and among the identified components.

Open-ended evolution vs top down design

"Top-down" engineering design begins with some required goal and seeks physical realizations to achieve it, while optimizing against some set of constraints. "Bottom-up" engineering, by contrast, begins with available physical capabilities and components and seeks increasingly useful configurations of them. Though both approaches often have roles in complex systems engineering, they can differ greatly in the character of their results, with top-down tending toward greater efficiency and fitness for purpose, while bottom-up yields greater resilience and reusability.

If a system is to be capable of open-ended evolution (OEE), it seems likely, or perhaps inevitable, that whatever passes for a high-level goal or "fitness function" within the system will itself change over time. Bottom-up approaches may therefore have an advantage, since even their 'base level' evolutionary components—living agents both as physical objects and as dynamical processes—are already built from smaller and reusable parts and mechanisms.

In artificial life—at least of the "soft alife" variety Bedau 2003—the "agent" that will be the star of the proceedings is often designed top-down. The programmer directly controls and represents all internal details of the agent—such as their size and size limits, what actions are possible and their effects, or what counts as genomic information along with its size and limits. Though this simplifies and accelerates model design, it also makes the agent's internal structures "opaque", in the sense that there are no interactions among those structures *except* those explicitly defined by the programmer. Unless it is explicitly programmed, there is no chance that an agent might suffer internal damage, no chance it might acquire a duplicate organ, no chance it might incorporate something from the environment, and so on.

Artificial chemistries (Banzhaf and Yamamoto, 2015, is a primary reference) are the poster children for bottom-up engineering in soft artificial life. Here, the programmer typically has direct control over atomic size and molecular shapes and limits, over what reactions may occur and their results. Then as an extra step, a "biological" level—the agents, their internal structures, their genetic information, and so forth—is defined by the programmer as an initial condition, rather than directly as "laws of physics".

In this case, whatever atoms and reactions have been taken as primitive, they have already been 'tuned' to work together at least sufficiently to define (at least an "ancestral" version of) the living agent under study. Moreover, because an agent's "organs" and internal processes are explicit within the model, and operating according to the same programmer-chosen laws of physics, radical alterations of what even constitutes an agent become possible.

It is true that even "top-down agents" can display such alterations and reinterpretations at the level of interacting agent populations, unlike in an artificial chemistry—but still, no such general interaction abilities were required in the ini-

tial top-down design. Open-ended evolution is a challenging and multifaceted concept, but artificial chemistries with bottom-up design at least cut directly to the chase.

Finding the agents in the data

At the first OEE meeting in York (Taylor et al., 2016), the first author proposed the goal of identifying hallmarks of evolution "from the outside"—via observations of a volume of space-time, without using any particular knowledge of the system(s) operating within that volume. Specifically, this "OEE research challenge" (Ackley, 2015) was presented:

> Develop a statistical method that, for a given discrete window of space-time-granularity,
>
> 1. Identifies potential evolutionary components via *near perfect* spatial autocorrelation, then
> 2. Infers 'life lines' by connecting spatiotemporally adjoining potential components
> 3. Eliminates time by projecting the life lines onto a phase space defined by component size, then
> 4. Assesses life line evolution by measuring distance travelled, compared to a random walk, in that space

This paper exploits knowledge of the *C211* membrane to finesse the general case of step 1, and focuses primarily on step 2. Using only external system observations of our artificial chemistry simulator, we attempt to locate and track all the "biological components" in the system, across space, across time, and across potentially evolutionary events like component splitting, merging, and death. We present a case study of one simulation run that spanned over 16,000 sites with an average of 600,000 events per site occuring during the simulation. Four initially-seeded protocells lead to hundreds by the time we stopped the run, and, depending on their parameter settings our tracking algorithms automatically recover all or nearly all of the "intuitively obvious" events that a human would perceive looking at the same data.

The evolutionary inference procedure

Figure 1 provides a high-level view of the entire evolutionary inference process, which operates in four phases. Starting from screen image files generated by the `mfms` artificial chemistry simulator, we sample the displayed grid of sites and identify key atomic types based on the colors found (Figure 1a and 1b). Note that only some atomic types can be uniquely identified by this process, and many details of internal atomic state are likewise unobservable, but the crucial `InnerMembrane` (`IM`) and `OuterMembrane` (`OM`) atoms have no internal state, and are displayed in unambiguous colors. In addition to the membrane types, we identify empty sites by the occurrence of background colors, and everything else is lumped into one 'Other' category, which in this case consists of the `Content` atoms that reside within the membranes.

Topological inference phase

Following atomic recovery, the second phase scans the grid to infer protocell membranes and contents (Figure 1c). Assuming the grid is *membrane consistent* as defined in Ackley (2018), any non-membrane site that is adjacent to an `IM` is necessarily inside that membrane, so when our topological inference scan first encounters such a site, it performs a flood fill to label all other sites that are internal to the same membrane.

A single complete grid scan, with flood fills as needed, suffices to identify all distinct closed membranes with their insides, and temporary labels are assigned to them as they are encountered. (Note that if membrane consistency is violated, such distinct labeling is in general not possible—but at least this phase will detect such violations.)

Temporal inference phase

The third phase (Figure 1d) compares the previously identified protocells ("Then" or T nodes) with the current crop ('Now" or N nodes), attempting simultaneously to explain where all the latter came from and all the former went. First, a matrix of spatial intersections is computed between all pairs of T and N protocells, and each pair produces two potential directed edges, one from Then to Now and one the reverse. Each edge is assigned a percent weight by dividing the intersection size by the size of protocell emitting the edge; T and N nodes are typically different sizes causing the two edge weights to be unequal even though their intersection is symmetric. In Figure 1d, for example, the `Ta`→`Na` edge has a higher weight than the `Na`→`Ta` edge because `Ta` is smaller (192 sites for `Ta` vs 221 for `Na`, with an intersection of 187, though that is far from evident in the figure).

The non-zero-weight edges are sorted into decreasing order and used to construct a directed bipartite *temporal transition graph* between Then and Now nodes. The edges are considered one at a time, and an edge is inserted into the graph if it is *useful*, where

$$\text{useful}(E) \equiv \text{out}(\text{from}(E)) = 0 \vee \text{in}(\text{to}(E)) = 0.$$

If an edge is not useful it is discarded; either way the edge consideration process continues until either there are no more edges or all nodes have been *used*, where

$$\text{used}(X) \equiv \text{out}(X) > 0 \vee \text{in}(X) > 1.$$

When all nodes are used, the graph construction process terminates, even if some edges remain uninserted. In Figure 1d, for example, the top three edges suffice to use `Ta`, `Tc`, `Td`, and `Nc`—but `Na` (with $\text{out}(Na) = 0 \wedge \text{in}(Na) = 1$) remains unused. After adding the next three edges, however, all nodes are used and graph construction is complete.

Evolutionary inference

Now working solely from the temporal transition graph, the final phase of the process is aimed at two goals, the first

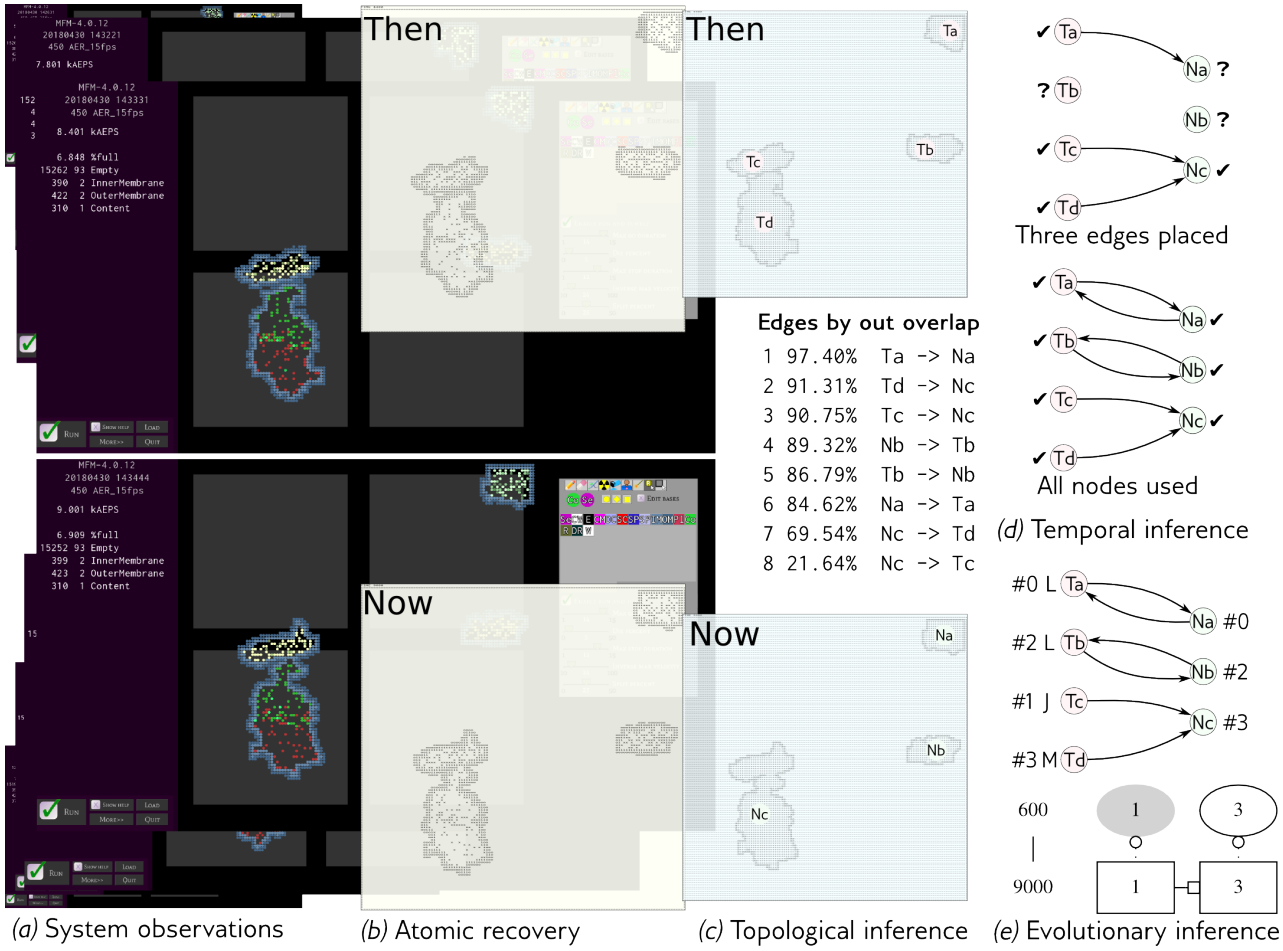


Figure 1: Overview of evolutionary inference process. *(a)* Screen captures from the `mfms` simulator provide crude but regular system observations. *(b)* Key atom types are located by color, after a once-per-run manual geometry calibration. *(c)* Closed protocell contours are inferred assuming membrane consistency, and assigned temporary labels. *(d)* Intersections are computed between prior and current protocells, and sorted by the percent of outbound cell sites in each intersection. Corresponding directed edges are added to a temporal transition graph until all nodes have been used. *(e)* Finally, *evolutionary events* (e.g., 'L'ive another day, 'J'oin with a larger cell in a 'M'erger) are inferred, and numeric *cell IDs* are carried forward (or created) as needed, creating protocell *lifelines* from which multigenerational 'family trees' and other statistics can be derived. See text.

of which is to infer *evolutionary events* for each transition between Then and Now. Overall, we identify five possible events, discussed below. The second goal is to create protocell *lifelines* by managing numeric "cell ID" labels that remain stable not just across one transition but across the entire life of a protocell, though it may move and split and merge over time. From the evolutionary events along the lifeline of any given cell, we can produce statistics and visualizations like protocell lifetimes, offspring count, family trees and so on.

Each of our five evolutionary events is defined by a characteristic *graph signature* specified in terms of node connectivity and their $in()$ and $out()$ degrees, as summarized in Table 1. Note that although the intuitions behind the signatures

Event	*Temporal transition graph signature*
Live	$T \leftrightarrow N \wedge \text{in}(N) = 1 \wedge \text{in}(T) = 1$
Split	$T \leftarrow N \wedge \text{out}(N) = 1 \wedge \text{in}(T) > 1$
Merge	$\text{in}(N) > 1$
Genesis	$\text{in}(N) = 0 \wedge \text{out}(N) = 0$
Death	$\text{in}(T) = 0 \wedge \text{out}(T) = 0$

Table 1: Evolutionary events and their signatures. Signatures depend on unidirectional or bidirectional node connectivity, represented by arrows, and the indegrees and outdegrees of the graph nodes. See text.

are reasonably obvious, and empirically they seem to work well enough in our as-yet limited testing, we do not have a clear theoretical framework to explain *why* these signatures are effective in any crisp way. Other and better approaches may well be waiting to be found.

In a **Live** event a current protocell N is connected in both directions to some prior protocell T, while no other edges are directed at either N or T. The detection of a **Live** event provides a story for both the prior and current nodes, and the durable cell ID assigned to T is transferred to N (as in the transfer of cell ID `#2` from `Tb` to `Nb` in Figure 1e).

The **Live** event represents the typical case of a protocell persisting over time, moving about and changing size and shape—but only by a small amount from Then to Now. Note how the graph construction procedure helps make this signature work: Although in complex situations it is common for some prior cells (especially large ones) to overlap *unrelated* current cells, such overlaps are typically small and so their associated edges are unlikely be useful by the time they are considered (if they are considered at all).

A protocell **Split** is detected when N points at one T but that T points back at more than one Now node. Recognizing Splits via $\mathrm{in}(T)$ takes advantage of the fact that the intersection of each 'splittee' with its presplit parent will be a huge fraction of the splittee's size, even though perhaps only a modest fraction of the parent's. Note also that—although it will typically be rare with fine-grained analyses—there is nothing preventing a protocell from splitting into more than two pieces at once, and we have found this signature works well even in that case. When a split occurs, the presplit cell in T receives a 'P'arent event, and its cell ID is transferred to the largest resulting protocell, while the other split offspring receive new cell IDs drawn from a globally-incrementing counter, with a 'S'plit as their first event.

In a similar, if time-reversed, fashion, a protocell **Merge** event is declared when multiple Ts point at the same N so $\mathrm{in}(N) > 1$. (It certainly seems there should be more fully-symmetric **Split** and **Merge** signatures that would do at least as well as those in Table 1, but that is future work.)

In the **Merge** event depicted in Figure 1e, `Tc` and `Td` merge into `Nc`. `Td`, the larger premerge protocell, is assigned a 'M'erge event, and its `#3` cell ID is transferred to `Nc`. `Tc`, by contrast, receives a terminal 'J'oin event, and its `#1` cell ID is then retired.

Finally, **Genesis** and **Death** events are declared when an N or T node, respectively, has no edges at all. Note that neither event can occur unless the graph construction algorithm ran out of edges, since the isolated nodes will perpetually be considered unused. This raises a concern that some low-weight edges might creep in and somehow disrupt signature recognitions, but so far that has not been observed. A 'G'enesis event acquires, and a 'D'eath event retires, one durable cell ID in the obvious fashion.

To complete one step of the evolutionary inference procedure, all the above phases are performed (taking an empty set for T on the first step), and the resulting evolutionary events are appended to cell data files named by their durable cell IDs. Finally, the N nodes become the T nodes, and the next observation is analyzed to produce a new N. We illustrate how it works concretely in the next section.

Going beyond *C211*: Easy to be hard

The report on the *C211* cell presented in the main conference (Ackley, 2018) had its pages full just covering the basic mechanisms of the cell membrane and the programming language and software engineering work behind its discovery. During that protocell development, of course, possible methods of using it as a basis for evolving artifical life systems were always in our mind, and as the design began to stabilize we did make a few exploratory stabs at adding autonomous growth and reproduction to *C211*.

Given that we had demonstrated initial growth from a seed, plus coordinated protocell movement triggered by randomly-generated "`Commander`" atoms, our unspoken thoughts about protocell reproduction ran something like this: *How about a command atom that says: Each* `Content` *should copy itself once, then send the copy and original in prechosen opposite directions? Once the groups separate enough the membrane will naturally fission and we'll have two protocells! With no evolution, how hard could it be?*

Of course, eventually we did try it, and soon started to appreciate how hard it could indeed be. But even though nothing worked as planned, basically the first grossly-wrong idea we tried was already quite fascinating to watch—and is the basis of the case study in this paper.

For starters, since the simplest thing to do was add a '`Reproduce`' atom to the set of commands our (now-modified) *C211* cell would randomly choose to deploy, that's what we did. Of course, with this approach there is no obvious feedback or regulatory mechanism to limit the occurrence of `Reproduce` commands. It turned out not to be quite true, but as far as we knew at the outset, all the protocells would attempt to reproduce forever.

But more immediate issues arose long before we had to face that problem. Issues we thought minor triggered a cascading series of disruptions leading to a major meltdown, typically along these lines:

- After gossiping a reproduction order, the `Content` originals generally managed to duplicate successfully, and the mass of the protocell soon doubled. However—especially because free space had been reduced by the doubling—the originals and copies would often block each other as they tried to move in opposite directions. As a result,

- The two groups began to get so spread out that their communications were increasingly disrupted, and disconnected `Content` subgroups began to form. As a result,

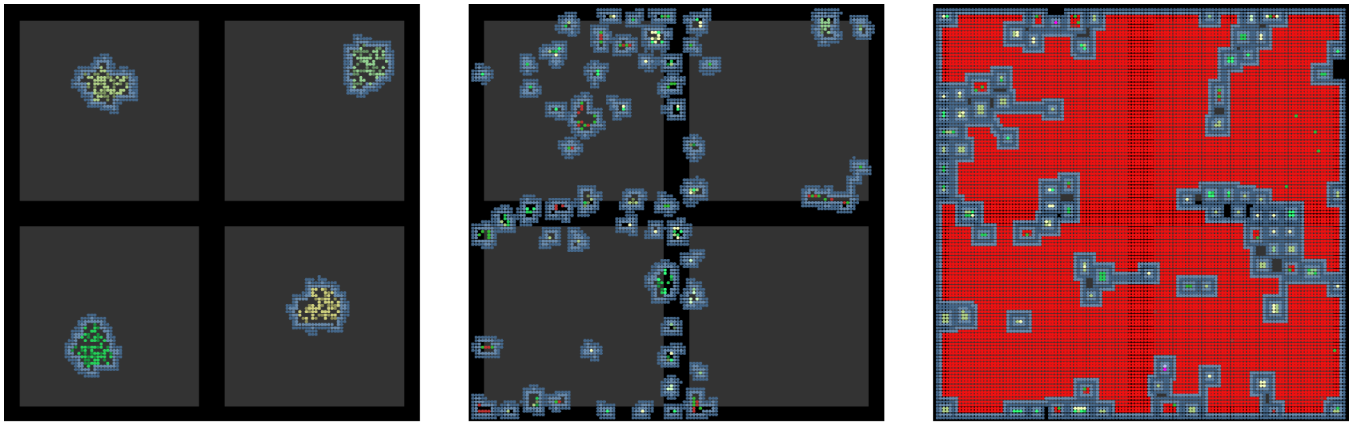


Figure 2: Early, middle, and late stages of the case study simulation run. *(Left):* On a simulated grid consisting of four separately-processing but interconnected hardware tiles *(dark grey squares)*, four protocells seeded via 'Genesis' events have grown to maturity after 300 Average Events Per Site *(300 AEPS = 0.3 kAEPS)*. *(Center):* By 300 kAEPS, mostly botched reproduction attempts have left the grid littered with scores of tiny—but still active—"miniprotocells." Cells often cluster near hardware tile boundaries because intertile overheads tend to slow the event rates in those regions. Cells displaying red/green `Content` are generally attempting to reproduce. One such protocell—that insignificant near-rectangle all the way down in the lower-left corner—happens to have cell ID `#73`. *(Right):* By the time the simulation was stopped at 600 kAEPS, cell ID `#73` has conquered the grid, consuming all accessible space, and completely engulfing the four groups of protocells not touching a grid edge. Barring hardware errors or external perturbations, it is believed this is a stable simulation configuration. See text.

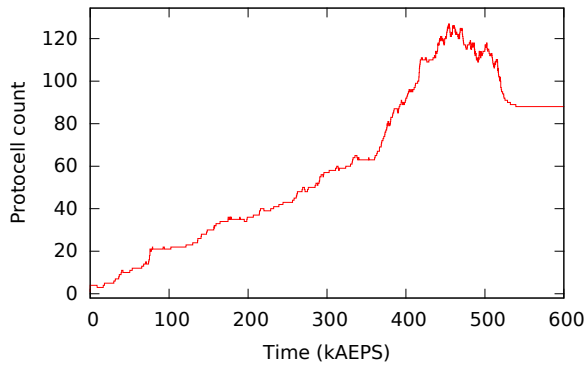


Figure 3: Number of living protocells vs time.

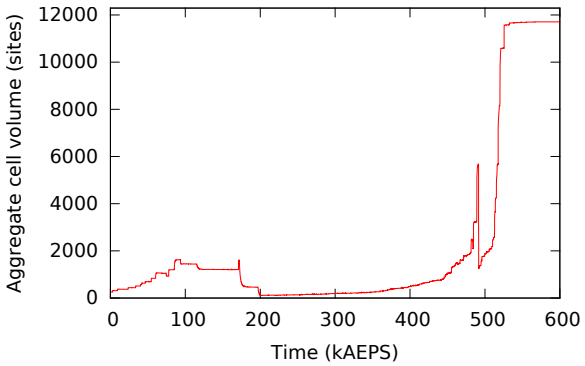


Figure 4: Total protocell size vs time.

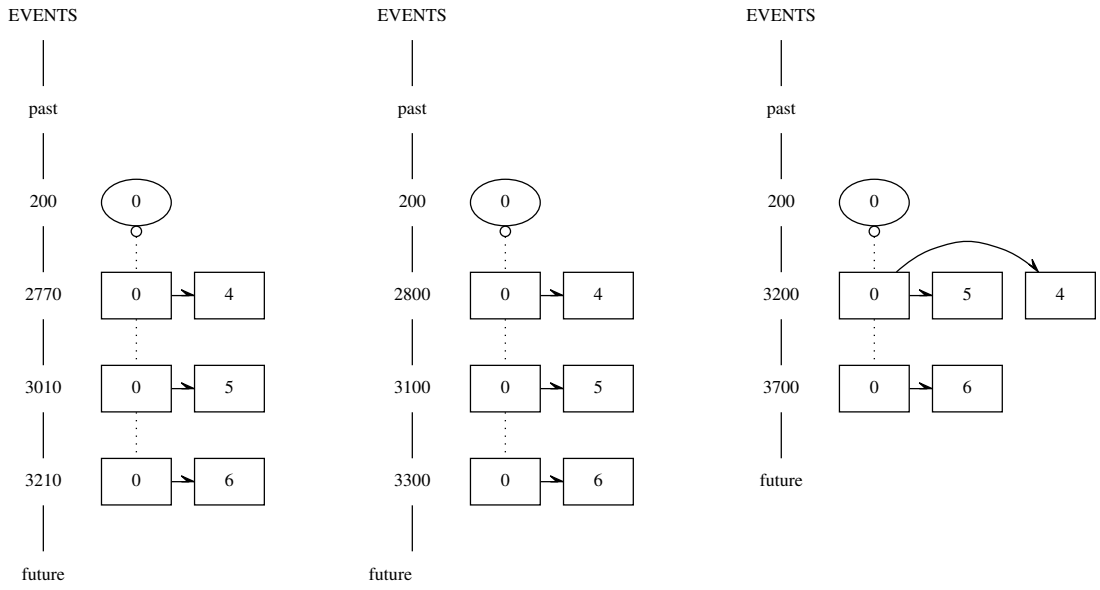
- The `Content` in a single protocell stopped acting in unison and started issuing conflicting orders, eventually with inconsistent command priorities.

The lives of cancerous protocells

Once multiple non-dominated orders begin circulating within protocells, basically chaos rules—but chaos of a particularly specific and life-like form. Protocells continue to move and stop, to grow and split, to merge, and to die, without any sort of regulation other than limited available space.

Figure 2 illustrates three sample observations of the simulation run we have to date studied the most. While it is debatable at best whether there's any sense or utility in these protocell dynamics, at least there is a *lot* of dynamics. Running on an old laptop, we ultimately let this simulation run for over a week, curious to find out if or when a stable state would be reached (discussed below).

Figure 3 plots the number of protocells versus simulation time, while Figure 4 depicts the total volume of all living protocells. The biomass collapse visible in Figure 4 shortly before 500 kAEPS was due to a few massive protocell deaths—including cell ID `#107`, which reached a peak volume over 1,000 sites around 483 kAEPS, only to die barely 1 kAEPS later. On the other hand, the resulting free space stimulated a small uptick in the protocell count soon afterwards, visible in Figure 3.



(a) Observing at 10 AEPS *(b)* Observing at 100 AEPS *(c)* Observing at 500 AEPS

Figure 5: The same simulation data analyzed at three different temporal resolutions. See text.

The subjectivity of evolutionary events

A slightly subtle point about our evolutionary inference process is that the inferred evolutionary events and derived family trees can vary depending on the granularity and phase we chose for system observations. To minimize analysis time it is preferable for the interval between Then and Now to be large—hundreds or perhaps thousands of AEPS. On the other hand, an excessive observation delays can introduce significant artifacts into the evolutionary inference process. In Figure 5, for example—using data from an earlier run than our case study for this paper—the same three split events not only occur at different times depending on the granularity of analysis, they shift from being sequential events to appearing simultaneous. Similarly, or even worse, excessive observational delay can cause loss of tracking on a fast-moving protocell, causing what "should" be a **Live** event to be rendered as a **Death + Genesis** pair, and thus unnecessarily breaking a cell lineage.

Discussion and conclusion

Cell ID #73's march towards grid domination is not particularly visible in the aggregate and average views of Figures 3 or 4. Its impact is hinted at in the declining protocell count with the rising protocell volume above 500 kAEPS, but its towering influence is much more visible in other—more individualized—visualizations. Figure 6, for example, plots only the initial portion of #73's descendant family tree, as the entire lineage is far too large to display for a page.

Stability = death

As suggested by the caption to Figure 2, we believe the final configuration of this simulation run is a stable state—and, mostly, we feel that is yet another bug in this hacked-up *C211* model. Because of the specific way that collective `Content` commands are mediated by `Commander` atoms—as discussed above and in Ackley (2018)—it turns out that *No commands can be given without an empty space*, which is needed to deploy the desired `Commander` atom.

Once cell ID #73 achieved absolutely fully-packed status—not just internally but also across the grid as a whole—no further internal dynamics could ever change that. Even a **Death** event is a just another type of `Commander`, so without an available empty site it cannot be deployed. Although a few empty sites do remain, they *seem* to be in configurations that the *C211* membrane will never choose to expand into. Indeed, most of the changes that occurred during the 500 kAEPS era were tiny position adjustments of miniprotocells, packing them more tightly and liberating a few empty sites that were soon snapped up by cell ID #73—who didn't, as it happened, choose to deploy a **Death** `Commander` into any of them.

Although reaching a stable state is often valued as a way of ascribing an overall "meaning" or "output" for a dynamical or computational process, here it seems quite clearly like a bug. We should view life as an operating system and not an algorithm; freezing solid should never be the only move.

Life in the shadows

For the soft alife OEE enterprise to succeed in a satisfying way, we believe it is essential that we gain experience and wisdom defining living components from increasingly purely observational data

These crazy broken cancerous protocells are among the best exemplars we have yet encountered of Tim Taylor's observation that artificial life should be something you can't stop watching. Even with no evolution whatever, these simulations simply generate a tremendous amount of *plot*. Of course, the narrative power of birth, living, and death is shared by all alife models, but here—in the grip of a rich and largely intuitive spatialized artificial chemistry implementing the biological mechanisms—unexpected but retrospectively satisfying *plot twists* seem to occur frequently, with life and death sometimes plainly hanging on the least shift of an atom.

We look forward to undertaking seriously the design of controllable reproduction and death for a future protocell in the *C211* lineage, but we do not regret the we have time spent developing membranes without reproduction, and reproduction without evolution. We suggest that perhaps this is what it looks like to design systems so that its primitives work well together.

Perhaps the most direct route to open-ended evolution begins with no evolution at all.

Acknowledgments

This work was supported in part by grant VSUNM201401 from VanDyke Software.

Author contributions

DHA designed and directed the research and wrote most of the text. ESA made text revisions, implemented much of the evolutionary inference processing code, gathered data, and produced graphs and statistics. DHA once again failed to save time to do any kind of justice to related work before the initial submission deadline.

References

Ackley, D. (2015). Indefinite scalability for open-ended evolution. Talk at the *OEE Workshop* at Alife 2016, video retrieved May 2018 from `https://youtu.be/a5FlbkB0vvo?t=439`.

Ackley, D. H. (2018). Digital protocells with dynamic size, position, and topology. In *Proceedings of the 2018 Conference on Artificial Life*, Tokyo, Japan. in press.

Banzhaf, W. and Yamamoto, L. (2015). *Artificial Chemistries*. MIT Press.

Bedau, M. A. (2003). Artificial life: organization, adaptation and complexity from the bottom up. *Trends in Cognitive Sciences*, 7(11):505 – 512.

Bedau, M. A., Snyder, E., and Packard, N. H. (1998). A classification of long-term evolutionary dynamics. In *Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life*, pages 228–237. MIT Press.

Taylor, T., Bedau, M., Channon, A., Ackley, D., Banzhaf, W., Beslon, G., Dolson, E., Froese, T., Hickinbotham, S., Ikegami, T., McMullin, B., Packard, N., Rasmussen, S., Virgo, N., Agmon, E., Clark, E., McGregor, S., Ofria, C., Ropella, G., Spector, L., Stanley, K. O., Stanton, A., Timperley, C., Vostinar, A., and Wiser, M. (2016). Open-ended evolution: Perspectives from the OEE workshop in York. *Artif. Life*, 22(3):408–423.

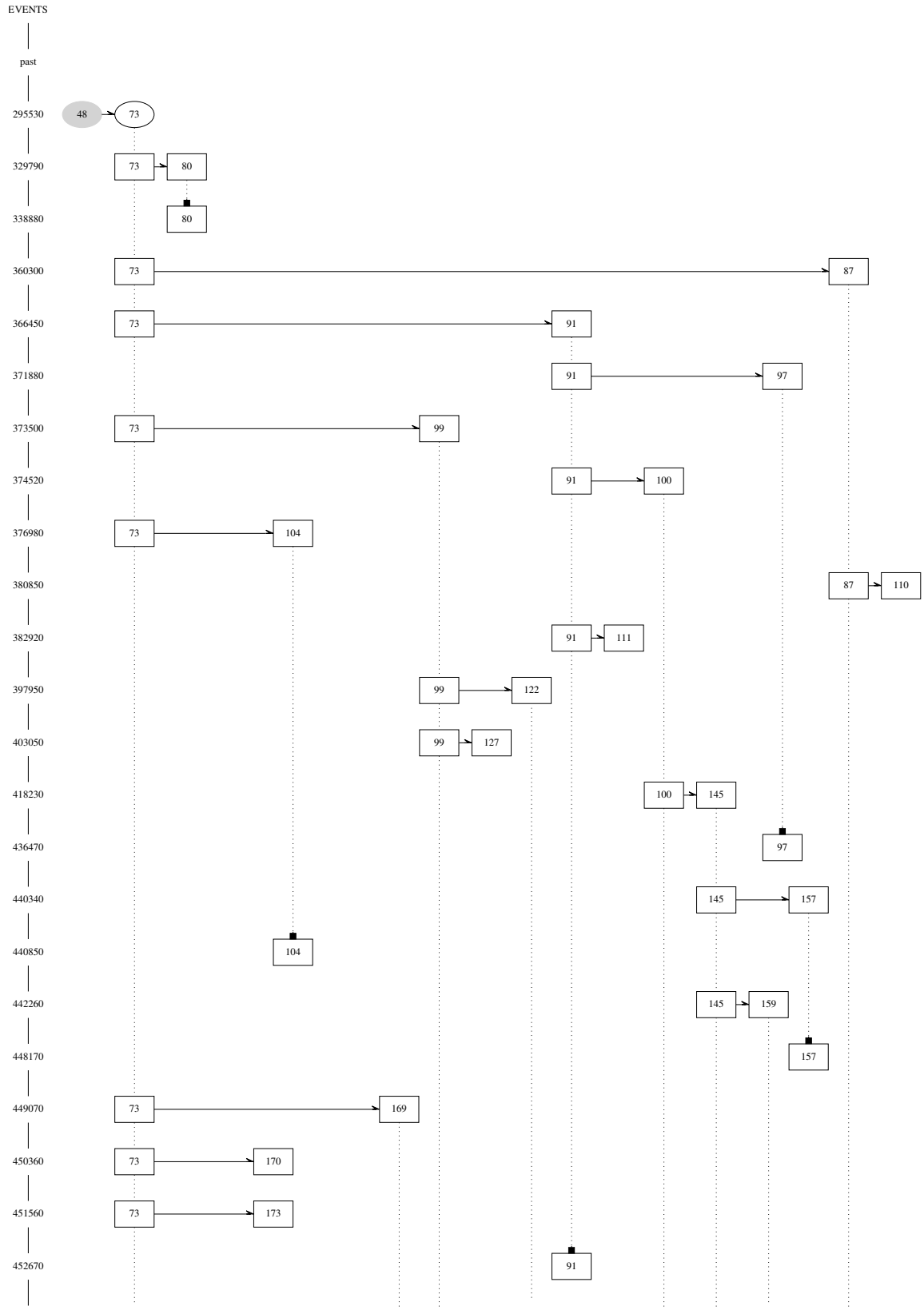


Figure 6: The descendant tree of cell 73 (initial portion only). Vertical position represents time in AEPS increasing down the page. The cell ID in the open oval is the subject of the tree; the shaded oval indicates that subject's immediate origin. Plain dotted lines represent **Live** events collapsed over time. A black square box indicates the **Death** event of a cell ID. An open circle (not seen in this figure but visible in Figure 1e) represents a **Genesis** event. An open square box (also visible in Figure 1e) depicts a **Merge** event, linking the Join ID to the Merge ID. A solid line with half-arrowhead depicts a **Split** event, pointing from the "parent" to a smaller offspring. Finally, a live box with no outgoing lines (e.g., `110`, `170`) means that cell ID remained alive but had no further events before simulation end. See text.