The Organic Builder: A Public Experiment in Artificial Chemistries and Self-Replication

Tim J. Hutton
Department of Computer Science,
University College London, UK
tim.hutton@gmail.com

February 21, 2007

Abstract

We describe some results submitted by users of the Organic Builder, a Java applet where the rules of an artificial chemistry can be chosen in order to achieve a desired behaviour. While initially intended as a set of challenges to be tackled, as a game, the users experimented with the system far beyond this and discovered several novel forms of self-replicators. When searching for a system with certain properties such as self-replication, making the system accessible to the public through a website is an unusual but effective way of making scientific discoveries, credit for which must go to the users themselves for their tireless experimentation and innovation.

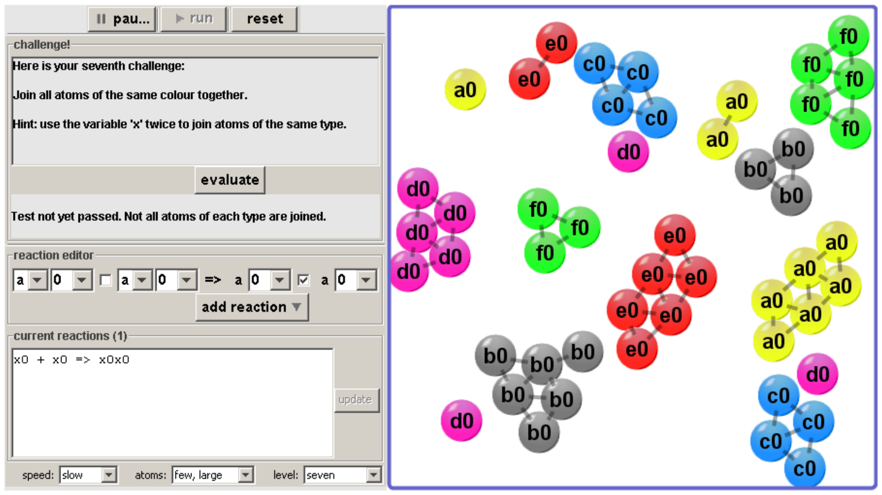
1 Introduction

The Organic Builder applet (Fig. 1) was conceived as a user-friendly environment in which the rules of an artificial chemistry could be experimented with by non-programmers. Artificial chemistries [1] can be thought of as cellular automata (CA) where the cells can move around, and this movement makes certain tasks such as self-replication arguably easier than in CA because the replicators can make use of the greater self-organization properties of the medium. However, while there are many accessible programs and websites for experimenting with cellular automata[1], for artificial chemistries (AChems) there are almost none. The Organic Builder was created to fill this gap.

There are many types of AChem (see [1]) with a wide variety of representations for the atoms and reaction rules. Here we have adopted a very simple scheme where each atom has a fixed type (`a-f`) and a changeable state (`0`, `1`, `2` ...), and reactions that involve two of them, eg. $\texttt{a3} + \texttt{b4} \rightarrow \texttt{a6b6}$. These reactions can bond or unbond the atoms, and change their states, and this is sufficient for much complex behaviour including template replication [2]. Variables x and y represent unknown types.

In the Organic Builder, the atoms are represented as circles moving around in a two-dimensional area. A simple spring physics causes them to bounce off each other and the walls while not stretching bonds too far. Two atoms can react when they collide with each other. Typically the reaction rules are chosen at the start of an experiment, and left unchanged throughout. A 'reset' button restores the states and bonds of the atoms to an initial configuration and randomizes their positions and velocities, allowing the experiment to

[1]For links, see: `http://cell-auto.com/links/`



Here is your seventh challenge:

Join all atoms of the same colour together.

Hint: use the variable 'x' twice to join atoms of the same type.

Test not yet passed. Not all atoms of each type are joined.

```
x0 + x0 => x0x0
```

speed: slow atoms: few, large level: seven

Figure 1: The Organic Builder interface, with a description of the challenge in the top-left, controls for designing the reactions in the bottom-left and the reaction chamber on the right. A solution to challenge seven is shown *(left)*: $x\mathtt{0} + x\mathtt{0} \rightarrow x\mathtt{0}x\mathtt{0}$ that joins atoms of the same type. The atoms *(right)* have not yet all bonded.

be re-run. While this random element means that no two experiments proceed in an identical fashion, the reaction pathways can be sufficiently controlled such that a similar endpoint is reached.

It is somewhat hard to appreciate the full range of possibilities of an AChem like this, and so a set of challenges were created as learning exercises, leading the user through the basics of bonding and unbonding and onto more advanced topics such as methods for membrane growth and template replication. While some of the details of how these work are specific to the form of AChem adopted, the general techniques are seen in other AChems and in biochemistry. One concept that is present in biochemistry but noticeably absent in our AChem is that of molecular shape which together with atomic charge determines which reactions take place and at what rate. In our AChem these concepts are approximately replaced with the notion of a numerical state which is used to achieve the same effect in a computationally-efficient manner.

In the next section we briefly look at the challenges that have been created for the Organic Builder, and how often they have been solved. In sections 3 and 4 we present two novel forms of self-replicator created by users, for naked molecules and cells respectively.

2 The challenges

The challenges are designed as a learning experience, with increasing difficulty. They are also intended as a hands-on demonstration of some of the engineering possibilities of artificial chemistries, even one as simple as this. Table 1 shows the instructions for each challenge, and also lists the smallest number of reactions needed to solve it.

The user enters the reactions desired, and sets the experiment running. Whenever they think they have succeeded, they can hit the 'evaluate' button and the applet will check the state of the atoms in the reaction chamber. If all the conditions are satisfied, the reactions specified are accepted and logged as a solution. This form of world-state evaluation limits the range of challenges that can be set to those with a specific outcome but this is sufficient for now.

Intentionally there is no method for editing the states or types or bonds of the atoms in the reaction chamber directly - they can only change through the application of the reaction rules. Thus there is no way for the user to 'cheat' and simply build a specific world.

The Organic Builder was first announced on the artificial life mailing list[2] in September 2005 with the initial creation of 15 challenge levels. A further announcement was made to the same list in December 2005 when four more challenges were created. The Organic Builder has not been widely advertised beyond this. Table 1 shows the number of times that each challenge has been solved between the time of this initial release and January 2007, the time of going to press - a total of 4085 times. These solutions came from 427 unique IP addresses, although there are likely fewer users than this because some users will visit from multiple IP addresses.

#	Instructions	Reactions required	Times solved (unique IPs)
1	Join all the '`a`' atoms up.	1	507 (361)
2	Make '`ec`' pairs.	1	371 (274)
3	Join all the '`c`' atoms up into a long line.	1	315 (240)
4	Join every atom together.	1	316 (239)
5	Connect the two fixed atoms.	2	326 (235)
6	Make '`abcdef`' chains.	5	263 (205)
7	Join all atoms of the same colour together.	1	228 (179)
8	Join a matching atom to each atom in the template.	1	231 (179)
9	Break the molecule in half.	1	238 (189)
10	Bond the trapped '`f`' atom with another '`f`'.	2	239 (184)
11	Pass a message down the molecule.	1	233 (179)
12	Split the molecule into two equal halves.	2	228 (152)
13	Insert a '`b`' atom into the middle of the molecule, in between the `e2` and the `e3`.	3	206 (156)
14	Join matching atoms to the template and join them up neatly like a ladder, or DNA.	5?	123 (92)
15	Make a free-floating copy of the molecule.	9?	75 (57)
16	Grow the membrane to incorporate all available '`a`' atoms.	3	73 (66)
17	Get all the '`e`' atoms out of the membrane and move all the '`f`' atoms inside. Leave the '`b`' atom inside.	12?	43 (36)
18	Divide the cell into two completely separate daughter cells.	6?	52 (45)
19	Make a free-floating copy of the cell.	114?	11 (7)

Table 1: The challenges in the Organic Builder and the number of times they have been solved. Later challenges are harder, which may explain the drop-off in solutions found. To date challenges were solved a total of 4085 times, from 427 unique IP-addresses.

[2]alife-announce: `http://lists.idyll.org/listinfo/alife-announce`

Challenges 1, 2, 4, 6 and 7 begin from an initial configuration that has all atoms unbonded and in state `0`. Other challenges have different starting configurations - challenge 3 provides one '`c`' atom in state `1`, for example, to start the chain off. Challenges 8, 14, 15 and 19 have a starting configuration that includes a chain of atoms `a-d` that must be matched with atoms of the same type. In these levels, the sequence of atoms is different every time, to enforce generic solutions rather than ones that will only work on a specific sequence. Challenges 10, 16, 17, 18 and 19 have an initial configuration that includes a loop of atoms - the physics prevents other atoms from passing through this loop thus forming an impermeable membrane. Only by interacting with the membrane by opening channels and so on can these challenges be solved. Additionally, challenge 19 introduces a foreign body into the reaction chamber, for the purpose of illustrating why a membrane might be a necessary structure in different environments. For full details, it is best to experiment with the Organic Builder itself.

Table 1 includes a column showing the number of reactions that are required to solve each challenge. For the more difficult challenges it is not known what the minimum number is and so a question mark is added. For challenge 19 in particular there are solutions that will only work some of the time, depending on what atoms happen to collide, and while these are accepted by the applet as solutions (when they work) they are obviously less satisfactory. The quoted number of 114 reactions for challenge 19 is for a solution that works reliably, and this solution is discussed in more detail in section 4.

The self-replication schemes discussed in sections 3 and 4 were created in October 2005 and late December 2005 respectively, in both cases mere days after their respective problems, 15 and 19, were released.

3 Self-replication of naked molecules using two-reactant reactions without interference

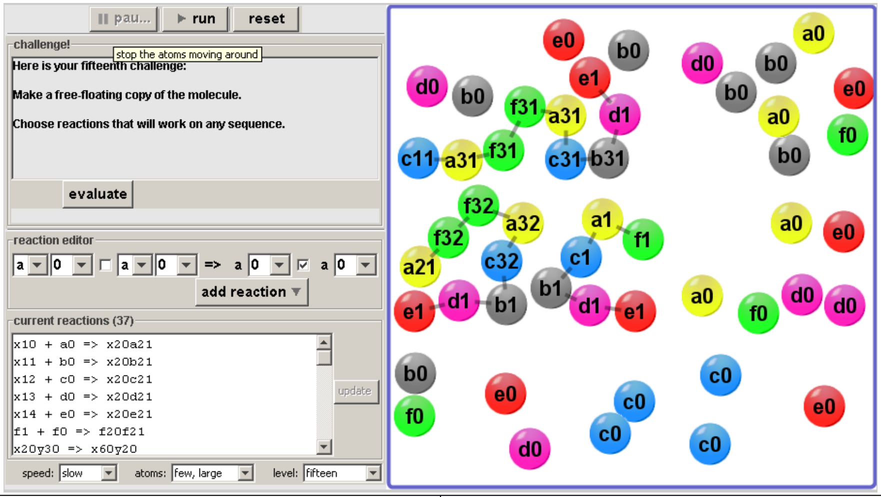
In [2] a set of two-reactant reactions (of the form described above) were shown that caused template replication of a chain of artificial atoms of different types. Repeated replication caused the reaction chamber to become full of copies of the initial molecule. The set of reactions used had a 'flaw' that while replication would be 100% reliable for the first molecule, when there were two or more molecules replicating near to each other, they would sometimes react with each other. The tangled molecules would eventually separate but only having exchanged part of each other, and the most obvious side-effect of this was that sometimes molecules of different length would appear.

Under conditions of periodic partial replenishment, any shorter molecules would tend to replicate at a faster rate and thus the long-term effect was for the molecules to evolve down to the shortest possible length, just two bases long. While this was an interesting result since mutation was appearing 'for free', it meant that the mutation rate could not be controlled experimentally being instead fixed at a high value.

To avoid this interference problem in further work, either three-reactant reactions were adopted [3] or cell membranes were included [4, 5]. It was thought that reliable template replication was not possible in this form of AChem with only two-reactant reactions.

Remarkably, however, through experimentation with the Organic Builder, Dave Mann[3] succeeded in doing just this, creating a set of two-reactant reactions that allow repeated replication of naked molecules without any problem of reliability. Instead of bonding across in a ladder shape, like DNA, his molecules pass signals down their length specifying what atom type should be bonded at the end. Figure 2 lists the reactions and shows some molecules in action within the Organic Builder interface.

[3]`dave.mann@ntlworld.com`



Here is your fifteenth challenge:

Make a free-floating copy of the molecule.

Choose reactions that will work on any sequence.

current reactions (37)

```
x10 + a0 => x20a21
x11 + b0 => x20b21
x12 + c0 => x20c21
x13 + d0 => x20d21
x14 + e0 => x20e21
f1 + f0 => f20f21
x20y30 => x60y20
```

speed: slow   atoms: few, large   level: fifteen

```
// grab atom of correct type
x10 + a0 => x20a21
x11 + b0 => x20b21
x12 + c0 => x20c21
x13 + d0 => x20d21
x14 + e0 => x20e21
f1 + f0 => f20f21
// send up message:  atom has been
// grabbed
x20y30 => x60y20
x20y31 => x60y20
x20y32 => x60y20
x20y33 => x60y20
x20y34 => x60y20
x20y35 => x60y20
// found next uncopied atom, so send down
// its type
x20a1 => x30a30
x20b1 => x31b31
x20c1 => x32c32
x20d1 => x33d33
x20e1 => x34e34
x20f1 => x35f35
// copy 'type' message down the molecule
x60y30 => x30y30
x60y31 => x31y31
x60y32 => x32y32
x60y33 => x33y33
x60y34 => x34y34
x60y35 => x35y35
// message has reached bottom of
// molecule, set required atom type
// (logic loops back to here until an 'e'
// has attached)
x21y30 => x10y30
x21y31 => x11y31
x21y32 => x12y32
x21y33 => x13y33
x21y34 => x14y34
//x21y35 => x15y35
// detect that copy is complete
// 2 step delay to give previous message
// time to propagate
e21x60 => e22x60
e22x60 => e23x60
// send 'copy complete' message
x23y60 => x2y23
// separate copy from original
f2f2 => f3 + f3
// tell everything to reset to state 1
x3y2=>x3y3
e3x3=>e1x3
x1y3=>x1y1
// make sure last pair always reset
e20x23 => e2x2
```

Figure 2: Molecular replication in the Organic Builder applet using Dave Mann's set of reactions (below). These reactions use the state information to pass messages saying what the next atom type to be bonded should be, allowing the molecule to grow lengthwise instead of in a ladder shape. For example, the `c11` atom is waiting to encounter a `b0` atom (reaction 2) as that is next in the sequence `facbde` that is the molecule being replicated. Lines beginning with '//' are comments.

These replicators are reminiscent of the CA ‘worms’ of [7][4] with spare atoms playing the role of the grid of cells. See also [11] for related work that additionally describes the potential benefits of a ‘fluidity of site arrangements’ that AChems offer over CA.

This form of self-replication might be useful in other artificial chemistries since it will work whether the world is two- or three-dimensional, and under different representations of atoms and reactions. While more costly in terms of the number of reactions (37) and states required (20) than the work presented in [2] (8 reactions, 10 states), the fact that it works reliably using only two-reactant reactions is significant.

4 Cellular reproduction in the presence of a caustic agent

In this section we discuss a full solution to challenge 19, where the task is to copy an entire ‘cell’ including a strip of atoms in an arbitrary order - this is randomised at the start of the experiment to assure compliance.

A foreign body is present in the reaction chamber in this challenge with the property that when it bump into an atom it will break all of its bonds unless the atoms is of type ‘`a`’, thus requiring the cell contents to be contained within a membrane of `a`’s throughout. This solution was created by Ralph Hartley[5] and builds a special ‘mouth’ structure (Fig. 3) that ingests a particular atom only if not disturbed by the caustic agent. It achieves this by detecting if the ‘food’ particle is still bonded as intended or has been unbonded. The 114 reactions required for this process are listed in Table 2.

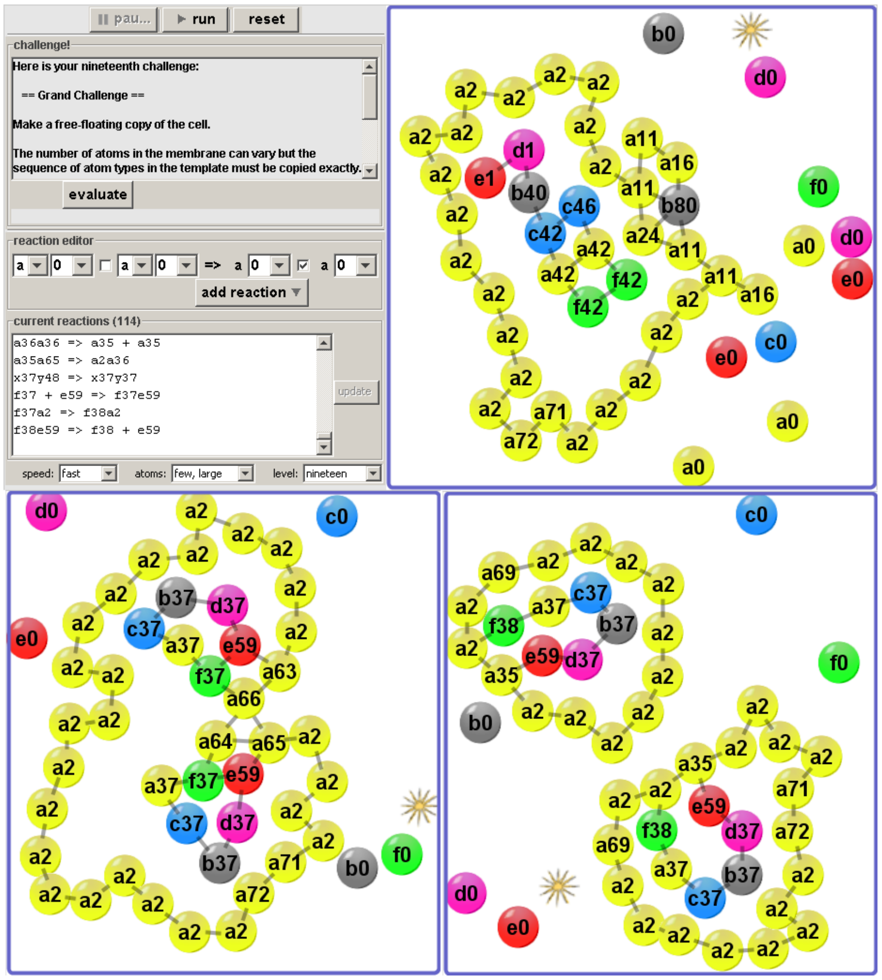
Using a dedicated mouth structure in this way allows the cell not only to protect itself from the caustic agent but also to ingest only those particles that are required for the cell to reproduce (specified from inside), improving its efficiency. Cell membranes in nature have transport channels that perform similar roles, either signalling across the membrane or allowing certain molecules to pass through in one direction or both.

While not required for solving challenge 19, the cell would be capable of repeated reproduction with slight modification, if sufficient space and material resources were available. Questions of evolvability are beyond the scope of the Organic Builder in its current incarnation but might be considered in future versions.

AChems that considered self-reproducing cells include [9, 6, 4, 5] - see [8, 10] for a discussion of why membranes are considered to be an important component of life. While AChems for self- and information-reproducing cells have been published previously [4, 5], the scheme described here does not require the membrane to be naturally permeable as was the case in those systems. The added complexity required to achieve this means that the number of reactions required is much higher (114 vs. 34) but the design of a mouth structure is precisely the sort of complex adaptation that evolution could work upon and in theory create in the first place. If on-going evolutionary activity is the goal, we do not want to design-in the very type of innovation that we would wish to appear - equally it is important to know that such structures are possible within the constraints of the system. The cell reproduction reactions discussed here achieve this.

[4]A Java applet of similar work can be found at `http://alife.co.uk/revoworms/`

[5]`hartley@aic.nrl.navy.mil`



Here is your nineteenth challenge:

== Grand Challenge ==

Make a free-floating copy of the cell.

The number of atoms in the membrane can vary but the sequence of atom types in the template must be copied exactly.

```
a36a36 => a35 + a35
a35a65 => a2a36
x37y48 => x37y37
f37 + e59 => f37e59
f37a2 => f38a2
f38e59 => f38 + e59
```

speed: fast atoms: few, large level: nineteen

Figure 3: Ralph Hartley's cell reproduction sequence. The cell forms a 'mouth' *(top)* that ingests whatever atom is required next (here a 'b' atom). The caustic agent (spiky) may unbond the 'b' atom at this point but this will be detected by the cell. After the necessary atoms are ingested and the atom-sequence copied, the two copies attach to the membrane *(bottom left)* and the cell division process starts. Eventually the cells separate completely *(bottom right)*.

```
// The import function is the only hard part.
// Import atoms in the presence of a caustic.
// Works like a lock, with inner and outer gates
// that cannot both be open simultaneously.
// It is important that the lock not be large
// enough to enclose both an atom and the
// caustic simultaneously.
// --- States (not complete) ---
// x0     - external atom, bound or unbound.
// a4-a8  - building gate
// a10    - outer gate, open or opening
// a11    - outer gate, closing
// a12    - outer gate, closed
// a15    - end of outer gate, open or opening
// a16    - end of outer gate, closing
// a17    - end of outer gate, closed
// a20    - inner gate, both gates closed.
// a21    - inner gate, opening
// a22    - inner gate closed opening outer gate
// a23    - inner gate closed opening outer gate
// a24    - inner gate, closed, after binding atom
// a25    - inner gate, open
// a26    - inner gate, open, bound to outer gate
// a27    - inner gate, closed, bound to outer gate
// a28    - inner gate, closed, waiting to bind
//          atom for import
// a29    - inner gate frame, open
// x80    - atom to be imported (hopefully bound
//          to gate)
// x43    - imported atom in interior
// Build the outer gate
a3e1 => a4 + e1
a4a2 => a8a6
a6a2 => a10a7
// The first one is always an f
a8a2 => a34a5
a5 + a0 => a10a7
a7 + a0 => a10a15
// Bind an atom for import
// The atom must be of the requested type
a30 + b0 => a24b80
a31 + c0 => a24c80
a32 + d0 => a24d80
a33 + e0 => a24e80
a34 + f0 => a24f80
// If the whole gate binds to the atom the
// gate will close faster. The mechanism will still
// work without this, with many fewer reactions,
// but the rate will be *much* lower, with the
// caustic causing more restarts.
a11 + x80 => a11x80
a16 + x80 => a16x80
// The above two reactions involve a non-A on the
// outside. So there is no point in keeping track of
// them. Just clean up all the possible states and
// race conditions (unless I missed some)
a2 + x80 => a2+x0
a16x0=>a16+x0
a17x80=>a17+x0
a17x0=>a17+x0
a17x43=>a17+x43
a17x46=>a17+x46
a11x0=>a11+x0
a12x80=>a12+x0
a12x0=>a12+x0
a12x43=>a12+x43
a10x43=>a10+x43
a12x45=>a12+x45
a10x45=>a10+x45
a12x46=>a12+x46
a10x46=>a10+x46
a2x48=>a2+x48
a28x0=>a28+x0
// Close outer gate around the bound atom
a24a10 => a24a11
a11a10 => a11a11
a11a15 => a11a16
a16 + a16 => a17a17
// Propagate message that outer gate is closed
a17a11 => a17a12
a12a11 => a12a12
a12a24 => a12a20
// The moment of truth.
// The atom might have escaped before the gate
// closed. Check that it is still present and bound.
// We have to check, because if an atom is not
// enclosed the caustic might be, and if it is not
// bound it might be the wrong atom.
a20x0 => a21x43
a20x80 => a21x43
// We need to give up eventually, and restart the
// import cycle from the beginning, in case the
// lock is empty (or worse, contains the caustic)
a20 + x1 => a22 + x1
a20 + x42 => a22 + x42
// Open outer gate
a22a12->a23a10
a23a12->a28a10
a10a12=>a10a10
a10a17=>a10a15
a15a15=>a15+a15
// Open inner gate
a21a12 => a90a91
a90a12 => a92a91
a91a2 => a93a2
a92a93 => a94a12
a94a91 => a95 + a29
// Hold the imported atom (and don’t restart
// the import cycle) until the requester knows that
// it has arrived.
a95x44 => a25 + x45
// Close inner gate. Binding the gates together while
// inner gate is closing prevents the lock from
// exporting an atom, and increases the rate.
a25+a17=>a26a17
a26+a29=>a27a12
a27a17=>a22+a17
// Add A’s to membrane
// wait until after first import
// so we don’t steal the materials for the
// import mechanism.
x45+a3 => x45+a70
x46+a3 => x46+a70
x41+a3 => x41+a70
x42+a3 => x42+a70
a70a2 => a71a72
a71 + a0 => a73a74
a74 + a72 => a72a76
a76a73 => a2 + a71
// Copy the String (start with the f)
f1a3 => f40 + a3
// Request that the import lock get the needed atom.
// If the import aborts, we will need to make the
// request again.
a28+b40=>a30+b40
a28+c40=>a31+c40
a28+d40=>a32+d40
a28+e40=>a33+e40
a28+f40=>a34+f40
// Import succeeded. Don’t renew the order, and let
// the gate release it.
x40 + y43 => x41+y44
// Start the new child strand.
f41 + f45 => f42f46
// Add the atom. We don’t need to check its type,
// because only the correct atom was imported.
x46 + y45 => x42y47
x47 + y41 => x46y42
x42y1=>x42y40
// A is a special case, get it from the membrane
// growth site instead of the import lock.
a40+a71 => a41a77
a77a2=>a77a79
a79+a72=>a81a82
a81a77=>a71+a83
a82a83=>a72+a44
a41a44=>a41+x45
// Copy done. Separate the copies.
e42e46=>e48+e48
x48y42=>x48y49
x49x49=>x48+x48
// Strike the import lock.
x48 + a28 => x48 + a50
a50a10 => a50a51
a51a2 =>a52a2
a52a10 =>a53+a54
a52a12=>a52a10
a54a17=>a0+a0
a54a15=>a0+a0
a53a50=>a2a55
a55a51=>a2a55
a55a10=>a2a56
a56a15=>a57+a0
// Form the septum. Make sure the two strings are
// attached to opposite sides, so they will end up
// in different cells.
a57+f48=>a58f37
a58a2=>a60a61
a61+e37=>a62e59
a62a60=>a63a64
a64a2=>a64a57
// This step is slow.
a63+a64=>a65a66
a65+a63=>a100a65
a100a66=>a101+a66
a101a64=>a65a67
// Close the loop.
a67+a2=>a38a68
a68a2=>a68a69
a66+a69=>a38a68
// Separate the cells.
a68a68=>a35+a35
a35a38=>a2a36
a36a36=>a35+a35
a35a65=>a2a36
// Re-attach the e end
x37y48=>x37y37
f37+e59=>f37e59
f37a2=>f38a2
f38e59=>f38+e59
```

Table 2: The reaction rules for cell self-reproduction created by Ralph Hartley.

5 Conclusions and future work

The Organic Builder is open source under the GNU public license. The source code and the applet itself are available at:

`http://www.sq3.org.uk/Evolution/Squirm3/OrganicBuilder/`

A mailing list for related discussions is maintained at:

`http://groups.google.com/group/OrganicBuilder/`

Further challenge levels are sought, both difficult ones to push the limit of what is possible, and easier ones to help with the learning process. Most of the feedback has said that the earlier levels are actually quite difficult because it takes a while to understand what the system does, so any improvement on this count would improve the accessibility of the system. Levels submitted to the mailing list will be considered for inclusion in later versions of the Organic Builder.

One desirable extension of the applet would be to enlarge the simulated world and include periodic partial replenishment, in order to be able to put replicators into competition with each other. A further extension would be to introduce mutation operators, to allow the replicators to evolve in the face of this competition. Creating artificial life-forms that evolve is a central goal of ALife research, and requires much experimentation with systems of sufficient dynamic richness. If questions of evolvability or the requirements for complexity growth can be posed as well-defined challenges, these can be put to the public - and the lesson from this public experiment is: expect solutions shortly afterwards.

References

[1] P. Dittrich, J. Ziegler, and W. Banzhaf. Artificial chemistries - a review. *Artificial Life*, 7(3):225–275, 2001.

[2] T.J. Hutton. Evolvable self-replicating molecules in an artificial chemistry. *Artificial Life*, 8(4):341–356, 2002.

[3] T.J. Hutton. Simulating evolution's first steps. In W. Banzhaf, T. Christaller, P. Dittrich, J.T. Kim, and J. Ziegler, editors, *Proc. Seventh European Conference on Artificial Life*, pages 51–58, Dortmund, Germany, 2003.

[4] T.J. Hutton. A functional self-reproducing cell in a two-dimensional artificial chemistry. In J. Pollack, M. Bedau, P. Husbands, T. Ikegami, and R.A. Watson, editors, *Proc. Artificial Life IX*, pages 444–449, 2004.

[5] T.J. Hutton. Evolvable self-reproducing cells in a two-dimensional artificial chemistry. *Artificial Life*, 13(1):11–30, 2007.

[6] B. Mayer and S. Rasmussen. Dynamics and simulation of micellar self-reproduction. *International Journal of Modern Physics C*, 11(4):809–826, 2000.

[7] K. Morita and K. Imai. Self-reproduction in a reversible cellular space. *Theoretical Computer Science*, 168:337–366, 1996.

[8] T. Oberholzer, R. Wick, P.L. Luisi, and C.K. Biebricher. Enzymatic RNA replication in self-reproducing vesicles: an approach to a minimal cell. *Biochemical and Biophysical Research Communications*, 207(1):250–257, 1995.

[9] N. Ono and T. Ikegami. Self-maintenance and self-reproduction in an abstract cell model. *Journal of Theoretical Biology*, 206:243–253, 2000.

[10] S. Rasmussen, L. Chen, M. Nilsson, and S. Abe. Bridging nonliving and living matter. *Artificial Life*, 9(3):269–216, 2003.

[11] H. Sayama. Self-replicating worms that increase structural complexity through gene transmission. In M.A. Bedau, J.S. McCaskill, N.H. Packard, and S. Rasmussen, editors, *Artificial Life VII: Proceedings of the Seventh International Conference on Artificial Life*, pages 21–30, Portland, Oregon, 2000. MIT Press.