Computational Autopoiesis: The Original Algorithm

Barry McMullin
mcmullin@eeng.dcu.ie

Santa Fe Institute
Working Paper Number: 97-01-001

Dublin City University
School of Electronic Engineering
Technical Report Number: bmcm9701

Abstract

This report presents a detailed review and re-presentation of the algorithm for (computational) realisation of *autopoiesis*, originally presented by Varela, Maturana & Uribe (1974). The review is from the perspective of one seeking to *re-implement* this algorithm. It arises from an on-going project to develop such a re-implementation using the Swarm simulation system.[1] The motivation for such re-implementation is firstly to critically re-examine the phenomenology generated by this model chemistry, and secondly to use it as a basis for exploring more complex systems. The algorithm is first reviewed and annotated for internal consistency and clarity; it is then separately reviewed for consistency with the experimental results which originally accompanied it.

A number of significant discrepancies are discussed. By kind permission of Francisco Varela, this report also includes, as an Appendix, some previously unpublished documentation and source code (in `FORTRAN IV`) associated with the original model. By critical consideration of the experimental results in conjunction with this code, an important—perhaps crucial—interaction, not included in any previous description of the model, has been re-discovered. This interaction is presented and discussed briefly.

Keywords: Autopoiesis, Artificial Life, Artificial Chemistry, Origin of Life.

[1] `http://www.santafe.edu/projects/swarm`

Contents

1 Introduction

Autopoiesis is a term coined by the biologists Humberto Maturana and Francisco Varela to denote a form of system organisation where the system as a whole produces and replaces its own components, and discriminates itself from its surrounding environment, on an on-going basis (Maturana & Varela 1973). The concept was inspired by the abstract organisation of relatively simple biological organisms, such as bacteria. It is intended to capture the essential organisational distinction between such an organism and a mere un-organised aggregation of the same molecular components.

Note that an autopoietic system is, by definition, collectively autocatalytic; but it differs from, or goes beyond, simply being collectively autocatalytic in that it must establish and maintain a *boundary* which contains and separates the reactions comprising the autopoietic system from the rest of its environment.

In an effort to present this concept in the most clear and concrete manner possible, Varela et al. developed a "minimal" abstract or model chemistry that would be capable of supporting autopoietic organisation. Furthermore, they used this as a basis for developing a computer simulation (realisation?) of this model chemistry, in which autopoietic organisations would be expected to spontaneously emerge and persist (Varela et al. 1974). The computer implementation thus served to test the claim that this particular model chemistry is indeed sufficient to support autopoietic organisation. More generally, such computerised models are open to detailed observation and experiment in ways that are currently impractical for real biological organisms.

Varela et al. (1974) presented a detailed algorithm for the computer implementation of their model chemistry. The current report presents a critical re-presentation and review of that algorithm, from the point of view of internal consistency and completeness, and also external consistency with the experimental runs illustrated in the original paper. A number of problems regarding the detailed interpretation of the algorithm will be identified. As far as possible, I shall also suggest "reasonable" resolutions for such problems, which may be helpful to anyone intending to re-implement the model.

I am greatly indebted to Francisco Varela for making available previously unpublished documentation and a source code listing associated with the original model, and for giving his permission to include these as an Appendix to this report. These materials are somewhat fragmentary, and it is not possible to be sure exactly how they relate to the version of the model actually presented in (Varela et al. 1974). Nonetheless, they provide a very important source of additional information, and allow some of the most difficult problems of interpretation of the original published description to be resolved. This general description of the model referred to it as `PROTOBIO`. The source code listing apparently comes from one single specific version or run of the model, labeled `EXP29.FOR`.

Note that this report will not attempt any detailed review of the actual phenomenology to be expected from this class of model—that is deferred for a subsequent companion report.

Fragments from the original paper (Varela et al. 1974) will be quoted verbatim here, where appropriate. However, to fully follow the detailed discussion in the current report it will generally be helpful to have a copy of that original paper to hand, for reference purposes.

2 Overview of the Model Chemistry

The model consists of a two dimensional space, with a discrete square lattice; each lattice position is either empty (referred to as a "hole") or contains one "particle"[2] of one of the defined chemical species or "elements".[3] The size of the space is not explicitly restricted or stipulated, but the illustrated example implementation in (Varela et al. 1974) is sized at 10×10. On the other hand, `EXP29.FOR` implemented a 30×30 space (albeit, this is largely parameterised, and would not be too difficult to change). Thus, it may be that the diagrams of (Varela et al. 1974) show only a part of a larger, simulated, space.

Behaviours at the edges of the space are not specified in detail, but it seems that the edges are to be regarded as hard limits or boundaries to the space—in the sense that particles cannot move through or across the edges. This would be consistent with the behaviour implemented in `EXP29.FOR`.

The particles engage in dynamic interactions based (roughly) on discrete timesteps. On each timestep, particles may move and/or engage in a

[2] Varela et al. (1974) uses the term "component" rather than "particle"; I prefer the latter term, because at any given time these entities may *or may not* be "components" of a higher level (autopoietic?) structure. Calling them "components" seems to me to obscure or prejudge this issue of the formation of higher level structures.

[3] Formally, "holes" could be equally regarded as another "element"; in which case we would say that every lattice position is always occupied by exactly one particle.

reaction.[4]

There are three distinct chemical species:

Substrate:	S
Catalyst:	K
Link:	L

These participate in three distinct reactions:

1. Composition:

$$\mathtt{K} + 2\mathtt{S} \rightarrow \mathtt{K} + \mathtt{L}$$

   In essence, this is a reaction in which two particles of S combine to produce one particle of L. However, the reaction is specified to occur only with the mediation of catalyst, K. The reaction will only occur when two particles of substrate are directly adjacent both to each other and a particle of catalyst. The catalyst particle is unaffected by the reaction (hence its name).

2. Disintegration:

$$\mathtt{L} \rightarrow 2\mathtt{S}$$

   This is a reverse reaction to composition. It occurs spontaneously, with a fixed probability per link, per timestep. It is independent of whether the link is bonded or not (see reaction 3).

3. Concatenation (Bonding): L particles can bond or polymerise into indefinitely long chains.[5] Each L can form at most two bonds. This is a spontaneous reaction, relying only on physical adjacency of two links each of which has either zero or one bond already. There is no reverse, "unbonding", reaction per se. Such chains will be broken down again only if or when constituent links spontaneously disintegrate back to substrate (when any associated bonds will also decay).

S, K and *free* (unbonded) L particles move essentially in random two-dimensional walks—except to the extent that other particles get in the way. This yields something approximating a well-stirred mixture. However, L particles which are bonded (whether single or doubly) do not move at all.[6] It is specified that S particles can permeate through chains of bonded L particles; but L and K particles cannot.

This model "chemistry" is, of course, highly abstract and schematised. Specifically, space and time are both discrete; motion is limited to random walks; there are no substantive analogs of newtonian mechanics, such as mass, force, energy etc.

3 Overview of Anticipated Phenomenology

The current report is explicitly *not* concerned with the phenomenology which this model chemistry may give rise to. However, in order to put the model properly in context, I will briefly summarise the significant phenomenon which might be anticipated.

First note that the given chemistry might plausibly allow the formation (spontaneously or otherwise) of "cellular" structures, consisting of one or more catalyst particles contained within a closed chain of L particles. The boundary or membrane of this structure would be permeable to S; thus, such particles could continue to permeate into the structure. Under the action of the K particles, these would then react to form L particles which would *also* be trapped within the membrane. Thus, a high concentration of L particles could be built up. On an on-going basis, the membrane will rupture, as constituent L particles disintegrate. However, because of the high concentration of L particles already produced within the membrane, it may be expected that one or more of these will diffuse to the rupture site, and allow a repair (possibly in a different conformation), before the K particle can escape. In this way, such a structure would continuously produce or renew itself, while also discriminating itself from its environment: it would be a (minimal) example of *autopoietic organisation*.

Incidentally, it is not apparent from this why the composition reaction was specified to require *two* S particles to generate a single L particle. Granted, if the space is initially filled with S particles plus a small number of K particles, it may be useful to generate some holes as a by product of composition, so as to make motion of particles generally easier; but it would seem equally easy to specify that the initial state of the space already contains a "reasonable" concentration of holes, and simplify the composition reaction to generate one L particle from one S particle. It certainly seems to me that there is nothing in the phenomenology outlined above that

[4] Again, formally, particle movement might be regarded as a special degenerate kind of "reaction", in which *all* reactants are conserved.

[5] It might have been preferable to term links "monomers".

[6] This is a particularly counter-intuitive, or unnatural aspect of this model; I speculate that this limitation was stipulated primarily in order to simplify the computer realisation of the model.

would rely on this 2:1 ratio in the composition reaction.

4 Internal Algorithm Review

This section reviews and annotates the algorithm of (Varela et al. 1974) for internal consistency, completeness, and clarity. In keeping with the original presentation, the discussion is divided into an initial, qualitative, overview, under the heading of "conventions", followed by the more formal algorithm proper.

4.1 *Conventions*

This section of (Varela et al. 1974) first defines alphanumeric codes for the distinct chemical species or "elements" in the model. These are the same codes already introduced above, with the addition of a separate code for *bonded* links (`BL`); However, notwithstanding this, the term "link", denoted `L`, is still usually explicitly qualified as either "free", "singly bonded" or "doubly bonded". In fact, the code `BL` is used only twice by Varela et al. in the subsequent discussion and definition of the algorithm. Since bonded links are otherwise generally regarded as simply link particles with bonds (rather than another distinct kind of element) I will generally dispense with the `BL` code.

Next, the overall structure of the algorithm is described as consisting of two separate phases, the first concerned with movement, the second with the various reactions. A final comment here is that "...[t]he rules by which `L` components bond to form a boundary complete the algorithm"; this is a somewhat unfortunate phrasing because it prejudges the issue of whether bonding will actually lead to *closed* chains ("boundaries").

The discrete space, with a rectangular co-ordinate system, or square lattice, is described next. It is specified that the initial state of the space should contain one or more `K` particles, but otherwise be completely filled by `S`. It is unclear to me why this restriction is imposed; in any case, one would expect any comprehensive computer implementation to allow arbitrary initialisation of the particles in the space (and, indeed, subsequent intervention and manipulation of it).

Both movement and reactions depend, in part, on the relative spatial positions of the interacting particles; so a convention for labeling relative ("neighborhood") positions in the space is introduced next. Figure 1 is a redrawn version of this scheme (drawn in terms of lattice *cells* rather than lattice *points* as in the original paper). Notice, in particular, that the neighborhood extends *two* positions away from the central cell, but only in the orthogonal (not diagonal) directions. This is an essential device to support `S` particles permeating through chains of `L`.

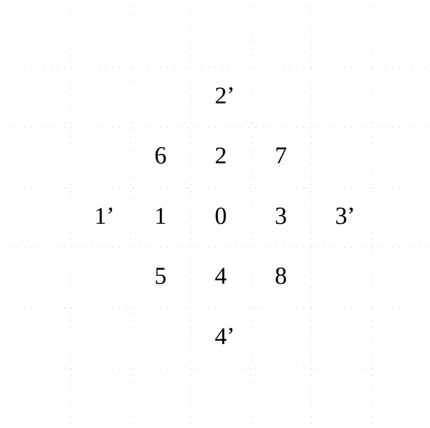

Figure 1: Numbering of Neighbor Positions

The only comment about behaviour at the edges of the space occurs parenthetically in the discussion of this neighborhood diagram:

> ...of course, near the array boundaries, not all of the neighbor locations identified in the figure will actually be found...

I interpret this to mean, as earlier indicated, that the edges are to be regarded as hard limits to the space—no interactions or motions can operate beyond these edges. This is in contrast to, say, making the space effectively toroidal, so that cells on the East edge are considered adjacent to the West edge, and similarly for the North and South edges. I suggest that the latter arrangement might actually be conceptually preferable, because the edges otherwise introduce an asymmetry into the space, which may give rise to artifact phenomena.

There follows a qualitative discussion of motion in the model. This is expressed in terms of the particles being "ranked" by "mass", where "mass" here denotes which kinds of particles may displace which others. This is, at best, a relatively weak analog of newtonian mass, and there are no analogs of force, momentum, energy etc. All motion in the space is fundamentally a form of simple diffusion or random walk.

Roughly speaking then, the ranking by increasing mass is in the order `S`, (free) `L`, `K`; so `K` particles can displace, or push away, free `L` and `S` particles, free `L` particles can displace `S` particles, and `S` particles can only move into holes. Bonded `L` particles cannot move at all. `S` may move through a single thickness chain of bonded `L` particles, but free `L` and `K` particles cannot.

The algorithm proper fleshes out these motion rules more precisely. However, I should note here that this idea of ranking by "mass", with the possibility of "displacement", does complicate the algorithm very substantially and yet seems to me to contribute little if anything to the utility of the model. The rules on permeability of chains of `L` particles are clearly crucial to the possibility of autopoietic organisation being realised–but that is a completely separate issue from whether certain particles are ranked by "mass" or can "displace" other particles. If the idea was to implement varying degrees of *mobility* for the different kinds of particles, I suggest this could have been more simply achieved by qualifying the motions by a probability (of "attempted" motion), per particle, per timestep. In any case, I conjecture that the relatively complicated model of motion actually proposed by Varela et al. does not significantly affect the capacity for the model to support autopoietic phenomena.

Next comes a qualitative discussion of the production reaction, specifying that the two `S` particles must be immediately adjacent to each other and to a `K` particle. Thus, taking the `K` to be in the centre of a neighborhood (position 0 of figure 1), the `S` particles might be in relative positions 2 and 7, or 5 and 4, but *not* 1 and 3, etc. It is explicitly stipulated here that the reaction rate is limited to, at most, production of one `L` per `K` per timestep. If there are multiple pairs of `S` particles which could react under the action of a given `K` in one timestep, then one pair is to be selected at "random". Since the production reaction consumes two `S` particles to produce one `L` particle, it also results, in effect, in production of an additional hole in the space—into which particles may subsequently diffuse.

The final paragraph of this section is concerned with the disintegration reaction and will be quoted in full:

> The disintegration of L's is applied as a uniform probability of disintegration per timestep for each L whether bonded or free, which results in a proportionality between failure rate and size of a chain structure. The sharply limited rate of "repair", which depends upon random motion of S's through the membrane, random production of new L's and random motion to the repair site, makes the disintegration a very powerful controller of the maximum size for a viable boundary structure. A disintegration probability of less than about .01 per timestep is required in order to achieve any viable structure at all (these must contain roughly ten L units at least to form a closed structure with any space inside).

This introduces the only explicitly specified parameter of the model (subsequently labeled P_d), being the probability of disintegration per `L` particle (bonded or otherwise) per timestep. The implication is that this is the only behaviour or interaction in the model which will be characterised by such a rate parameter; one must presume, therefore, that the other dynamics are intended to proceed at the "maximum" rate (consistent with any explicitly stated constraints or enabling conditions). Thus, all particles *will* move whenever they can; `S` particles *will* react (under the action of `K`) whenever they can; `L` particles *will* form additional bonds whenever they can.

It is clear that the model could be easily generalised by *introducing* new rate parameters (effectively, probabilities of the action, per site, per unit time), analogous to the disintegration rate parameter. Setting these parameters to 1 would then recover precisely the model explicitly specified by Varela et al. (1974). The "mobility" probabilities mentioned earlier are specific examples of this. I suggest, therefore, that a computer implementation of the model should support such additional parameters, thus allowing a wider range of models, and corresponding phenomenology, to be explored. These potential additional parameters will be elaborated below in the detailed discussion of the algorithm proper.

The final point to be made here is that while Varela et al. stipulate that P_d should be less than 0.01 (which would yield a composite probability of disintegration for a 10 particle `L` chain of just under 0.1, or an expected lifetime, for the chain, of about 10 timesteps), they do not say precisely how *much* less. Clearly, the phenomenology of the model will alter as this parameter is made smaller; thus, it would have been useful to specify what precise value(s) were used in the particular experiments reported in the paper.

4.2 *Algorithm*

In this section I reproduce the original algorithm from (Varela et al. 1974), and intersperse commentary and discussion. The original algorithm will be presented verbatim in *italic* font, with the commentary in normal, roman, font. But first, some comments on the overall layout and presentation of the algorithm.

The original presentation of the algorithm is as an hierarchical list. At the topmost level there are 7 sections, numbered 1 to 7. Within these, there are subsections, numbered, for example, 1.1, 1.2, 1.3 etc. Some of these subsections have further subsidiary subsubsections, numbered, for example, 4.21, 4.22 etc.

A simplistic interpretation of this structure is that this complete set of steps should be sequentially executed to implement one timestep of the model. In practise, this is only very roughly the case. It appears in fact that one timestep consists of the sequential execution only of top-level sections 1 through 5; the top level sections labeled 6 (Bonding) and 7 (Rebond) are to be regarded only as *sub-procedures* which may be invoked at various points in sections 1 through 5. Indeed, there are other, lower-level, fragments of the algorithm which also seem to be re-used in this manner.

Further, while the subsections and subsubsections might be assumed to be sequential refinements of the containing section or subsection, in practise their status is slightly more complex than this. In particular, they sometimes function as restrictions or adjustments of what has been specified at the higher level, to suit certain specified circumstances. My interpretation in this sort of case is that the higher order action or rule is to be carried out *unless* overridden by the more detailed subsidiary section.

With that preamble, I now reproduce the algorithm itself, with annotations and discussion. A small number of purely typographical errors in the original have been corrected without comment. The original used only one level of indentation; I have introduced additional indentation for every sublevel.

1. *Motion, first step*

   This top-level step essentially scans each hole in the space, and permits each one to swap position with an immediately adjacent `S`, or with an `S` which can be reached through a single thickness of bonded `L`. As such, one might think that this step would be formally equivalent to scanning the `S` particles, and potentially moving them into neighboring holes. Furthermore, the latter would seem like a more consistent presentation, since the subsequent two top-level steps will be concerned with motion of free `L` and `K` particles respectively. However, as we shall see, it turns out that there are built in asymmetries between holes and `S` particles in this step, which prevent such an equivalent reformulation.

   1.1. *Form a list of the coordinates of all holes* h_i.

   The notational convention here seems to be that h denotes the list of holes, and h_i will then denote a typical member of the list.

   Presumably, an implementation may, for efficiency reasons, opt to *maintain* this list, rather than regenerate it afresh on each iteration.

   It is not stipulated what the ordering of this list should be. This is a significant point, because subsequent substeps will process the members of the list in order; thus, regularities in the ordering of the list may possibly give rise to subtle unexpected (and unwanted) regularities in the movements of holes. It would therefore be preferable to stipulate that, whatever the ordering of the list as such, the subsequent processing steps should select elements in a random order, without repetition, until the list is exhausted (or, equivalently, that the list should be randomly permuted, on each timestep, before processing).

   1.2. *For each* h_i*, make a random selection,* n_i*, in the range 1 through 4, specifying a neighboring location.*

   As previously outlined, a generalisation of the model would introduce a prior substep here, making motion of any given hole (or `S` particle?) occur only with some probability, say P_{hm}. Making P_{hm} equal to 1 would then recover the exact algorithm specified by Varela et al.

   I presume that in this step, and in all similar subsequent cases, the intention is to use a *uniform* probability function over the specified event set.

   The numbers 1 through 4 refer here to the neighborhood diagram (figure 1). Thus the selection is of an immediately neighboring position in one of the four orthogonal directions. The *diagonally* neighboring positions, 5 through 8, are *not* to be allowed or investigated. The reason for this is presumably that, as we shall see below, a subsequent refinement of this step will potentially investigate the positions one unit further away (the positions labeled 1' through 4'); such a "two unit" distant position, and especially the position "in between", will be unique in any of the orthogonal directions, but would be ambiguous for the diagonal directions. For simplicity of expressing the algorithm then, and without any serious loss of generality in the expected phenomenology, it is probably best just to restrict attention to the orthogonal directions.

1.3. For each h_i in turn, where possible, move occupant of selected neighboring location in h_i.

The way the algorithm has been organised here suggests that the intention is to first iterate over the entire list, selecting a candidate movement direction (step 1.2) and then, separately, iterate over the list again to actually make each move (step 1.3). However, it seems to me that this will be formally equivalent to iterating over the list just once, in each case picking a direction and then making the move before moving on to consider the next element. This would also be somewhat more efficient, so an implementation may presumably opt to prefer this approach.

The key phrase in this step seems to be "where possible". Two subsidiary steps or rules are now presented (1.31 and 1.32); since both of these stipulate conditions in which the movement should *not* be carried out (is not "possible"), the intended interpretation seems to be that the motion is allowed *except* where 1.31 or 1.32 specifically disallow it.[7]

This is significant because (as we shall see) it implies that the movement *will* be carried out if the selected position contains any of an `S`, free `L`, or `K` particle. For that reason, this whole step is not symmetrical with respect to holes and `S` particles, and actually cannot be equivalently expressed in terms of `S` particles, as earlier suggested. However, since there are separate subsequent top-level steps (2 and 3) concerned explicitly with the motion of free `L` and `K` particles, it is doubtful whether the possibility of moving those particles in *this* step adds any significant functionality to the model. Further, by forcing the current top-level step to be expressed in terms of hole rather than `S` particle movement, this stipulation obscures the overall operation of the algorithm and introduces inconsistency with the expression of the other movement steps. Therefore, I suggest that it would have been preferable to explicitly exclude the possibility of free `L` or `K` particle movement in this step.[8]

1.31. If the neighbor is a hole or lies outside the space, take no action.

If the neighbor position holds a hole then exchanging the particles would have no net effect on the overall configuration of the space. However, it is still good to make it explicit that these holes should *not* be swapped, as this *will* have some effect on the subsequent behaviour of the model. This is because, although the overall configuration of the space would be unchanged by this swap, the *other* hole may not yet have been processed through step 1—so moving it now would alter how it *will* be processed. Granted, it is doubtful whether this would have any substantial effect on the qualitative behaviour of the model; but it is surely simpler to simply rule it out, as 1.31 does.

The stipulation that, if the point is "outside the space" no further action should be taken, is useful, as it rules out the other possible behaviour one might consider, of choosing an alternative neighboring position and trying again.

1.32. If the neighbor n_i contains a bonded L, examine the location n_i'. If n_i' contains an S, move this S to h_i.

This captures the idea that an `S` particle can diffuse or permeate through a single layer of bonded link. However, it somewhat begs the question: if an `S` particle can permeate through a *bonded* `L` particle, why could it not permeate through a *free* `L` particle? The reason appears to

[7]That is, it seems that 1.31 and 1.32 are not to be interpreted as a sequential *refinements* of 1.3, but rather as concurrent qualifications or constraints on it.

[8]I conjecture that part of the explanation of this, and some other related peculiarities of the algorithm, may be that the algorithm was actually derived from a functional implementation (and thus inherited artifacts from it), rather than the algorithm having been written first and used as the specification of the implementation. However, this is of only historical interest at most.

be that, if a free L is found in the neighboring position then *it* is to be exchanged with the hole, rather than considering the possibility of an S particle beyond it. However, it has already been noted that incorporating free L particle movement into this part of the algorithm is questionable anyway; if the possibility of free L particle movement at this stage were dropped, then it would seem perfectly consistent (and even simpler) to permit S particles to permeate through a single thickness of both free and bonded L particles.

1.4. Bond any moved L, if possible (Rules, 6). This makes it clear that top-level section 6 is to be regarded as a sub-procedure.

The reference to "any moved L" again reinforces the earlier interpretation of 1.3, that it is indeed intended to allow a hole to exchange positions with a free L, if one is encountered (despite the fact that free L particles will get an opportunity to move in top-level step 2 anyway).

Again, the layout of the algorithm suggests that there is to be a *separate* iteration over the (moved) free L particles, implementing the bond procedure, taking place *after* all the movements of 1.3 have been completed. This seems a little perverse from an implementation point of view, since it requires recording a list of the moved free L particles, to allow this separate iteration. It would seem simpler to just bond any moved free L *when it is moved*, rather than recording it for processing later, in a separate iteration. In the previous case of separate iterations (steps 1.2 and 1.3), I argued that the outcome would be formally equivalent if they were combined into a single iteration—and that therefore an implementation could opt to combine them. However, that argument does *not* carry through to the current case. It is evident that the ability of an L particle to form bonds depends on what other L particles are in its neighborhood. Therefore, the outcome of this bonding iteration will be potentially different depending on which L particle movements have been previously completed. Thus, it is not strictly free to an implementation to combine 1.4 into the same iteration as 1.3.

However, this does raise a more general criticism of the algorithm. As we shall see, there are single top-level steps concerned with realising the composition and disintegration reactions. These essentially iterate over each potential reaction site in the space, and permit one such reaction per timestep. One might have thought that the bonding reaction could (and, for consistency and simplicity, *should*) be implemented in essentially the same way. However, the bonding reaction is actually handled in a quite different manner. Instead of a single top-level step for bonding, there are two distinct sub-procedures (6 and 7), both concerned with the bonding reaction, but with quite different behaviours; of these, one is invoked in only one part of the algorithm (7 is invoked only as part of top-level step 5, i.e. following disintegration), and the other is repeatedly invoked in the other top-level steps. This certainly makes expected behaviour of the bonding reaction significantly more obscure, while also making the overall structure of the algorithm a good deal more complex. It is not at all clear what the rationale for this might have been. For example, it is possible that the particular implementation of bonding which was chosen had a significant effect on the qualitative phenomenology (and especially the emergence and stability of autopoietic structures); however, if that was the case, it would have been very helpful for this to be made explicit and elaborated in some detail.

2. Motion, second step

This step is concerned with the movement of free L particles. It is significantly complicated by the idea of the various particle types having different "masses" and the possibility of "displacements". It also incorporates the possibility of bonding reactions, as with top-level step 1 (rather than deferring the bonding reaction to a single top-level step in its own right).

2.1. Form a list of the coordinates of free L's, m_i.

Again, the presumed notation here is that m denotes the list, and m_i a typical element of the list. However, whereas h was presumably mnemonic for the list of holes formed in top-level step 1, there is no apparent mnemonic intention here.

As with 1.1, the implementation may presumably opt to maintain this list rather than regenerating it; and, again, it would probably be preferable to randomly permute it before processing on each timestep, to guard against artifacts arising from any systematic ordering.

2.2. For each m_i, make a random selection, n_i, in the range 1 through 4, specifying a neighboring location.

As with step 1.2, movement is allowed only in the four orthogonal directions.

2.3. Where possible, move the L occupying the location m_i into the specified neighboring location.

We interpret this, as with 1.3, as a general rule which will be qualified by the following subsidiary steps. That is, it is assumed that the move is possible, unless these substeps rule it out. Also, as with 1.2 and 1.3, we assume that 2.2 and 2.3 may be combined into a single iteration if there is some efficiency benefit in doing that.

2.31. If location specified by n_i contains another L, or a K, then take no action.

As with a hole swapping with a hole, a free `L` particle swapping with a free `L` particle would have no net effect on the overall configuration of the space; but again, it could have an effect on the behaviour of the model, because the *other* free `L` might not yet have been processed for movement, and would be subsequently processed differently if it is moved now. So it does seem worthwhile to explicitly exclude this.

The rule does imply two other significant conditions. First that an `L` particle will not swap with a `K` particle. This is presumably intended to partially reflect the idea that `K` is of greater "mass" than `L`; but it is somewhat inconsistent, because `K` particles *can* swap with the (zero "mass"?) holes in step 1. The second condition implied here is that a free `L` particle will not swap positions with a *bonded* `L` particle. This reflects the general stipulation that bonded `L` particles are absolutely immobile.

2.32. If location specified by n_i contains an S, the S will be displaced.

This step attempts to capture the idea that because an `L` particle is of greater "mass" than an `S` particle, then it can "displace" the `S` particle to some other position, thus making a hole that the `L` can then move into. However, the details are necessarily somewhat complex, because there may be some difficulty in actually displacing the `S`. This is detailed in the following three steps, subsidiary to 2.32, which attempt to explain exactly how this idea of "displacement" is to be realised.

2.321. If there is a hole adjacent to the S, it will move into it. If more than one such hole, select randomly.

2.322. If the S can be moved into a hole by passing through bonded links, as in step 1, then it will do so.

Presumably, 2.322 is intended as a fallback, only to be applied if the `S` particle cannot be moved in accord with 2.321.

The phrase "as in step 1" deserves some elaboration. The intention is evidently that the `S` particle is allowed to exchange with a hole which is behind a single thickness of bonded `L` particles. However, whereas in step 1, a single direction (out of four) is first chosen, and the `S` particle is either moved in that direction or not at all, the intention here seems to be that all directions (three only) should be assessed for this possibility;[9] and if more than one is possible, select one at random. I take this interpretation, even though it is not fully stated, for consistency with the immediately prior step, 2.321, and because the next step (2.323) explicitly refers to the case that "the S cannot be moved into a hole", implying that *every* possible way of "displacing" the `S` should be checked first.

[9] Only three directions arise here, since one direction is already accounted for by the free `L` particle we are attempting to move at the higher level of the algorithm.

Thus, although 2.322 does involve some similarity to step 1, there are significant differences also (especially that fact that step 1 was centered on a hole not an `S` particle, the fact that all available directions are tried rather than plumping for one only, and the fact that only three directions, at most, might be possible). This is of interest in that it effectively rules out any idea of isolating this functionality into a single subprocedure, shared both by step 1 and step 2.322.

2.323. If the S cannot be moved into a hole, it will exchange locations with the moving L.

This is the final fallback case: in essence, if the `S` particle cannot actually be "displaced", then the free `L` particle will simply swap positions with it.

2.33. If the location specified by n_i is a hole, then L simply moves into it.

We can note again that, since an `L` particle can swap position with a hole at this step anyway, it is questionable what purpose is served by allowing equivalent swaps in step 1 also.

Note that steps 2.31 through 2.33 have actually enumerated *all* possible contents of the neighbor position, and specified a corresponding behaviour. These rules were ordered (roughly) as `K`, `L`, `S`, and finally hole. It might have been somewhat more clear to order them with hole before `S`, since this would get the simpler cases out of the way before dealing with the cumbersome case of `S`. In any case, from an implementation point of view, it is clear that these cases are mutually exclusive, and can thus be considered in any desired order.

2.4. Bond each moved L, if possible.

As with 1.4, the bonding reaction has been inserted as a subprocedure invocation here. Unfortunately, there are two top-level steps concerned with bonding—6 and 7—so it is not immediately clear which should be invoked in this case. However, as we shall see, step 6 expects an argument of the position of a free `L` particle, whereas step 7 apparently expects the position where a free `L` has just disintegrated. Therefore, it may be assumed that the intention is to invoke top-level step 6 as the subprocedure here.

Similar criticisms of the overall approach to the bonding reaction as detailed under step 1.4 apply again here. One additional point is worth making explicit now: *because* the possibility of a bonding reaction is being reevaluated here, certain free `L` particles may be assessed for bonding more than once in any given timestep (because they happen to be moved in, say, both steps 1 and 2), whereas others will not. This seems a rather arbitrary variability in reaction rate, which would be better avoided.

3. Motion, third step

This is primarily concerned with movement of `K` particles. As with top-level step 2, it is significantly complicated by the idea of the various particle types having different "masses" and the possibility of "displacements"; however, the situation is even more complex this time, as the `K` particle may displace a free `L` particle which must (first) displace a `S` particle. This step also allows for movement and potential bonding of free `L` particles, even though possibilities for both these actions have already been allowed in both the previous top-level steps.

3.1. Form a list of the coordinates of all K's, c_i.

Again, the presumed notation here is that c denotes the list, and c_i a typical element of the list. c is presumably mnemonic for "catalyst"—though for consistency, it would have been preferable either to use k, or use `C` as the code for the catalyst particle.

As with 1.1, and 2.1, the implementation may presumably opt to maintain this list rather than regenerating it; and, again, it would probably be preferable to randomly permute it before processing on each timestep, to guard against artifacts arising from any systematic ordering.

3.2. For each c_i, make a random selection n_i, in the range 1 through 4, specifying a neighboring location.

As with step 1.2 and 2.2, movement is allowed only in the four orthogonal directions.

3.3. *Where possible, move the K into the selected neighboring location.*

We interpret this, as with 1.3 and 2.3, as a general rule which will be qualified by the following subsidiary steps. That is, it is assumed that the move is "possible", unless these substeps rule it out. Also, as with 1.2 and 1.3, and 2.2 and 2.3, we assume that 3.2 and 3.3 may be combined into a single iteration if there is some efficiency benefit in doing that.

3.31. *If the location specified by n_i contains a BL or another K, take no action.*

As with a hole swapping with a hole (step 1.31), or a free `L` particle swapping with a free `L` particle (step 2.31), a `K` particle swapping with a `K` particle would have no net effect on the overall configuration of the space; but again, as with those previous two cases, it could have an effect on the behaviour of the model, because the *other* `K` might not yet have been processed for movement, and would be subsequently processed differently if it is moved now.

This rule also incorporates the condition that a `K` particle will not swap positions with a *bonded* `L` particle, again reflecting the general stipulation that bonded `L` particles are absolutely immobile.

3.32. *If the location specified by n_i contains a free L, which may be displaced according to the rules of 2.3, then the L will be moved, and the K moved into its place. (Bond the moved L, if possible).*

This step seems straightforward, essentially invoking step 2.3 as a subprocedure. However, there is a difficulty, or *ambiguity* with this. Step 2.3 was designed to assess the possibility of moving a free `L` particle in one, prespecified or chosen, direction only. It is not at all clear that the same applies to the current case. It is possible that the intention is to try to displace the free `L` particle in the same direction as the attempted move of the `K` particle (n_i in the context of 3.32). However, that would be somewhat inconsistent with the policy specified in 2.321 where an `S` particle, being displaced by a free `L` particle, may move in *any* available direction. Furthermore, as we shall see, step 3.34 will specify a "fallback" action for the case that the free `L` particle is "not movable by rules 2.3", which suggests to me that the current step (3.32) is intended to attempt all possible directions of movement for the free `L` particle. In any case, whatever the "correct" interpretation, the rule, as stated, is certainly ambiguous.

This rule also parenthetically mentions that, if the free `L` particle is moved, it should be bonded. As in step 2.4, the bonding sub-procedure has not been stipulated explicitly, but it seems clear that top-level step 6 is intended.

Note that the invocation of bonding at just this point in the algorithm is actually inconsistent with the way it was handled at steps 1.4 and 2.4. In the previous cases, *all* free `L` particle movements were completed before *any* were processed for bonding. Whereas, in the current case, the possibility of bonding is considered as each free `L` particle is moved, rather than as a separate iteration. This change in bonding procedure or policy is repeated in step 3.34; but then, in top-level step 4, we shall see a reversion to the same policy as steps 1 and 2, where bonding is deferred to the end of the (top-level) step. There is no apparent rationale for this variation in policy. It further re-inforces the arguments made earlier that the bonding reaction should be processed just once, as a top-level step of its own.

3.33. *If the location specified by n_i contains an S, then move the S by the rules of 2.32.*

Again, this looks like 2.32 should be simply invoked, in effect, as a subprocedure; but again, there is a difficulty with this. Step 2.323—being part of 2.32—covers the case where it turns out that the `S` particle *cannot* be displaced, and stipulates that the `S` particle will then "exchange locations with the moving L". The problem is that, in the context of 3.33, there is no "moving L", but rather a "moving K". Of course, the intended meaning here is clear enough; but to express it cleanly, step 2.3 ought to be explicitly separated out *as* a subprocedure, and given an argument, being the position of an *indefinite* particle (actually either a free `L` or a `K`—but the subprocedure will be oblivious to this) which is trying to displace it.

3.34 If the location specified by n_i contains a free L, not movable by rules 2.3, exchange the positions of the K and the L. (Bond L if possible).

This is analogous to step 2.323.

However: since it is essentially a fallback, or alternative, to step 3.32, it would preferable if it came immediately after 3.32; or, for consistency with the way 2.32 was expressed, 3.32 and 3.34 could have been made subsidiary cases of a single rule. In any case, as the various substeps of 3.32 are mutually exclusive, it will be open to an implementation to re-order the consideration of them if convenient.

3.35. If the location specified by n_i is a hole, the K moves into it.

A similar comment applies as at step 2.33. Given that this step allows a `K` particle to exchange with a hole, it seems unnecessary to allow for that at top-level step 1 also.

In closing top-level step 3, we may finally note that, unlike top-level steps 1 and 2, there is no final step here concerned with bonding any moved link—because the bonding was integrated into substeps 3.32 and 3.34, as already discussed.

4. Production

This step implements the reaction earlier termed "composition" but now (apparently) renamed "production". This is specified to occur only under the influence of `K` particles; therefore it is efficient to focus only on those positions in the space where there are such particles, and iterate over each in turn. Each particle can trigger at most one production reaction per timestep. To do so, there must be two `S` particles immediately adjacent both to the `K` particle and to each other (in this case, *diagonal* neighbors are counted as immediately adjacent). If there are multiple adjacent pairs of `S` particles, adjacent to a single `K` particle, one is chosen at random, as the reacting pair. The reaction proper is then implemented by deleting the two `S` particles, and replacing them by a single `L` particle. This is located randomly at either of the two positions previously occupied by the `S` particles (the other position becoming a hole). Finally, the possibility of bonding is processed for the newly created `L` particle (and, again, I would argue that it would be better for this to be handled as a single, separate, top-level step, rather than conflated with the production reaction).

As the algorithm is currently specified, if it is possible to carry out a production reaction in the neighborhood of a given `K` particle, then the reaction *will* happen. As previously discussed, this could instead be made subject to a reaction rate parameter (probability of reaction), to add an additional degree of flexibility to the range of models that could be investigated.

4.1. For each catalyst c_i, form a list of the neighboring positions n_{ij}, which are occupied by S's.

Strangely, this top-level step does *not* start with an explicit instruction to "form a list of the coordinates of all K's, c_i", even though that is clearly implied, and such an instruction was made explicit in each of the previous top-level steps.

In any case, the same comments should be noted as in each of the previous such cases: it may be convenient, for implementation efficiency, to maintain this list rather than regenerate it; and, to avoid any artifacts arising from the order of this list, it should be randomly permuted before processing the production reaction.

Steps 4.1 through 4.3 are now interpreted as, in effect, substeps to be repeated for each `K` particle in the space.

Step 4.1 is simply concerned with identifying or listing the positions of `S` particles adjacent to each `K` particle. As step 4.1 is phrased, it reads as an instruction to form these lists (n_{ij}) for *all* `K` particles *before* proceeding to carry out the production reactions for any of them. One might immediately assume, as in the previous somewhat similar cases[10], that it is free to the

[10] Steps 1.2 and 1.3, or 2.2 and 2.3, or 3.2 and 3.3.

implementation to choose to combine these two iterations into one. However, the situation in this case is actually more complex, and there is no real choice; despite the phrasing of 4.1, these two iterations surely *must* be combined into one. The reason is that a given `S` particle *may* be (initially) adjacent to more than one `K` particle. If the lists n_{ij} are all generated first, before any of the production reactions are carried out, then "early" production reactions may actually render some of the "later" lists invalid—because an `S` particle that was listed as adjacent to a certain `K` particle has actually, in the interim, reacted to give an `L` particle under the action of a *different* `K` particle.

Of course, it is possible that the algorithm is "correct" as stated—in the sense that the authors actually did intend separate iterations to be used, and had not recognised the potential difficulty this could lead to. Provided the density of catalysts is "low", the probability of encountering this problem will also be low. The actual implementation run illustrated in (Varela et al. 1974) shows only a single `K` particle in the space—and, in that case, of course, the difficulty identified here cannot actually arise. However, it is a real defect in the algorithm, nonetheless; so I recommend that any new implementation should choose to combine the iterations, as outlined. It is apparent that this combination should make no difference to the effect on the space *except* in the case that the reactions at different `K` sites would interact—and for those cases, one *needs* the robust form of the algorithm.

4.11. Delete from the list of n_{ij} all positions for which neither adjacent neighbor position appears in the list (i.e., "1" must be deleted from the list of n_{ij}'s, if neither 5 nor 6 appears, and a "6" must be deleted if neither 1 nor 2 appears).

The numbers here refer, of course, to the neighborhood diagram presented earlier (figure 1).

Essentially what is happening here is that the original list, n_{ij} contained *all* `S` particles immediately adjacent to the given `K` particle (c_i), whereas production is possible only for `S` particles that come in *pairs*—i.e. that are adjacent both to each other and to the `K` particle. Presumably, it is free to an implementation to generate this "reduced" list "directly"—rather than first generating a list of all adjacent `S` particles and then removing some elements from it again. In any case, it is not at all clear why this step is expressed as a *sub*-step of step 4.1. There are no additional such substeps; it would seem that 4.11 could either have been made a separate step at the same level as 4.1, or 4.1 and 4.11 could have been combined into a single step.

4.2. For each c_i with a non-null list of n_{ij}, choose randomly one of the n_{ij}, let its value be p_i, and at the corresponding location, replace the S by a free L.

Again, the way this is phrased, it suggests a separate iteration over the `K` particles, but this is not a reasonable interpretation for the reason already explained.

The essential point of step 4.2 is to say that if there are multiple candidate pairs of `S` particles which may react under a given `K`, then one should be picked "at random". This does not mean exactly "uniformly" at random, because the choice is made in two stages. Instead of explicitly listing all possible *pairs* and then (say) choosing with a uniform distribution over them, the rule says to first pick a single (eligible) `S` particle and then two subsidiary steps (4.21 and 4.22) will deal with the issue of picking the second `S` particle to pair with this one. This can slightly skew the probability distribution among the possible pairs. For example, with four adjacent particles, the probability of choosing either end pair will be 3/8 versus 1/4 for the middle pair. On the other hand, the algorithm, as stated, *does* have the effect of ensuring that which `S` particle gets replaced with a free `L` *is* uniformly chosen from the candidates (whereas, if a pair was first picked with a uniform probability function, and then one position out of the pair chosen uniformly to be replaced by the `L`, this could skew the probability of locating the `L` particle somewhat in favour of the "inner" `S` particle locations).

In any case, it seems highly unlikely that these relatively fine distinctions in probability distribution for the production reaction would have any substantive effect on the qualitative phenomenology of the system.

  4.21. If the list of n_{ij} contains only one which is adjacent to p_i, then remove the corresponding S.

  4.22. If the list of n_{ij} includes both locations adjacent to p_i, randomly select the S to be removed.

4.3. Bond each produced L, if possible.

5. *Disintegration*

The disintegration reaction is superficially very straightforward: each `L` particle (bonded or otherwise) is considered for disintegration with a probability P_d, where P_d is an explicit parameter of the model chemistry. However, as we shall see, there is considerable complexity involved in actually carrying out the disintegration.

5.1. For each L, bonded or unbonded, select a random real number, d, in the range (0,1).

As with top-level step 4, there is presumably an implied prior step here, of forming a list of `L` particles; again, we assume that an implementation may opt to maintain this list instead of generating it anew each timestep; and, to avoid artifacts arising from the ordering, the list should be permuted before proceeding with the disintegration processing.

Again, the phrasing of step 5.1 seems to imply a separate iteration, generating a random number for each `L` particle, before coming back and processing the disintegration reaction. However, unlike the previous such cases, the notation here has changed subtly: the step refers to a single random number d, rather than a list, d_i. That may suggest that, on this occasion, a separate iteration to first generate the random numbers is *not* intended. This is perhaps supported by the fact that the rest of this top-level step consists of only two *sub*-steps of 5.1—suggesting that they form part of the same, single, iteration over the `L` particles. In any case, there seems to be no formal distinction between using a single iteration to both generate the random numbers and process the disintegrations or using two separate iterations—so this choice is presumably free in the implementation.

  5.11. If $d \leq P_d$ (P_d an adjustable parameter of the algorithm), then remove the corresponding L, attempt to rebond (Rules, 7).

  5.12. Otherwise proceed to next L.

The effect of 5.1 combined with 5.11 and 5.12 is to say that each `L` particle should be subject to disintegration with probability P_d. This choice can be achieved in the manner specified (i.e. by indirectly using a sample of a uniform random variable); however, I presume that if an implementation has a source of bernoulli random variables, with programmable probability, that may be substituted here instead.

In any case, the real complexities of this reaction come into play in the case that a given `L` is, indeed, to be subject to disintegration (step 5.11).

There are two problems or difficulties to be considered.

Firstly, if the `L` particle was bonded, what should happen to those bonds? The obvious answer would be that they should simply be destroyed. However, the algorithm specifies something much more complex. Instead of simply destroying existing bonds, the algorithm specifies than an attempt must be made to "re-bond". The details of this are contained in a separate top-level step (7). The implication is that step 7 is to be regarded as a subprocedure. However, given that this is the only place where this subprocedure is invoked, it might have been simpler to include it as an explicit part of top-level step 5 instead.

Much more seriously, the "re-bonding" procedure is actually concerned not with destroying whatever bonds were associated with the disintegrated `L` particle—that is actually left implicit—but with forming *new* bonds among whatever `L` particles remain in the neighborhood of the disintegration site. In effect, this is therefore a special purpose variant of the bonding reaction.

As already discussed at length, there is no clearly articulated reason why the bonding reaction should be handled in such a complex way, conflated into the other reactions, rather than dealt with once per timestep in its own right. The current case is particularly troublesome because disintegration is the way that an autopoietic entity may be disrupted; if the autopoietic phenomenon is robust, one would not expect to have to insert special measures relating specifically to the reactions arising from such disruption. Yet, prima facie, it appears that just such special measures have been built into the algorithm here with this notion of "rebonding". As a further ancillary point: rebonding is actually specified to be invoked here regardless of whether the original `L` particle was previously bonded or not; again, this further serves to obscure the operation of the bonding reaction in general, providing yet another separate opportunity for bonding to happen per timestep. In any case, detailed consideration of rebonding will be deferred to the discussion of step 7 proper below.

The second outstanding difficulty with disintegration is that this reaction involves replacing a single `L` particle with *two* `S` particles. One of these will clearly occupy the position formally occupied by the `L` particle—but where is the other to be positioned? Presumably, it should be located in a hole—but that raises the non-trivial question of which hole? The algorithm gives absolutely no guidance here—it merely states categorically that, if $d \leq P_d$ then the disintegration *will* take place.

For chemical "plausibility" one would expect to choose a "neighboring" hole. Therefore, one might formulate a rule such as: if there is a hole immediately adjacent to the disintegration site, locate the extra `S` particle here; if there is more than one such hole, choose one uniformly at random. But what if there is *no* adjacent hole? In the general case, it may actually be necessary to scan arbitrarily far away to locate a hole.[11] Scanning arbitrarily far away from a reaction site, within a single timestep, clearly violates the basic idea of *local* interaction, which seems to be essential to the idea of a model "chemistry" of any sort, and should not be introduced without some clear rationale. Indeed, if I were developing this algorithm from scratch, I would suggest that the appropriate rule would be that, if there is no immediately adjacent hole, the disintegration should simply not occur. I doubt very much whether this would significantly affect the qualitative phenomenology. Alternatively, and more radically, I noted earlier that there was no clear requirement for the production reaction to require two `S` particles anyway; if the reaction were redefined so that a single `S` could be transformed to a single `L` particle (and vice versa for disintegration) then this whole difficulty would disappear anyway.

Nonetheless, we must record that a strict interpretation of the original algorithm appears to require searching arbitrarily far from the reaction site, until a hole is located. It appears to be entirely free to the implementation to decide all details of this search.[12]

6. *Bonding*

This step must be given the coordinates of a free L.

This is the (main) subprocedure for implementing the bonding reaction. As already noted, it is invoked from each of the top-level steps 1 through 4. In each such case, there is a specific free `L` particle relative to which bonding will be attempted. I will refer to this below as the *target* `L` particle.

There are two imposed constraints on bonding. Firstly, any given `L` particle can have at most two bonds. Secondly, a *second* bond can form at any singly bonded `L` particle only if the angle between it and the prior bond will be at least 90°. This bond angle is illustrated in figure 2 (redrawn from figure 4 of the original paper). Note that this diagram clearly indicates that diagonal bonds *are* allowed in general, subject only to the condition that a bond angle of 45° is not permitted.

[11] Note that, with the specified initial condition of the space—holding only `K` and `S` particles, then, by conservation rules, there would always be guaranteed to be a hole available *somewhere* in the space, given that an `L` is being disintegrated; but if that initial constraint were relaxed, it would be possible that a hole might not be available at all.

[12] Indeed, since the algorithm makes no stipulations at all about how this is to be done, locality *could* be completely ignored—for example by saying that the extra `S` particle should be put in a hole simply chosen uniformly from all holes in the space, without any regard to proximity to the disintegration site.

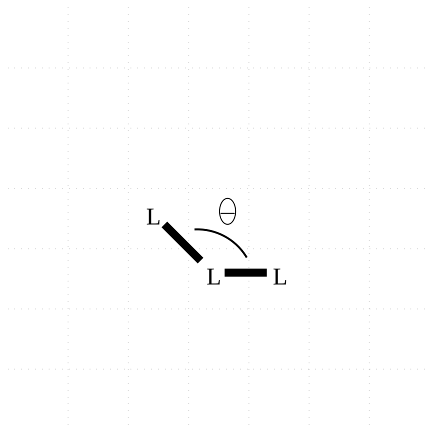

Figure 2: Definition of Bond-Angle θ.

One presumes that the rationale for disallowing bond angles less than 90° (i.e. either 45°, or the degenerate case of 0°) is to encourage the formation of more "open" chains; however, this is not discussed in detail in the paper. It is difficult to judge how significant it may be for the qualitative phenomenology. In any case, one would like any implementation to make it easy to relax this constraint if desired.

Bond formation, where possible, is "forced". That is, there is no "reaction rate parameter", or overriding probability, governing bond formation. Such a parameter could be added; but its effect would be seriously complicated by the fact that any given `L` particle might be considered (directly or indirectly) for bonding several different times within one timestep.

In outline, the strategy here is to first preferentially form bonds between the target `L` particle and neighboring `L` particles that *already* have one bond; if the target `L` particle is then still not fully bonded (i.e. does not yet have two bonds) then further bonding will be attempted with neighboring *free* `L` particles. In any case, the given `L` particle will be fully bonded, if that is possible. Wherever there are multiple possibilities, choices are made randomly (presumably uniformly among the possibilities). However, because of the multitude of possible neighborhood configurations, the algorithm for achieving all this is quite complex.

The flow of control through this procedure is particularly hard to follow. It seems important to note that, in contrast to earlier cases, the *sub*-steps in the part of the algorithm specify additional sequential steps, rather than providing clarification or constraints on the higher level, containing step.

Again, no specific rationale is offered for preferentially bonding to *singly bonded* `L` particles; but I speculate that this was intended to make the repair of a ruptured chain more likely or robust.

6.1. Form a list of the neighboring positions n_i, which contain free L's, and the neighboring positions m_i which contains singly bonded L's.

6.2. Drop from the m_i any which would result in a bond angle less than 90°. (Bond angle is determined as in Figure [2]). [Figure 4 of the original.]

The "bond angle" here must be that formed at the *neighboring*, already singly bonded, `L` particle (in some position m_i)—since any one of these particles would only specify a single bond to the target free `L`, and would thus not serve to specify a bond angle at the target `L`.

6.3. If there are two or more of the m_i, select two, form the corresponding bonds, and exit.

This expresses the preferred outcome: if possible, the free `L` particle should form its two bonds with two neighboring, singly bonded, `L` particles.

However, there is a difficulty with the algorithm as so far expressed. Consider, for example, the configuration shown in figure 3. The two neighboring, singly bonded particles are each *separately* eligible to form a bond with the target particle (neither such bond would generate a 45° bond angle *at the neighboring particle*); but if *both* bonds are formed, this will result in a bond angle of 45° at the target `L` particle, which presumably should not be allowed.

I suggest that the "correct" (or at least, non-defective) procedure in this case would be as follows. First, pick one of the m_i (uniformly) at random, and form the corresponding bond. Remove that m_i from the list m, and also any other position(s) that would *now* result in a bond angle of less than 90° at the target particle (because the target particle is, by now, singly bonded, this is a well-defined constraint). If the list m is not now empty, pick another

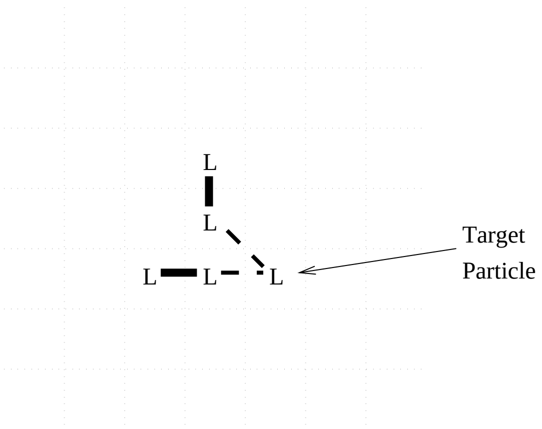

Figure 3: Scenario for formation of bad bond angle.

m_i (uniformly) at random, form the corresponding bond, and exit; but if the list m is empty, attempt to form a second bond with a neighboring *free* `L` particle (the precise procedure for this will be specified in substeps 6.41 through 6.43).

6.4. If there is exactly one m_i, form the corresponding bond.

Given my reformulation of step 6.3, step 6.4 could now be collapsed into it (6.3 would then begin "If the list m is non empty…"). However, the substeps 6.41 through 6.43 are still required, to deal with the situation of attempting to form a second bond with a neighboring free `L` particle. This set of steps is actually invoked from two possible points (within 6.4 and 6.61); thus it is effectively used as a subprocedure, and might have been better explicitly separated out as such. In any case, note carefully that the scenario on entry to substep 6.41 is always that the target particle is now singly bonded, and an attempt is being made to form a *second* bond to a neighboring *free* `L` particle; it will follow that the overall bonding procedure (top-level step 6) will definitely terminate somewhere within substeps 6.41 through 6.43. To put that another way, if control reaches substep 6.41, it will *not* subsequently proceed to step 6.5 or following steps. 6.5 is invoked *only* if the list m becomes empty without *any* bonds having been formed with the target particle.

6.41. Remove from the n_i any which would now result in a bond angle of less than 90°.

6.42. If there are no n_i, exit.

In this case, the target particle must be left just singly bonded.

6.43. Select one of the n_i, form the bond, and exit.

In this case, two bonds have been formed with the target particle.

6.5. If there are no n_i, exit.

As already noted, control reaches this point only in the case that the list m is empty and the target `L` particle still has no bonds (it is still free). In that case we try to form an initial bond with an neighboring free `L` particle. Step 6.5 covers the case that there *are* no neighboring free `L` particles; in which case there is nothing further to try, and the target particle must be left free.

6.6. Select one of the n_i, form the corresponding bond, and drop it from the list.

6.61. If the n_i list is non-null, execute steps 6.41 through 6.43.

6.62. Exit.

Steps 6.61 and 6.62 seem to be redundant (and consequently confusing). The situations they would cover are already covered by steps 6.41 through 6.43 anyway, so it would seem simpler just to stipulate an *unconditional* transfer to 6.41 immediately after 6.6.

Note that the bonding algorithm, as presented, does not rule out the formation of "crossed over" bonds. For example, in the configuration of figure 4, the algorithm would result in the formation of a crossed over bond. This seems counter intuitive, and contrary to the chemical inspiration of the model. However, in the absence of some constraint to the contrary in the algorithm, we must assume that it is intended to be allowed. There is no explicit discussion of this, and it is not apparent that it would contribute anything of significance to the phenomenology of the model.

7. Rebond

This final top-level step of the algorithm is invoked as a subprocedure from the disintegration reaction (step 5.11). While it is not explicitly stated, we assume this subprocedure receives, as an argument, a position where an `L` particle has just disintegrated. The essential effect of this

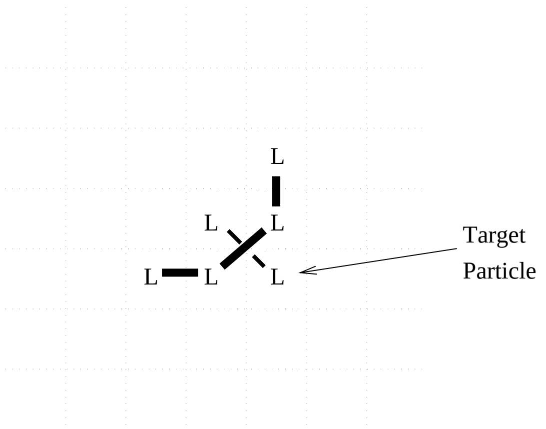

Figure 4: Scenario for formation of crossed bond.

subprocedure is to "reform", as far as possible, bonds in the neighborhood of this position.

"Re-form" is something of a misnomer here: the bonds that have actually been broken as a result of the disintegration obviously cannot be "reformed", since the relevant `L` particle is no longer present. But the step is not even restricted just to "reforming" bonds with `L` particles that were previously bonded to the disintegrated `L`; in fact, this step blindly attempts to form bonds among *all* `L` particles in the neighborhood of the disintegration site, regardless of whether they were involved in, or otherwise affected by, the disintegration. Furthermore, it is perfectly possible that some of the `L` particles that are processed for bonding here may yet actually be subject to disintegration on this same timestep.

The rebonding algorithm itself is deceptively short; it actually hides some moderately complex processing. As with step 6, bonding is effected preferentially on `L` particles that are already singly bonded; bonding to or between free `L` particles is then considered secondarily.

7.1. Form a list of all neighbor positions m_i occupied by singly bonded L's.

7.2. Form a second list, p_{ij}, of pairs of the m_i which can be bonded.

The force of the phrase "can be bonded" is not made explicit here, so we must ask why particular pairs may be excluded. Three distinct cases arise.

Firstly, it is clear that only pairs of `L` particles that are adjacent to each other are eligible for bonding to each other.

Secondly—and trivially—some of the pairs of adjacent, singly bonded, `L` particles may already be bonded to each other.

Finally, since each of the `L` particles currently under consideration is already singly bonded, the proposed new bond would, in each case, be a second bond—and would therefore cause a new bond angle to be established at each of the two particles involved. If either of these two new bond angles would be less than $90°$ then, presumably, that pair must be excluded from consideration.

7.3. If there are any p_{ij}, choose a maximal subset and form the bonds. Remove the L's involved from the list m_i.

The difficulty here is with the phrase "maximal subset", which is not explained in any detail.

However, the idea seems to be as follows. Of the pairs listed in p (say there are k entries in this list), it may not be possible to form *all* the corresponding bonds, because there could be conflicts between them. Specifically, if the same (singly bonded) `L` particle occurs in two different pairs then, clearly, only one of those two pairs can be chosen for bonding. So, the requirement of the algorithm seems to be that various possible *subsets* of pairs may have to be investigated. This would start by considering whether *all* (q) of the pairs can indeed be bonded (i.e. there are actually no conflicts). If so, they are all bonded, and control can pass to the next step; but, if not, then all possible subsets of $k-1$ of the pairs must be considered. If there are no conflicts in one of these subsets, then all of those bonds should be formed (and if there are multiple such, mutually compatible, subsets, then one should presumably be chosen uniformly at random). If no mutually compatible subset of $k-1$ pairs exists, then the subsets of $k-2$ pairs should be considered next and so on. Clearly this process will terminate at $k = 1$ since there can be no conflicts in such subsets.

There is nothing fundamentally difficult about carrying out this procedure. However, it seems extraordinarily complex, for no obvious or explained benefit. A much simpler alternative—which would be expected to yield essentially the same phenomenology—would be to simply pick an element of p at random, and form that bond; then remove any other elements from p that are no longer eligible, and then pick another at random for bonding; continue this process until p is empty. Of course, this procedure is not deterministically guaranteed to generate a "maximal" subset of bonds; but it would seem very strange indeed if the phenomenology of the system depended critically on this.

Nonetheless, it seems that any strict implementation of the algorithm will be required to search deterministically for a maximal subset of p.

7.4. *Add to the bond m_i any neighbor locations occupied by free L's.*

The list m was potentially reduced at step 7.3, by removing from it the positions of any singly bonded `L` particles that have now become doubly bonded. Step 7.4 now simply adds to this list the positions of any free `L` particles in the neighborhood of the disintegration site. Note carefully that the list m now potentially contains both free and singly bonded `L` particles. This should be clearly distinguished from the situation in top-level step 6, where two separate lists were maintained for free and singly bonded links.

7.5. *Execute steps 7.1 through 7.3, then exit.*

The reference to step "7.1" here must be a typographical error: since step 7.1 is concerned with forming the list m in the first place; regenerating it at this point would be incoherent. So we must assume that the intention is to "reexecute" steps 7.2 and 7.3 only. Even this "reexecution" is not completely straightforward. In particular, steps 7.2 and 7.3 must now be interpreted as applying to the more general case that the list m contains a *mix* of both singly bonded and free `L` particles, whereas in the original context it could be assumed that it contained singly bonded particles only.

5 External Review

This section is concerned with comparing the detailed algorithm, as presented in the previous section, with the specific experimental runs documented in (Varela et al. 1974). This comparison reveals a number of discrepancies.

The original paper presents diagrams of two sequences of configurations of the space in the computer model, both taken from a single run, and show "instants" 0–6 and 44–47 respectively (figures 1 and 2 of the original). These figures are described as "...drawn from the print-outs with change of symbols used in the computations". They use an alternative (iconic) set of symbols for the various types of particles. I have redrawn these figures again, this time substituting the same (alphanumeric) symbols as used consistently throughout the rest of the original paper, and in this report. I have also added numeric co-ordinates along two sides of each diagram of the configuration of the space (to facilitate reference to individual cells or particles), and running counts of the numbers of `L` particles and holes in the space (which should be always equal, according to the described reaction scheme). My figures 5 and 6 correspond to the original figure 1; my figure 7 corresponds to the original figure 2.

Varela et al. refer to the separate configurations shown as "successive instants". I presume this is equivalent to single iterations of the algorithm, or "timesteps". This is consistent with `EXP29.FOR`, but will be discussed further below.

The first sequence (steps 0–6, my figures 5 and 6) illustrates the initial, spontaneous, formation of a structure consisting of a closed chain of `L` particles enclosing a `K` particle and also some free `L` and substrate particles. The second sequence (steps 44–47) illustrates the spontaneous rupturing and repair of the bounding chain. It is stated that "...[w]ithin this universe these systems satisfy the autopoietic organisation".

Various problems arise in reconciling these experimental data with the given algorithm.

1. **Hole versus `L` Particle Counts**

   The given reaction scheme has two `S` particles being required to generate one `L` particle; and one `L` particle disintegrating into two `S` particles. On the experimental run the space is initially configured with one `K` particle and is otherwise filled with `S` particles ($T = 0$). Each production event should generate one `L` particle and one hole, removing two `S` particles; and each disintegration event should generate two `S` particles, removing one `L` particle and one hole. It follows that the number of `L` particles ($N(L)$) should always equal the number of holes ($N(H)$). However, in the diagrams shown, this is *never* the case (except, of course, for $T = 0$). For most of the time, the number of `L` particles exceeds the number of holes by about 6:1. This

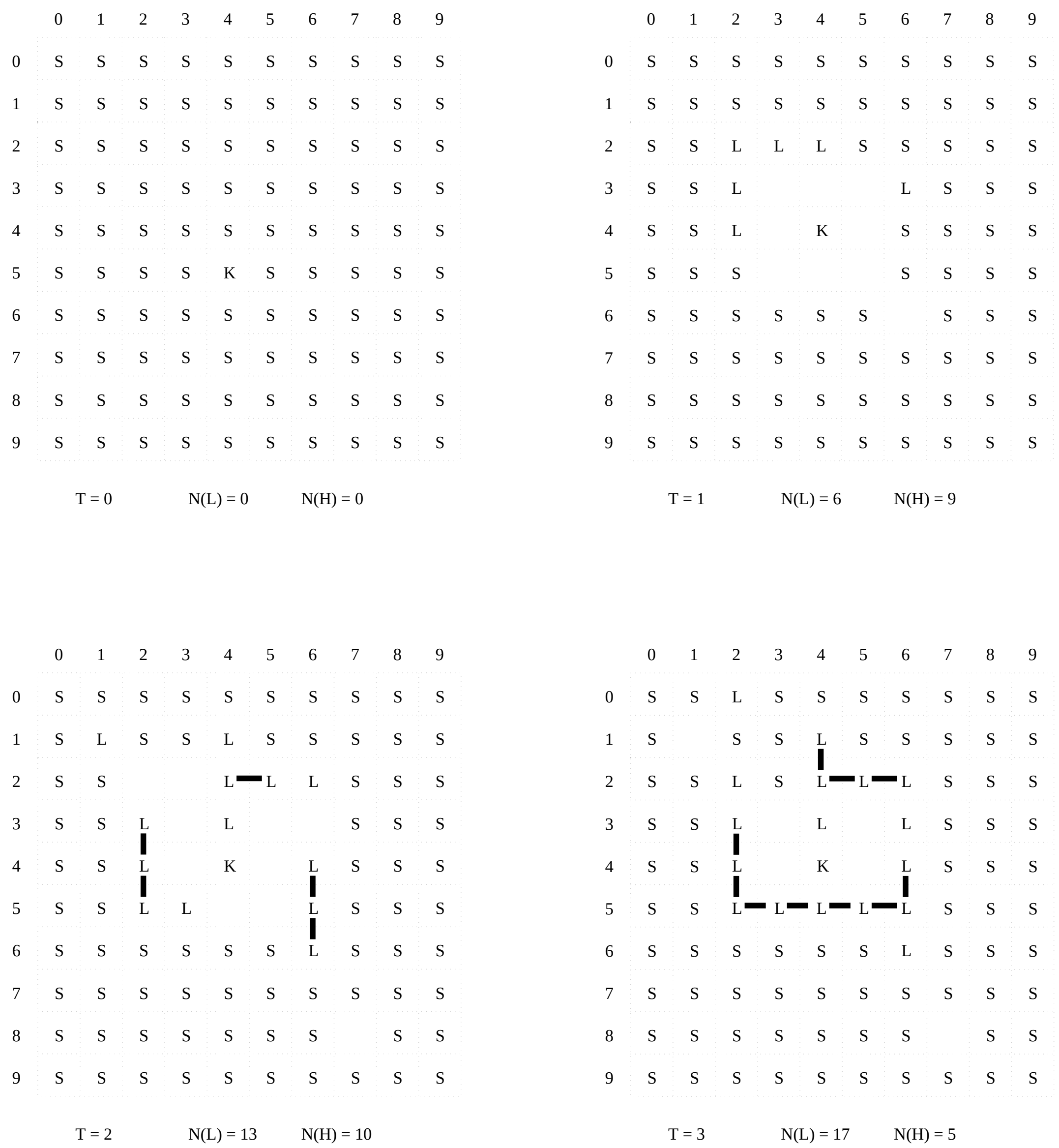

Figure 5: Experimental run: steps 0–3.

is a gross discrepancy, which is not dependent on any fine details of the algorithm, and is difficult to explain. It is conceivable that the diagrams are showing only a window onto a larger space (such as the 30×30 space used in `EXP29.FOR`), and that the extra holes have migrated out of the area shown. However, this interpretation would face a special additional difficulty with timestep $T = 1$. At this time the number of holes *exceeds* the number of `L` particles, and, further, there has not been time for any significant interaction from the boundaries of the area shown. I can see no consistent and reasonable way of fully reconciling these variations in the ratio between $N(L)$ and $N(K)$ with the specified reaction scheme.

2. **Production Reaction Rate**

   The algorithm restricts production to a rate of one `L` particle per timestep per `K` particle. Yet, in the experimental run, we apparently observe production of *six* `L` particles (by the one available `K` particle) in going from $T = 0$ to $T = 1$. Similarly, from $T = 1$ to $T = 2$ we see the number of `L` particles increasing to 13, implying production of at least a further 7 `L` particles in that step. Similar, though somewhat less dramatic, discrepancies occur between $T = 2$ and $T = 3$, and between $T = 4$ and $T = 5$. Even on those steps where the reaction rate is not apparently excessive, it seems likely that this is due only to limited availability of `S` particles in the vicinity of the `K` particle.

   A conceivable interpretation of this is that the time "instants" in the diagram correspond to multiple iterations of the algorithm (time "steps"). However, there is *no* evidence in the discussion of the paper that would support this. Further, it would not be consistent with the fact that, at $T = 1$, *none* of the produced `L` particles are yet bonded.

   Even if we suppose that multiple production events might be allowed per `K` particle per timestep, there are additional, subsidiary, peculiarities of the configuration at $T = 1$. First, the production reaction requires two `S` particles adjacent to the `K` particle. Thus, to produce six `L` particles would require twelve `S` particles—but there are, at most, only eight positions immediately adjacent to the `K`. Furthermore, the spatial distribution of the produced `L` particles seems strange. None of the produced particles at $T = 1$ are immediately adjacent to the `K` particle—even though the production reaction specifies that a newly produced `L` is *always* adjacent to the `K`.[13] Finally, of the six produced `L` particles, five are immediately adjacent to each other—which would seem very unlikely if the choice of reaction sites, in the vicinity of the `K` particle, were stochastic.

   However: it turns out that `EXP29.FOR` yields almost exactly the same pattern of initial production as illustrated at $T = 1$ and an examination of its code reveals answers to all the discrepancies noted here.

   Firstly, `EXP29.FOR` does *not* restrict production to just one event per `K` particle per timestep, but actually permits up to one event *per (Moore) neighbor* of a `K` particle per timestep—which is to say, up to 8 production events per catalyst per timestep.

   Secondly, for each Moore neighbor of a `K` particle, if that neighbor is an `S` particle, `EXP29.FOR` searches in the Moore neighborhood *of that `S` particle in turn*, seeking a second `S` particle for the production reaction. Thus, while the two reacting `S` particles must be adjacent to each other, only one of them need be adjacent to the mediating `K` particle. This effectively increases the neighborhood for the production reaction to two positions away from the `K` particle, an area of size 25. This greatly increases the availability of potential reactants. It also introduces the possibility that the produced `L` will actually be two positions away from the `K` particle (even before any movement).

   Finally, in `EXP29.FOR`, the searching for reactants, and the selection of which to replace by a `L` particle, is deterministic. This fully accounts for the systematic bias of the initial production to the Northwest of, and two positions away from, the `K` particle.

3. **Catalyst Mobility**

   In the course of the times $T = 0$ through $T = 6$, the `K` particle moves just once. In the times $T = 44$ through $T = 47$ it does not move at all—and, indeed, is still in the same position as at times $T = 1$ through $T = 6$. This is not consistent with the algorithm (top-level step 3) which should force the `K` particle to move virtually every timestep (its maximum "mass" meaning that it can displace, or at worst, swap with, *any* other kind of particle).

[13] Note that `L` and `K` particle movements come *before* production in the algorithm; so produced `L` particles cannot separate from the `K` particle within a single iteration of the algorithm.

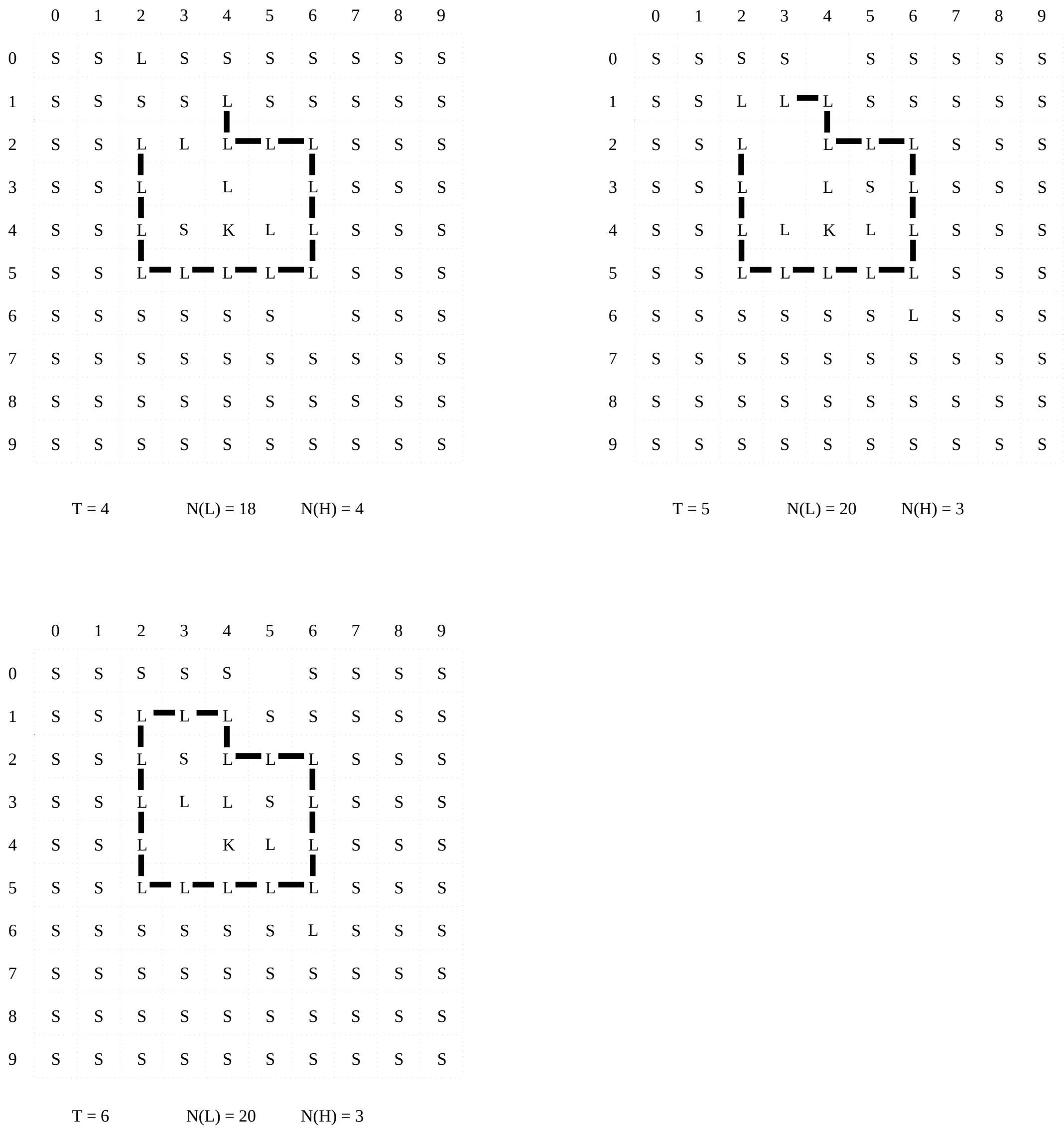

Figure 6: Experimental run: steps 4–7.

It might be conjectured that, in redrawing the diagrams, the authors simply recentered on the `K` particle each time. However, this idea is not consistent with the fact that the `K` particle *does* move once; and, more substantively, it is not consistent with the fact that the *bonded* `L` particles, which are specified to remain absolutely immobile in the space, do not change their positions *relative* to the `K` particle.

4. **Bonding Angles**

   The algorithm stipulates that bond angles less than 90° are to be disallowed. The diagram illustrating this (figure 4 of the original, redrawn as figure 2 here) shows a bond angle of 135°. This clearly suggests that, in general, bonding is to be allowed along diagonals of the space, except only in the case where that would result in a bond angle of less than 90°. Yet, in all the experimental configurations shown, all bonds occur only in the orthogonal directions, never in the diagonal directions. Clearly, the possibility of diagonal bonding would have a significant effect on the shapes of `L` particle chains which may form, and may thus significantly affect the dynamics of any autopoietic organisation which the model might support.

   Again, the `EXP29.FOR` code may shed *some* light on the this. In that program, bonds are not explicitly represented or displayed, though they may be said to be implicitly encoded in the transformation and movement processes. If the diagrams of (Varela et al. 1974) derive from a closely related program, then, in redrawing the diagrams from the original printouts, the drawing of explicit bonds would have been at best schematic, or suggestive, rather than reflecting definite operations of the program.

5. **Bonding Reaction Rate**

   As already discussed at length, the published algorithm makes provision for bonding to occur several different times within each iteration. In particular, every time a free `L` particle is produced or moves, it is supposed to be considered for bonding. Considering now the experimental configuration at $T = 1$ we have (apparently) six `L` particles, all of which have just been produced, and five of which are adjacent to each other. If the bonding procedure (top-level step 6) were applied to the latter five particles then they would all have become bonded. Yet no bonds are shown between any of them. This is the most dramatic discrepancy with regard to the bonding algorithm—but there are other instances also.

   In particular, consider the transition from $T = 46$ to $T = 47$. A free `L` particle apparently moves from position (3,2) to position (3,3), where it becomes immediately adjacent to another free `L` at position (4,3)—yet no bond is formed between them.

   This is extremely significant, as the formation of such bonds—between free `L` particles enclosed within the "membrane"—would prevent such particles being available to migrate to a rupture site. The possibility of such reactions would thus appear to undermine a critical aspect of the claimed autopoietic organisation.

   Note that this discrepancy is *not* a stochastic or probabilistic one: the algorithm as given makes bonding effectively deterministic where possible, (the only stochastic element arises in choosing between multiple available bonding configurations, not in deciding whether bonds should form at all). Thus, it does not seem plausible that, in the given experimental configurations, it simply "chanced" that certain `L` particles did not bond to each other.

   However, we again find that, if the program in use was closely related to the `EXP29.FOR` code (rather than the algorithm as stated) this may provide some explanation.

   Firstly, the overall algorithm of `EXP29.FOR` involves scanning the space in a systematic, West to East, then North to South, raster pattern. There are several such scans per timestep; but production and bonding are implemented in the same scan. This happens to interact with the deterministic pattern of production events in just such a way that all but one of the initially produced `L` particles appear in positions *that have already been processed in that scan.* Thus they are not processed for bonding until the second timestep.[14]

   Secondly, `EXP29.FOR` implements some special constraints on bonding. While I have not been able to fully decipher the intentions here, I have been able to explicitly identify *some* of these constraints. The most crucial seems to be that

[14] There is one `L` initially produced by `EXP29.FOR` that *is* still processed for bonding; but it is isolated from the other `L` particles and thus cannot bond in practise within that first timestep. In any case, that particular `L` particle does *not* appear in figure 5, indicating either an error in redrawing, or some detailed discrepancy between the code of `EXP29.FOR` and that actually used to generate this diagram.

	0	1	2	3	4	5	6	7	8	9
0	S	S	S	S	S	S	S	S	S	
1	S	S	L	L	L	S	S		S	S
2	S	S	L	S	L	L	L	S	S	S
3	S	S	L	L	L	L	L	S	S	S
4	S	S	L		K	L	S	S	S	S
5	S	S	L	L	L	L	S	S	S	S
6	S	S	S	S	S	S	L	S	S	S
7	S	S	S	S	S	S	S	S	S	S
8	S	S	S	S	S	S	S	S	S	S
9	S	S	S	S	S	S	S	S	S	S

T = 44 N(L) = 19 N(H) = 3

	0	1	2	3	4	5	6	7	8	9
0	S	S	S	S	S	S	S	S	S	S
1	S	S	L	L	L	S	S	S	S	S
2	S	S	L	L	L	L	L	S	S	S
3	S	S	L		L	L	L	S	S	S
4	S	S	L	S	K		S	S	S	S
5	S	S	L	L	L	L	S		S	S
6	S	S	S	S	S	S	L	S	S	S
7	S	S	S	S	S	S	S	S	S	S
8	S	S	S	S	S	S	S	S	S	S
9	S	S	S	S	S	S	S	S	S	S

T = 45 N(L) = 18 N(H) = 3

	0	1	2	3	4	5	6	7	8	9
0	S	S		S	S	S	S	S	S	S
1	S	S	L	L	L	S	S	S	S	S
2	S	S	L		L	L	L	S	S	S
3	S	S	L	L	L		L	S	S	S
4	S	S	L	S	K	L	S	S	S	S
5	S	S	L	L	L	L	S		S	S
6	S	S	S	S	S	S	L	S	S	S
7	S	S	S	S	S	S	S	S	S	S
8	S	S	S	S	S	S	S	S	S	S
9	S	S	S	S	S	S	S	S	S	S

T = 46 N(L) = 18 N(H) = 3

	0	1	2	3	4	5	6	7	8	9
0	S	S	S		S	S	S	S	S	S
1	S	S	L	L	L	S	S	S	S	S
2	S	S	L	S	L	L	L	S		S
3	S	S	L		L		L	S	S	S
4	S	S	L	L	K		L	S	S	S
5	S	S	L	L	L	L	L	S	S	S
6	S	S	S	S	S	S	L	S	S	S
7	S	S	S	S	S	S	S	S	S	S
8	S	S	S	S	S	S	S	S	S	S
9	S	S	S	S	S	S	S	S	S	S

T = 47 N(L) = 19 N(H) = 5

Figure 7: Experimental run: steps 44–47.

on bonding of a free `L` particle, and appears at lines 232–3 of the listing. The effect of this is that a free `L` particle can form a bond only if there is no more than one doubly-bonded `L` particle in its immediate (Moore) neighborhood. In effect, this tends to inhibit the formation of bonds by free `L` particles that are immediately adjacent to an existing *chain* of `L` particles. It establishes a sort of bond inhibition "field" around such a chain. In particular, it is a mechanism to prevent free `L` particles, produced inside a membrane formed by the closure of such a chain, from bonding to each other (provided, of course that the space inside is not too large).

Clearly, this can explain the otherwise puzzling lack of bond formation between $T = 46$ and $T = 47$, already discussed. More importantly, it provides a possible mechanism to ensure the continued mobility of the `L` particles trapped inside the membrane, and thus their continued availability to migrate to a rupture site. This certainly appears to be a very important mechanism in the attainment of the desired autopoietic closure. *Yet, there is no hint of such a mechanism either in (Varela et al. 1974), nor even in the contemporary descriptive account of `PROTOBIO`.* This suggests to me the possibility that, at the time of writing the paper, the authors themselves may no longer have been fully aware of all the interactions (or their significance) coded into the model.

We have seen how a number of the most puzzling discrepancies between the experimental results and the algorithm of (Varela et al. 1974) can be explained by assuming that the diagrams were produced by a program at least closely related to that of `EXP29.FOR`. However, a corollary of this is that the algorithm implemented by `EXP29.FOR` is indeed significantly different from that actually described in (Varela et al. 1974). This suggests the possibility that the illustrated experimental results may have derived from a relatively early version of the model (i.e. closely related to `EXP29.FOR`), while the algorithm may date from a much later (more complex and/or sophisticated) version. It is impossible to determine this more exactly, so long after the event. However, in support of this conjecture, I may note that Varela's own published account (Varela 1996) explains that this work was done under very complex and volatile circumstances in Chile, shortly before the violent overthrow of the Allende government. Furthermore, there was a substantial delay between the original work and the preparation of the final published manuscript. Taken together, it seems to me that these factors could reasonably explain an accidental (albeit substantial) version mismatch between the published algorithm and the experimental results.

6 Conclusion

The substantive criticisms of (Varela et al. 1974) presented in this report can be summarised as follows:

- The presented algorithm implements the model chemistry in a way which is obscure and counter-intuitive, particularly in relation to the bonding reaction.
- The algorithm itself is unclear and ambiguous in places.
- The presented experimental data are not consistent with the algorithm.
- In fact, it seems likely that the experimental data actually derived from a program closely related to that of `EXP29.FOR`, and thus quite different in substantial detail from the presented algorithm.
- By critical consideration of the experimental results and of the `EXP29.FOR` code, an important—perhaps crucial—constraint on the bonding reaction has been re-discovered: *namely that bonding of free `L` particles be inhibited in the immediate vicinity of a chain of such particles.* This interaction can now be clearly recognised as important to the intended autopoietic phenomena. However, this interaction has not, to my knowledge, been explicitly mentioned in any previous description of the model.

All these defects arise at least in part from the fact that the target paper dates from the very early days of this kind of biological modeling. Indeed, the idea of using this particular form of computational model should be seen as highly original for its time, and the authors deserve considerable credit for that. I regard the concept of autopoiesis, although still somewhat vague, to be a very useful one in clarifying the conditions for discussing the origin of life. To this extent, the concept, and its computational modeling or realisation, is potentially very significant for the development of the field of *Artificial Life* (Langton 1989).

The criticisms raised here should not be regarded as detracting from these genuine and important achievements.

Nonetheless: from the point of view of continuing or developing the particular model presented in the target paper, these criticisms represent real and serious obstacles, which need to be overcome. In particular, I make the following concluding points:

- The discrepancies between the experimental runs and the given algorithm mean that the demonstration of basic autopoietic organisation in computational media claimed by the target paper must be regarded as weak.

- There are published descriptions of independent implementations of autopoietic organisation in computational media (Zeleny 1977, Zelany & Pierre 1976). However, I have found the corresponding algorithm descriptions even more difficult to follow than the original presented in (Varela et al. 1974). The corresponding code *was* made available at the time, but I understand it is no longer so.[15]

- As a general principle, the publication of adequate information to allow independent assessment and testing of any claims made is a crucial distinguishing feature of the scientific enterprise. At the time of original publication of (Varela et al. 1974) it was neither technically nor economically feasible to include full source code with the publication (in machine readable format). Therefore the authors opted to include a semi-formal algorithm. This was surely the best practise possible at the time. However, as we have seen, it suffers from serious practical deficiencies, and does not actually meet the desired standard of allowing independent assessment and testing. Critically, it seems, in this particular case, that there was an important interaction present in the code but not mentioned at all in the description; and this may not be all that exceptional an occurrence.

- The technology is now available to augment the use of a published *summary*, or *algorithm*, of model code. At a basic level, the *World Wide Web* allows full source code to be made easily available to any interested researcher, in machine readable format. At a more advanced level, the emerging availability of standardised modeling environments, such as the Swarm simulation system[16], should significantly ease the burden involved in independent examination and assessment of such published models.

- I suggest that it would thus be beneficial to re-implement the original algorithm—and various related (ideally, simpler) systems modeling the same qualitative chemistry—to establish how robust, or otherwise, the autopoietic phenomenology is in these systems; and it may be especially beneficial to do this using the Swarm simulation system, and make full source code of the implementation available via the World Wide Web. This is the focus of a continuing research effort, and is the subject of a planned companion report.

[15] Milan Zeleny, personal communication.

[16] `http://www.santafe.edu/projects/swarm`

References

Langton, C. G. (1989), Artificial Life, *in* C. G. Langton, ed., 'Artifical Life', Vol. VI of *Sante Fe Institute Studies in the Sciences of Complexity*, Addison-Wesley Publishing Company, Inc., Redwood City, California, pp. 1–47.

Maturana, H. R. & Varela, F. J. (1973), Autopoiesis: The Organization of the Living, in *Autopoiesis and Cognition: The Realization of the Living* (Maturana & Varela 1980), pp. 59–138. Dated 1973. First published 1972 in Chile under the title *De Maquinas y Seres Vivos*, Editorial Universitaria S.A.

Maturana, H. R. & Varela, F. J. (1980), *Autopoiesis and Cognition: The Realization of the Living*, Vol. 42 of *Boston Studies in the Philosophy of Science*, D. Reidel Publishing Company, Dordrecht, Holland. With a preface to 'Autopoiesis' by Stafford Beer. Series editors: Robert S. Cohen and Marx W. Wartofsky.

Varela, F. J. (1996), 'The Early Days of Autopoiesis: Heinz and Chile', *Systems Research* **13**(3), 407–416.

Varela, F. J., Maturana, H. R. & Uribe, R. (1974), 'Autopoiesis: The Organization of Living Systems, its Characterization and a Model', *BioSystems* **5**, 187–196.

Zelany, M. & Pierre, N. A. (1976), Simulation of Self-Renewing Systems, *in* E. Jantsch & C. H. Waddington, eds, 'Evolution and Consciousness: Human Systems in Transition', Addison-Wesley Publishing Company, Reading, Massachusetts, chapter 7, pp. 150–165.

Zeleny, M. (1977), 'Self-Organization of Living Systems: A Formal Model of Autopoiesis', *International Journal of General Systems* **4**, 13–28.

Acknowledgments

I am particularly grateful to Francisco Varela for helping me with questions about the original model, and in giving me access to related, previously unpublished, materials (some of which are now presented in appendix A). I am also grateful to Marita Prandoni, of the Santa Fe Institution, for her assistance in translating some of these materials.

Initial work on this detailed re-analysis of (Varela et al. 1974) was carried out with the help of my former student, Hyder Aswad.

All errors remain, of course, my own responsibility.

This report was written while I was a visiting researcher (1996/97) with the Swarm Project Group at the Santa Fe Institute. This visit was made possible by generous support from Dublin City University and the Swarm Project. Swarm has been supported by Grant No. N00014-94-1-G014 from the Department of the Navy, Naval Research Laboratory, acting in cooperation with the Advanced Research Projects Agency. Swarm has also benefited from earlier support from The Carol O'Donnell Foundation and from Mr. and Mrs. Michael Grantham.

I am indebted to the School of Electronic Engineering in DCU (particularly through the agency of its Head, Charles McCorkell) for continuing encouragement.

The html version of the article could not have been created without the benefit of the **LaTeX**2HTML[17] translator developed by Nikos Drakos[18] and the Harvard[19] extensions for author/date citations developed by Peter Williams[20] and Thorsten Schnier.

Revision History

First published in January 1997, simultaneously as Working Paper No. `97-01-001` of the Santa Fe Institute and as Technical Report No. `bmcm9701` of the School of Electronic Engineering of Dublin City University.

[17] http://cbl.leeds.ac.uk/nikos/tex2html/doc/latex2html/latex2html.html

[18] http://cbl.leeds.ac.uk/nikos/personal.html

[19] http://www.arch.su.edu.au/~peterw/latex/harvard/

[20] http://www.arch.su.edu.au/~peterw/index.html

Author Contact Information

Barry McMullin

Santa Fe Institute
(Visiting Researcher 1996/97)
1399 Hyde Park Road
Santa Fe, NM 87501
USA

Voice: +1-505-984-8800
Fax: +1-505-982-0565
E-mail: McMullin@santafe.edu

School Of Electronic Engineering
Dublin City University,
Dublin 9,
IRELAND

Voice: +353-1-704-5432
Fax: +353-1-704-5508
E-mail: McMullin@eeng.dcu.ie
Web: http://www.eeng.dcu.ie/~mcmullin

Copyright

Retrieval

The resources comprising this report are electronically retrievable, in various formats, via the World Wide Web. The root (directory) URL is:

`http://www.santafe.edu/sfi/publications/Working-Papers/97-01-001/`

This is also mirrored at:

`http://www.eeng.dcu.ie/~autonomy/bmcm9701/`

Either of these URL's will directly access the root node of the HTML format report. Alternatively, an additional filename may be appended, to retrieve one of the other formats, or related resources, as follows:

- The complete set of files making up the HTML format of the report is available in the single compressed archive file `bmcm9701.html.tar.gz` (c. 124 KByte), which can be conveniently downloaded for local and/or offline browsing.
- A printable version of the report is available in postscript format as file `bmcm9701.ps` (c. 549 KByte, 42 pages) or in compressed form as file `bmcm9701.ps.gz` (c. 145 KByte).
- The original LaTeX source of the report is available as file `bmcm9701.tex` (c. 145 KByte) or in compressed form as file `bmcm9701.tex.gz` (c. 46 KByte).
- The BibTeX source for the reference list is available as file `bmcm9701.bib` (c. 4 KByte).
- The `FORTRAN IV` source code listed in appendix A.3 is available in the files `exp29.for` (c. 9 KByte) and `nal.for` (<1KByte).
- Finally, a complete set of all the files just listed is available as a single compressed archive file `bmcm9701.tar.gz` (c. 317 KByte).

A Appendix: PROTOBIO

This appendix contains two previously unpublished resources, relating to the early computer model of Autopoiesis by Varela et al., then referred to as `PROTOBIO`.

First there is a short text describing the model. I provide here both the original Spanish text, verbatim, and a rough translation into English[21]. I include some additional comments of my own interspersed with the translation.

Secondly, I include the `FORTRAN IV` source code listing of a version of the described program.[22] This was produced by manually rekeying a photocopy of the original program listing. In general, I attempted to preserve the original listing exactly verbatim, including comments, indentation, etc. The quality of the photocopy was poor, and there may be resulting errors in the rekeyed version. However, it has been successfully compiled. The program uses one external subroutine, `NAL`. A handwritten comment on the listing identified this as a "library" routine; however, no further information is available. From inspection of the code I have inferred that this was a random number generator. Thus, I also provide here a listing for a sample implementation of this subroutine, based on my inferred semantics for it; using this, I have successfully built and executed the program on a Sun (SunOS 5.5, Sun f77 compiler) platform.

These materials are undated, but probably originated c. 1971. The authors are not explicitly recorded, but Francisco Varela attributes them collectively to himself, Humberto Maturana and Ricardo Uribe. The documents were only recently rediscovered by Francisco Varela, and I am very grateful to him both for allowing me access to them, and for giving his permission for their publication here.

Note that it is not possible to be sure how the materials presented here relate to the results finally presented in (Varela et al. 1974).

[21] I am indebted to Ms. Marita Prandoni of the Santa Fe Institute for her assistance in making this translation.

[22] This is also available as a standalone source code file—see the *Retrieval* section.

A.1 Spanish Text

<u>PROTOBIO</u>

El programa `PRO1` simula un ambiente bidimensional en el cual interactúan tres elementos: `A`, `B` y `M`, representados `A` por `"@"`, `B` por `"·"` y `M` en sus distintas fases pro `"-"`, `"+"` y `"*"` en las figuras y por `M-`, `M+` and `M*` en el texto, respectivamente.

El ambiente es una superficie de coordenadas X e Y de dimensiones 30 por 30 en nuestros experimentos que se observa en cada instante. Sobre esta superficie se ubican uno o más elementos `"@"` y numeros `"·"` sobre cuya generación no nos pronunciaremos por ahora.

Los elementos `"·"` se mueven al azar sobre la superficie y permanencen en cada neuva posición durante tres instantes.

`"@"` se mueve al azar sobre la superficie, sin salirse de ella, un espacio en cada instante. El movimiento de `"@"` es similar al del rey del ajedrez y sólo se mueve si la nueva ubicación está desocupada. Los `"·"` vecinos a `"@"` no se mueven mientras permanezcan en esa situación.

`"@"` es capaz de "sintetizar" un elemento `M` a partir de dos `"·"` que se ubiquen en su vecindad. Por comodidad para la explicación este elemento `M` se representará en las figuras por `"-"` y en el texto por `M-`

<u>Ejemplos:</u>

```
  .  .              -

@         =>    @                @       =>    @

                                    .

                                      .             -


.

. @       =>  - @
```

Este elemento `M-` se mueve un espacio en cada instante en una dirección al azar. A diferencia del elemento `"@"`, si encuentra la nueva ubicación ocupada, busca (al azar) otra ubicación desocupada, durante ocho veces[23]. El elemento `M-` se mueve hasta que queda vecino a otro `M-` (sólo uno), y luego queda inmovil. Los elementos `M` que lleguen a esta situación

[23] *[Escritura] NOTA: En esta busqueda no cuenta las veces en que encuentra elementos* `M` *en las fases + & * en la neuva ubicación. Esto sultimo lo hace moverse preferentemente a lo largo de canales de* `M`*.*

se representarán en las figuras por "+" y en el texto por M+.

Ejemplos:

```
                    -        +       -       +
                        =>               =>
 - -   =>  + +    -        +       -       +
```

Este par de elementos M+ es capaz de "captar" e inmovilizar otros M- que se ubiquen en cualquiera de las posiciones señaladas por 0:

```
0 0 0 0

0 + + 0

0 0 0 0
```

Ejemplos:

```
   -          +
       =>
 + +        + +          + + -  =>  + + +


                                -          +

   -          +               +    =>   +
       =>
 + +        + +             +          +
```

Todos aquellos M+ flanqueados por otras M+ perderán la capacidad de "captar" elementos M- y se representarán en las figuras por "*" y en el texto por M*. Mientras M+ esté en el extremo de una cadena de M*, podrá "captar" elementos M- para anexarlas a la cadena.

Los elementos M decaen después de un número finito de instantes, en dos elementos "·".

Los M* que pasen a ser extremo de una cadena de M* por decaimiento de sus vecinos volverán a ser M+ y podrán "captar" elementos M- en las siguientes posiciones (0):

```
0 0

0 + * * *
        *
0 0     *
        *
        *
      0 + 0

      0 0 0
```

Las cadenas de M* pueden cerrarse como si formasen una membrana y si en este proceso queda un "@" englobado por ellas, se observa que el sistema {A, B, M} se mantiene como una unidad dinámica a pesar del decaimiento de los M y de la movilidad de los "@", mientras este no deje de estar suficientemente englobado por el conjunto de M. Si el elemento "@" se escapa de la cadena englobante de M, la unidad desaparece.

La mantención de este sistema dinámico se debe a que el elemento "@" sigue generando elementos M- a partir de elementos "·", para los cuales la "membrana" es permeable. De esta manera, la "membrana" (o cadena cerrada) de elementos M* llega a englobar una población de elementos M- que está pronta a regenerar la "membrana" en los puntos en que decaen los elementos M* que la forman. En otras palabras, se produce el siguiente ciclo: El elemento "@" sintetiza elementos que formarán una "membrana" que a su vez impedirá que este elemento se escape.

Todas aquellas cadenas de M* que se cierren sin un elemento "@" en un interior están destinadas a desaparecer con el decaimiento de las M*. La probabilidad de que sean regeneradas por elementos M- que vienen desde afuera de la "membrana" es muy baja.

Por último, es evidente que estos sistemas {A, B, M} pueden extenderse a sistemas tridimensionales. Se estima que el realizar un modelo tridimensional implica dificultades que no se justifica superar en este momento dado que la extensión del modelo propuesto a tres dimensiones no produce modificaciones conceptuales. *[Escritura] En el modelo tridimensional, los movimientes de @ y M- se produciuan [?] en un espacio tridimensional. Las cadenas de M* serian [?] ahora superficies cuyos bordes serian [?] M+ capaces de "captar" neuvos M- que hacian [?] crecer otras superficies posibilitando il cierre de ellos, englobando o no un elemento @. El volumen interno de esta "mebrana-esfera" se poblará de elementos M- que esterán pronta a regenerar la "membrana" en los puntos en que se destruya [?] por decaimiento de las M*.*

A.2 English Translation

PROTOBIO

The program PRO1 simulates a two dimensional environment, within which three [kinds of] particles interact: A, B and M. A is denoted by "@" and B by "·". The distinct phases of M are denoted by "-", "+" and "*" in the diagrams, and by M-, M+ and M* in the text, respectively.

The environment is a plane with coordinates X and Y, of dimensions 30 by 30 in our experiments. This can be observed at each timestep. One or more "@" particles, and numerous "·" particles are located on the plane. The generation [origin?] of these will not be discussed here.

The "·" particles move randomly over the plane, remaining in each new position for three timesteps.

> [Comment: This was presumably designed to reduce the mobility of substrate relative to the other dynamics of the model. No other details are provided of the rationale for this. - BMcM]

"@" [particles] move randomly over the plane, without leaving it, one space at a time. The movement of "@" is similar to that of the king in chess, and can occur only if the new location is vacant. "·" particles do not move as long as they are adjacent to an "@".

> [Comment: Making "·" particles immobile while adjacent to @ should tend to increase the rate of the production reaction. - BMcM]

"@" is capable of "synthesising" an M particle from the combination of two "·" particles which are located in its vicinity. For simplicity of explanation, this M particle is represented in the diagrams by "-" and in the text by M-.

Examples:

```
 . .           -

@       =>    @              @       =>    @

                              .

                               .           -



 .

 . @    =>  - @
```

This M- particle moves one position in a random direction in each timestep. In contrast to the "@" particles, if it encounters a new location which is occupied, there will be a search (at random) for another vacant location, for a maximum of eight iterations.[24] The M- particle will continue moving until it comes into the vicinity of another M- (just one), and then becomes immobile. The M particles which arrive in this situation are represented in the diagrams by "+" and in the text by M+.

Examples:

```
                        -        +        -       +
                           =>                 =>
 - -  =>  + +       -        +        -       +
```

This pair of M+ particles has the capacity to "capture" and immobilise another M- if it arrives in any of the locations symbolised [below] by 0:

```
0 0 0 0

0 + + 0

0 0 0 0
```

Examples:

```
    -         +
       =>
 + +       + +          + + -  =>  + + +


                              -         +

   -       +            +    =>    +
       =>
 + +       + +         +          +
```

Each of the M+ particles which is flanked [on both sides?] by another M+ loses the capacity to "capture" M- particles and is represented by "*" in the diagrams and by M* in the text. Meanwhile, those M+ [particles] at the ends of the chain of M* particles can still capture M- particles and incorporate them into the chain.

[24] *[Handwritten] Note: In this search, the times when an M particle in the + or * state is encountered in the new location are not counted. In this way there will be preferential movement along channels [?] of M.*

[Comment: Note that this early discussion did not refer explicitly to bonds, bond angles, etc.; nor were bonds explicitly represented in the FORTRAN code. However, the basic concepts described here seems to be entirely compatible with the subsequently published description in terms of explicit bonds (Varela et al. 1974). - BMcM]

The M particles disintegrate into two "·" particles after a finite number of timesteps.

[Comment: This is somewhat ambiguous. It might be interpreted as meaning that all M particles have the same, fixed, lifetime; or, perhaps, that they are assigned lifetimes chosen (randomly) from some specified range when they are created. Either way, however, it is noticeably different from the mechanism eventually described in (Varela et al. 1974), where disintegration may occur with a fixed probability per timestep, independently of the "age" of the particle. Having said this, it is not apparent that these variations would have any substantive effect on the phenomenology. - BMcM]

The M* [particles] which happen to be at the end of a chain of M* [particles] following a disintegration in their vicinity, revert to being M+ [particles] and can "capture" M- particles in the following positions (0):

```
O O

O + * * *
         *
O O      *
         *
         *
       O + O

       O O O
```

[*sic*: It seems clear that this diagram was originally mistyped, since the * particles are not aligned on the same lattice as the + particles and the O markers. However, the general intention is clear, and actually seems to anticipate the later stipulation that bond angles must not be less than 90°. - BMcM]

A membrane will be formed if the chain of M* [particles] closes; and if an "@" particle is enclosed in this process then it will be observed that the system {A, B, M} maintains itself as a dynamic unity by balancing the disintegration of the M [particles] against the mobility of the "@" particle, while keeping [the "@" particle?] sufficiently enclosed by the conjunction of the M [particles]. If the "@" particle should escape from the enclosing chain of M [particles], the unity will disappear.

The maintenance of this dynamic system requires that the "@" particle must continue to produce M- particles by combining the "·" particles; for which reason the "membrane" is permeable to the latter [the "·" particles]. In this way, the "membrane" (or closed chain) of M* particles leads to the enclosure of a population of M- particles, which are ready to repair the "membrane" at those points where the M* particles forming it disintegrate. In other words, the following cycle is produced: the "@" particle synthesises particles which form a "membrane" which, in turn, prevents this particle from escaping.

Any chain of M* [particles] which closes without enclosing an "@" particle is destined to disappear with the disintegration of the M* [particles]. The probability that it would be repaired by M- particles that come from outside the "membrane" is very low.

In conclusion, it is evident that the system {A, B, M} could be extended to a three dimensional system. [However] it is judged that the implementation of a three dimensional model would involve difficulties which are not worth overcoming at this stage, as the extension of the proposed model to three dimensions would not involve any conceptual modifications. *[Handwritten] In the three dimensional model the motion of the @ and M- particles would take place in a three dimensional space. The chains of M* [particles] would now become surfaces, with edges composed of M+ [particles] capable of "capturing" new M- [particles] allowing the surface to grow and close, enclosing the @ particle. The volume inside this "spherical-membrane" would become populated by M- particles, ready to repair the "membrane" at the points where it is destroyed [disrupted?] by disintegration of the M* particles.*

A.3 FORTRAN IV Program Listings

Dec 16 12:13 1996 exp29.for Page 1

```
C EXPERIMENTO 29
      INTEGER*2 HAB(30,30),REL(30,30),MOV2(30,30)
      INTEGER   B(8)/'.','-','+', '*',' ',' ',' ','@'/
      INTEGER ESC(30,30)
      INTEGER*2 P8X(10),P8Y(10)
       INTEGER XMAX,YMAX
      INTEGER*2 DM,VID1,VID2,VID3,VID4,VID8,DT,TMAX,TESC,T
      INTEGER*2 XHR1,XHR2,YHR1,YHR2,X,Y
      INTEGER*2 XH1,XH2,YH1,YH2
      INTEGER*2 X8,Y8,D1,D2,D,X81,X82,X83,Y81,Y82,X8A,Y8A,XPF8,YPF8
      INTEGER*2 XPF0,YPF0
      INTEGER*2 X1,Y1,X11,X12,X13,Y11,Y12,X1A,Y1A
      INTEGER*2 X2,Y2,XD2,YD2,XD21,YD21,XPF2,YPF2,X21,X22,X23,Y21,Y22
      INTEGER*2 X2A,Y2A,C2,XPFA2,YPFA2,C23,C24
      INTEGER*2 X3,Y3,C3,X31,X32,X33,Y31,Y32,X3A,Y3A
      INTEGER*2 X4,Y4,C4,X41,X42,X43,Y41,Y42,X4A,Y4A
      INTEGER VMIN4,VMAX4
      XMAX=30
      YMAX=30
      VID1=3
      VID2=60
      VID3=30
      VMIN4=5
      VMAX4=50
       VID8=300
      MIDX2=-1
      MADX2=1
      MIDY2=-1
      MADY2=1
      NMAX2=8
      MIDX8=-1
      MADX8=1
      MIDY8=-1
      MADY8=1
      MIX8=8
      MAX8=23
      MIY8=8
      MAY8=23
      N1=0
      NUM1=0
      NUM8=4
      P8X(1)=20
      P8Y(1)=10
      P8X(2)=10
      P8Y(2)=10
      P8X(3)=20
      P8Y(3)=20
      P8X(4)=10
      P8Y(4)=20
      N8=0
```

```
      DT=1
      TESC=DT
      TMAX=300
      T=1
      NIM1=735
      NAL1=NAL(0,0,NIM1)
C
C **********************************************************************
C
C HACER UNO HAB(X,Y),REL(X,Y) Y MOV2(X,Y)
 1000 XHR1=1
      XHR2=XMAX
      YHR1=1
      YHR2=YMAX
       DO 1020 Y=YHR1,YHR2
        DO 1010 X=XHR1,XHR2
      HAB(X,Y)=1
      REL(X,Y)=1
        MOV2(X,Y)=0
 1010 CONTINUE
 1020 CONTINUE
C
C **********************************************************************
C
C UBICAR LOS 8´S
       DO 1810 I=1,NUM8
       X=P8X(I)
       Y=P8Y(I)
       HAB(X,Y)=8
       REL(X,Y)=VID8
 1810  CONTINUE
C
C **********************************************************************
C
C ADVANCE DEL PELOJ
 1800  CONTINUE
       DO 1720 Y=1,YMAX
        DO 1710 X=1,XMAX
      IF(HAB(X,Y).EQ.1) GO TO 1705
        IF(HAB(X,Y).EQ.0) GO TO 1703
        IF(REL(X,Y).NE.0)REL(X,Y)=REL(X,Y)-1
C DETECTAR SI HAY ELEMENTOS QUE DEBEN DECAER O 1´S QUE DEBEN MOVERSE
        IF(REL(X,Y).GT.0) GO TO 1705
        IF(HAB(X,Y).EQ.2) NUM1=NUM1+1
        IF(HAB(X,Y).EQ.3) NUM1=NUM1+1
        IF(HAB(X,Y).EQ.4) NUM1=NUM1+1
      HAB(X,Y)=1
      REL(X,Y)=1
        GO TO 1705
 1703   CONTINUE
```

```
C MOVIMIENTO DE LOS 1´S (MEDIANTE MOVIENTO DE LOS 0´S)
        XPF0=NAL(1,XMAX,0)
        YPF0=NAL(1,YMAX,0)
        IF(HAB(XPF0,YPF0).NE.1) GO TO 1710
        HAB(X,Y)=1
        REL(X,Y)=1
        IF(NUM1.GT.0) GO TO 1702
        HAB(XPF0,YPF0)=0
        REL(XPF0,YPF0)=0
        GO TO 1710
 1702   NUM1=NUM1-1
        GO TO 1710
 1705   CONTINUE
 1710  CONTINUE
 1720   CONTINUE
      NUM1A=NUM1
 1100 CONTINUE
C
C ***********************************************************************
C
C MOVIMIENTO DE LOS 8´S
 1200 CONTINUE
       DO 1820 I=1,NUM8
       X=P8X(I)
       Y=P8Y(I)
       NAX8=NAL(MIDX8,MADX8,0)
       NAY8=NAL(MIDY8,MADY8,0)
       XPF8=X+NAX8
       YPF8=Y+NAY8
       IF(HAB(XPF8,YPF8).NE.1) GO TO 1820
       IF((HAB(XPF8,Y).EQ.4).AND.(HAB(X,YPF8).EQ.4)) GO TO 1820
       IF((XPF8.LT.MIX8).OR.(XPF8.GT.MAX8)) GO TO 1820
       IF((YPF8.LT.MIY8).OR.(YPF8.GT.MAY8)) GO TO 1820
       HAB(X,Y)=1
       HAB(XPF8,YPF8)=8
       REL(XPF8,YPF8)=REL(X,Y)
       REL(X,Y)=1
       P8X(I)=XPF8
       P8Y(I)=YPF8
 1820  CONTINUE
C
C ***********************************************************************
C
 1821 CONTINUE
C RECORRER  HAB(X,Y)
      XH1=1
      XH2=XMAX
      YH1=1
      YH2=YMAX
       DO 1220 Y=YH1,YH2
```

```
          DO 1210 X=XH1,XH2
          IF(HAB(X,Y).EQ.0) GO TO 1210
          IF(HAB(X,Y).EQ.1) GO TO 1210
          IF(HAB(X,Y).EQ.2) GO TO 200
          IF(HAB(X,Y).EQ.3) GO TO 300
          IF(HAB(X,Y).EQ.4) GO TO 400
          IF(HAB(X,Y).EQ.8) GO TO 1
          GO TO 2990
C
C **********************************************************************
C
C ELEMENTO 8
 1    X8=X
      Y8=Y
C RECORRER EL ENTORNO DE ESTE 8
      D=1
      X81=X8-D
       X82=X8+D
       Y81=Y8-D
       Y82=Y8+D
        DO 20 Y8A=Y81,Y82
        X83=2
        IF((Y8A.EQ.(Y8-D)).OR.(Y8A.EQ.(Y8+D))) X83=1
         DO 10 X8A=X81,X82,X83
         IF(HAB(X8A,Y8A).EQ.0) GO TO 10
         IF(HAB(X8A,Y8A).EQ.1) GO TO 3
         GO TO 10
 3     IF((HAB(X8A,Y8).EQ.4).AND.(HAB(X8,Y8A).EQ.4)) GO TO 10
       GO TO 100
C
C **********************************************************************
C
C ELEMENTO 1
 100  X1=X8A
      Y1=Y8A
      IF(D.NE.1) GO TO 10
C RECORRER EL ENTORNO DE ESTE 1
      X11=X1-1
      X12=X1+1
      Y11=Y1-1
      Y12=Y1+1
       DO 120 Y1A=Y11,Y12
       X13=1
       IF(Y1A.EQ.Y1) X13=2
        DO 110 X1A=X11,X12,X13
        IF(HAB(X1A,Y1A).NE.1) GO TO 110
      IF((HAB(X1A,Y1).EQ.4).AND.(HAB(X1,Y1A).EQ.4)) GO TO 110
        HAB(X1A,Y1A)=2
        REL(X1A,Y1A)=VID2
        HAB(X1,Y1)=0
```

```
          REL(X1,Y1)=0
          GO TO 10
 110      CONTINUE
 120     CONTINUE
 10        CONTINUE
 20       CONTINUE
        GO TO 1210
C
C ********************************************************************
C
C ELEMENTO 2
 200  X2=X
      Y2=Y
C TRANSFORMACION DE ESTE 2
 208  C2=0
      C23=0
      C24=0
C RECORRER EL ENTORNO DE ESTE 2
      X21=X2-1
      X22=X2+1
      Y21=Y2-1
      Y22=Y2+1
       DO 220 Y2A=Y21,Y22
       X23=1
       IF(Y2A.EQ.Y2) X23=2
        DO 210 X2A=X21,X22,X23
      IF(HAB(X2A,Y2A).EQ.2) C2=C2+1
      IF(HAB(X2A,Y2A).EQ.3) C23=1
      IF(HAB(X2A,Y2A).EQ.4) C24=C24+1
 210    CONTINUE
 220   CONTINUE
C SI NO HAY MAS DE UN 4 Y HAY UN 2 O UNO O MAS 3´S, ESTE 2 SE TRANSFORMA
      IF(.NOT.((C24.LE.1).AND.((C2.EQ.1).OR.(C23.EQ.1)))) GO TO 221
        HAB(X2,Y2)=3
        REL(X2,Y2)=VID3
      GO TO 1210
C MOVIMIENTO DE ESTE 2
 221  N2=0
      IF(MOV2(X2,Y2).NE.1) GO TO 223
      MOV2(X2,Y2)=0
      GO TO 1210
 223  NAX2=NAL(MIDX2,MADX2,0)
      NAY2=NAL(MIDY2,MADY2,0)
      XPF2=X2+NAX2
      YPF2=Y2+NAY2
      IF((HAB(XPF2,YPF2).EQ.4).OR.(HAB(XPF2,YPF2).EQ.3)) GO TO 223
      IF((XPF2.LT.2).OR.(XPF2.GT.(XMAX-1))) GO TO 1210
      IF((YPF2.LT.2).OR.(YPF2.GT.(YMAX-1))) GO TO 1210
      IF((HAB(XPF2,Y2).EQ.4).AND.(HAB(X2,YPF2).EQ.4)) GO TO 205
      IF(HAB(XPF2,YPF2).EQ.1) GO TO 203
```

```
 205  N2=N2+1
      IF(N2.GE.NMAX2) GO TO 1210
      GO TO 223
 203  HAB(X2,Y2)=1
      HAB(XPF2,YPF2)=2
      REL(XPF2,YPF2)=REL(X2,Y2)
      REL(X2,Y2)=1
      IF((XPF2.GT.X2).OR.(YPF2.GT.Y2)) MOV2(XPF2,YPF2)=1
      GO TO 1210
C
C ***********************************************************************
C
C ELEMENTO 3
 300  X3=X
      Y3=Y
C RECORRER EL ENTORNO DE ESTE 3
C MOLINETE SINISTROGIRO
      X31=X3+1
      Y31=Y3+1
      C3=0
      MAR3=0
      IF((HAB(X31,Y31).EQ.3).OR.(HAB(X31,Y31).EQ.4)) MAR3=1
 305  CONTINUE
      X3A=X31
      Y3A=Y31
      IF((X31.GE.X3).AND.(Y31.LT.Y3)) X3A=X31-1
      IF((X31.LE.X3).AND.(Y31.GT.Y3)) X3A=X31+1
      IF((Y31.LE.Y3).AND.(X31.LT.X3)) Y3A=Y31+1
      IF((Y31.GE.Y3).AND.(X31.GT.X3)) Y3A=Y31-1
      X31=X3A
      Y31=Y3A
      IF((.NOT.((HAB(X31,Y31).EQ.3).OR.(HAB(X31,Y31).EQ.4)))
     2.AND.(MAR3.EQ.1)) C3=C3+1
      IF(((HAB(X31,Y31).EQ.3).OR.(HAB(X31,Y31).EQ.4)).AND.(C3.EQ.0))
     2 MAR3=1
      IF(((HAB(X31,Y31).EQ.3).OR.(HAB(X31,Y31).EQ.4)).AND.(C3.GT.0))
     2 MAR3=0
      IF(.NOT.((C3.GT.0).AND.(C3.LT.6).AND.(MAR3.EQ.0))) GO TO 307
C TRANSFORMACION DE ESTE 3 EN 4
      HAB(X3,Y3)=4
      REL(X3,Y3)=NAL(VMIN4,VMAX4,0)
      GO TO 1210
 307  CONTINUE
      IF((X31.EQ.X3).AND.(Y31.EQ.(Y3+1))) GO TO 1210
      GO TO 305
C
C ***********************************************************************
C
C ELEMENTO 4
 400  X4=X
```

```
      Y4=Y
C RECORRER EL ENTORNO DE ESTE 4
C MOLINETE SINISTROGIRO
      X41=X4+1
      Y41=Y4+1
      C4=0
      MAR4=0
      IF((HAB(X41,Y41).EQ.4).OR.(HAB(X41,Y41).EQ.3)) MAR4=1
 405  CONTINUE
      X4A=X41
      Y4A=Y41
      IF((X41.GE.X4).AND.(Y41.LT.Y4)) X4A=X41-1
      IF((X41.LE.X4).AND.(Y41.GT.Y4)) X4A=X41+1
      IF((Y41.LE.Y4).AND.(X41.LT.X4)) Y4A=Y41+1
      IF((Y41.GE.Y4).AND.(X41.GT.X4)) Y4A=Y41-1
      X41=X4A
      Y41=Y4A
      IF((.NOT.((HAB(X41,Y41).EQ.3).OR.(HAB(X41,Y41).EQ.4)))
     2.AND.(MAR4.EQ.1)) C4=C4+1
      IF(((HAB(X41,Y41).EQ.3).OR.(HAB(X41,Y41).EQ.4)).AND.(C4.EQ.0))
     2 MAR4=1
      IF(((HAB(X41,Y41).EQ.3).OR.(HAB(X41,Y41).EQ.4)).AND.(C4.GT.0))
     2 MAR4=0
      IF((C4.GT.0).AND.(C4.LT.6).AND.(MAR4.EQ.0)) GO TO 1210
      IF((X41.EQ.X4).AND.(Y41.EQ.(Y4+1)))GO TO 408
      GO TO 405
 408  CONTINUE
C TRANSFORMACION DE ESTE 4 EN 3
      HAB(X4,Y4)=3
      REL(X4,Y4)=VID3
      GO TO 1210
C
C ***********************************************************************
C
 1210   CONTINUE
 1220  CONTINUE
C
C ***********************************************************************
C
C ESCRIBIR HAB(X,Y)
 2000 CONTINUE
      IF(TESC.LT.DT) GO TO 2100
      TESC=0
      WRITE(3,2052) NUM1A,T
 2052 FORMAT('1',///,15X,'NUM1=',I5,45X,'T=',I5,///)
       DO 2020 Y=1,YMAX
        DO 2010 X=1,XMAX
        NN=HAB(X,Y)
        IF(NN.NE.0) GO TO 2005
        ESC(X,Y)=B(7)
```

Dec 16 12:13 1996 exp29.for Page 8

```
           GO TO 2010
  2005     ESC(X,Y)=B(NN)
  2010     CONTINUE
  2020    CONTINUE
         WRITE(3,2092) ((ESC(X,Y),X=1,XMAX),Y=1,YMAX)
  2092   FORMAT(´ ´,50X,30A1)
C
C ********************************************************************
C
C   AVANCE DEL TIEMPO
  2100 IF(T.GE.TMAX) GO TO 3000
       T=T+1
       TESC=TESC+1
       GO TO 1800
  2990 WRITE(3,2992)
  2992 FORMAT(´0´,///,3X,´ELEMENTO DESCONOCIDO´,//)
  3000 STOP
       END
```

Nov 14 16:15 1996 nal.for Page 1

```
       INTEGER FUNCTION NAL (LO, HI, SALT)
       INTEGER LO, HI, SALT
       INTEGER SEED
       IF (SALT.NE.0) SEED=SALT
       NAL = LO + INT((RAN(SEED) * FLOAT(HI-LO)) + 0.5)
       RETURN
       END
```