

Search for Life in Catalytic Reaction Systems

Ina Trolle Andersen, Lin Nan
&
Maiken Ina Siegismund Kjærsgaard

August 13, 2010

Abstract

The ability of systems of molecular reactions to be simultaneously autocatalytic and sustained by some food source of simple molecules may have been an essential step in the origin of life. In this report we will continue the work by Hordijk and Steel (2004) and Mossel and Steel (2005) of investigating the probability of autocatalytic sets arising. We first describe an alternative to the *RAF* algorithm in Hordijk and Steel (2004) that determines whether a system of molecules, reactions and catalysations contains a subsystem that is autocatalytic and able to be sustained from a given subset of molecules. We use the algorithm to investigate random catalytic networks. In particular two new models, Template based model and Thermodynamic model, will be described and investigated both analytically and numerically—it will be investigated if the rate of catalyses needed for the existence of autocatalytic and self-sustaining subsystems is realistic from a biological point of view.

Contents

1	Introduction	1
1.1	Catalytic reaction systems	1
1.1.1	Definitions	2
1.1.2	Example	2
1.2	Autocatalytic sets generated by a food source	3
1.2.1	Definitions	3
1.2.2	Example	3
2	The <i>RAF</i> algorithm	5
2.1	The existence of an <i>RAF</i> set	5
2.2	The <i>RAF</i> algorithm by Hordijk and Steel	5
2.3	A new algorithm based on reference counting	6
2.4	A more efficient way of handling the reactions	9
2.5	A more efficient way of handling the closure of <i>F</i>	9
2.6	Example	10
2.7	The average running time of our implementations	12
3	Random catalytic reaction systems	15
3.1	The Kauffman model	15
3.2	Random sequence based model	18
3.3	Template based model	19
3.4	Thermodynamic models	21
4	Simulation	23
4.1	The Java-program	23
4.1.1	The set of molecules and the food set	24
4.1.2	The set of reactions	24
4.1.3	The set of catalysations	24
4.2	Results	26
4.2.1	Simulation 1	26
4.2.2	Simulation 2 and Simulation 3	27
4.2.3	Simulation 4	31
5	Extended results in the Template based model	33
5.1	Notations	33
5.2	Generalisation into <i>k</i>	35
6	Conclusions and further work	36
	Appendix	38
	Implementation 1	38
	Implementation 2	40
	Implementation 3	42

1 Introduction

The origin of life remains an elusive problem. DNA requires catalysts (proteins) to be replicated and expressed, and proteins require DNA to be encoded and formed. Two main paradigms for this 'chicken and egg' problem currently exist: the RNA-first theory and the protein-first theory. In the RNA-first theory it is assumed that life started with self-reproducing RNA molecules using template complementarity, which later evolved to encode genetic information translated into proteins. The problem with this theory is, however, that RNA replication is difficult without catalysts and RNA has only limited catalytic capability. In the protein-first theory it is assumed that life started as self-reproducing sets of catalytic proteins. Proteins form easily from freely available amino acids and are highly catalytic. The problem with this theory is, however, that proteins do not have a template structure, and thus need some sort of genetic code. Another theory, which allows both scenarios, is that of autocatalytic sets. Despite differences between various proposed scenarios, the ability of systems of molecular reactions to be simultaneously autocatalytic and sustained by some food source of simple molecules seems to be a common element. It has been suggested that in sufficiently complex chemical reaction systems an autocatalytic set will emerge spontaneously, but this gives rise to the question: how complex is 'sufficiently complex'?

In this report we will continue the work by Hordijk and Steel (2004) and Mossel and Steel (2005) of investigating the probability of autocatalytic sets arising in chemical reaction systems. This section formally introduces catalytic reaction systems and autocatalytic sets generated by a food source (called *RAF* sets). Section 2 introduces the polynomial-time algorithm, which determines if a catalytic reaction system contains an *RAF* set, from Hordijk and Steel (2004), and aims towards an improvement of this algorithmic framework. The section presents a new algorithm for finding *RAF* sets and shows results for the running time. Section 3 introduces two already known models, the Kauffman model and the Random sequence based model, and two natural extensions of those, which are called the Template based model and the Thermodynamic model, are explored. Section 4 presents simulations and the results of those for the four models. Section 5 provides an extension of the results obtained for the Random sequence based model on upper and lower bounds for the probability of autocatalytic sets arising in chemical reaction systems given in Mossel and Steel (2005) in the case of the Template based model, and in Section 6 we make some concluding comments, and raise some questions for future investigation.

1.1 Catalytic reaction systems

The introduction of formal models in the context of the origin of life is gaining pace. Formal models should attempt to capture some essence of the empirical problem and with time give a more realistic description of the phenomena. The formalization of catalytic reaction systems or CRS by Steel (2000) consists of

- a set of molecules
- a set of reactions where each reaction converts one set of molecules (called the reactants) into another set (called the products)

- a set of catalysations, that is molecules that accelerate a reaction or a set of reactions
- a food set which is a small set of molecules assumed to be freely available and constantly replenished

In this section, we review the notion of catalytic reaction systems and reaction graphs. We mostly follow the notation of Hordijk and Steel (2004).

1.1.1 Definitions

Let X denote a set of molecules and \mathcal{R} a set of reactions, where a reaction is an ordered pair $r = (A, B)$ where A and B are subsets of X called the reactants and products, respectively. Let F denote a distinguished subset of X . For $r \in \mathcal{R}$ let $\rho(r) = A$ and $\pi(r) = B$ and for a set $\mathcal{R}' \subseteq \mathcal{R}$ let

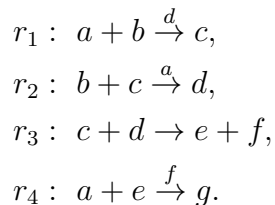
$$\begin{aligned}\rho(\mathcal{R}') &:= \cup_{r \in \mathcal{R}'} \rho(r) \\ \pi(\mathcal{R}') &:= \cup_{r \in \mathcal{R}'} \pi(r) \\ \text{supp}(\mathcal{R}') &:= \rho(\mathcal{R}') \cup \pi(\mathcal{R}'),\end{aligned}$$

i.e. $\text{supp}(\mathcal{R}')$ denotes the molecules in X that are used or produced by at least one reaction in \mathcal{R}' . A catalysation is a pair (x, r) where $x \in X$ and $r \in \mathcal{R}$ indicating that molecule x catalyses reaction r . Let $C \subseteq X \times \mathcal{R}$ be a set of catalysations. The triple $\mathcal{L} = (X, \mathcal{R}, C)$ is called a catalytic reaction system over F or CRS over F .

A catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$ may be presented as a directed graph on vertex set $X \cup \mathcal{R}$, and with two types of arcs, described as follows. For each $r = (A, B) \in \mathcal{R}$, we place an arc from each element $a \in A$ to r and from r to each element $b \in B$, and we will refer to these as reaction arcs. Also, we place an arc from $x \in X$ to any $r \in \mathcal{R}$ when $(x, r) \in C$, and we will refer to these as catalytic arcs.

1.1.2 Example

As an example of a catalytic reaction system we will consider the following four reactions:



Let the food set be $F = \{a, b\}$. Then $\mathcal{L} = (X, \mathcal{R}, C)$ given by $X = \{a, b, c, d, e, f, g\}$, $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ and $C = \{(d, r_1), (a, r_2), (f, r_4)\}$ is a catalytic reaction system over F . This CRS can be represented by the catalytic reaction graph shown in Figure 1. In this graph molecules (the set X) are represented as solid nodes, reactions (the set \mathcal{R}) are represented as open nodes, solid arrows represent reaction arcs and dashed arrows represent catalytic arcs.

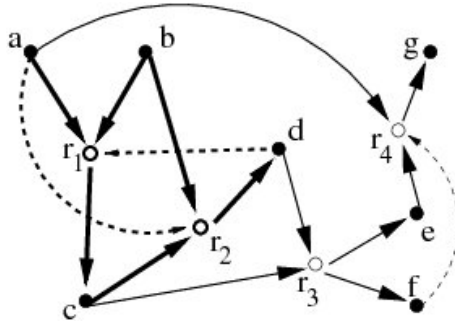


Figure 1: An example of a catalytic reaction system (Hordijk and Steel (2004), Fig. 1).

1.2 Autocatalytic sets generated by a food source

The question of interest in a catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$ are conditions for the appearance of a subset \mathcal{R}' of the reactions \mathcal{R} that are self-sustainable, i.e. every reaction in the subset \mathcal{R}' is catalysed by at least one molecule involved in a reaction in \mathcal{R}' , and every molecule can be constructed from the food set F by successive application of reactions from \mathcal{R}' . The idea of autocatalytic sets was introduced by Kauffman (1986), and formalized as *RAF* sets and studied more extensively by Steel (2000), Hordijk and Steel (2004) and Mossel and Steel (2005). In this section, we review the notion of reflexively autocatalytic and F -generated sets. Again, we mostly follow the notation of Hordijk and Steel (2004).

1.2.1 Definitions

Given a subset \mathcal{R}' of \mathcal{R} and a subset X' of X , the closure of X' relative to \mathcal{R}' , denoted $cl_{\mathcal{R}'}(X')$, is defined as the (unique) minimal subset W of X that contains X' and that satisfies the condition that, for each reaction $r = (A, B) \in \mathcal{R}'$,

$$A \subseteq X' \cup W \Rightarrow B \subseteq W,$$

i.e. $cl_{\mathcal{R}'}(X')$ is X' together with all the molecules that can be constructed from X' by repeated application of reactions in \mathcal{R}' .

Suppose we are given a catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$ and a subset F of X . A nonempty subset \mathcal{R}' of \mathcal{R} is said to be:

- reflexively autocatalytic (RA) for \mathcal{L} if for all $r \in \mathcal{R}'$ there exists an $x \in \text{supp}(\mathcal{R}') : (x, r) \in C$,
- F -generated if $\rho(\mathcal{R}') \subseteq cl_{\mathcal{R}'}(F)$,
- reflexively autocatalytic and F -generated (*RAF*) for \mathcal{L} if \mathcal{R}' is RA for \mathcal{L} and generated by F . We will then refer to \mathcal{R}' as an *RAF* set for \mathcal{L} .

1.2.2 Example

As an example the subset $\mathcal{R}' = \{r_1, r_2\}$ from section 1.1.2 forms an *RAF* set, and is shown in Figure 2. Each reaction in \mathcal{R}' is catalysed by a molecule in $\text{supp}(\mathcal{R}')$

and all reactants can be constructed from F by one or more applications of r_1 and r_2 : (1) the molecules involved in \mathcal{R}' are $\{a, b, c, d\}$, reaction r_1 is catalysed by d and r_2 is catalysed by a , so \mathcal{R}' is RA, (2) the reactants in \mathcal{R}' are either already in the food set (a and b) or can be constructed from it through r_1 and r_2 (c and d), so \mathcal{R}' is also F -generated.

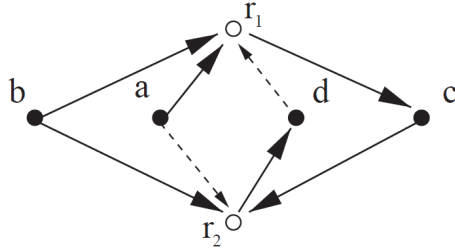


Figure 2: An example of an *RAF* set with two reactions $\{r_1, r_2\}$ for $F = \{a, b\}$ (Hordijk and Steel (2004), Fig. 2).

2 The *RAF* algorithm

2.1 The existence of an *RAF* set

In this section we introduce some notation and state one of the main results for *RAF* sets. The theorem will be given without proof.

The following reduction rules can be applied to any catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$.

(R1) For any $r \in \mathcal{R}$ with $\{x \in \text{supp}(\mathcal{R}) : (x, r) \in C\} = \emptyset$, delete r from \mathcal{R} .

(R2) For any $r \in \mathcal{R}$ for which $\rho(r) \not\subseteq \text{cl}_{\mathcal{R}}(F)$, delete r from \mathcal{R} .

Let $\gamma(\mathcal{R}) \subseteq \mathcal{R}$ and $\delta(\mathcal{R}) \subseteq \mathcal{R}$ denote a set of reactions obtained by repeated applications of (R1), respectively (R2), to $\mathcal{L} = (X, \mathcal{R}, C)$ until no further reductions can be made. Using the operations γ and δ it is now possible to construct a decreasing (in size) and nested sequence $\mathcal{R} = \mathcal{R}_1, \mathcal{R}_2, \dots$ of subsets of \mathcal{R} where $\mathcal{R}_{i+1} = \delta(\gamma(\mathcal{R}_i))$. Let $\mathcal{R}_{\infty} = \bigcap_{i \geq 1} \mathcal{R}_i$.

Given a catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$, and a subset \mathcal{R}' of \mathcal{R} , we say that \mathcal{R}' is an irreducible *RAF* set for \mathcal{L} if \mathcal{R}' is an *RAF* set for \mathcal{L} , but no proper subset of \mathcal{R}' has this property. We have the following important result stated as Theorem 6.1 in Hordijk and Steel (2004).

Theorem 2.1. *A catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$ has an *RAF* set if and only if \mathcal{R}_{∞} is non-empty, in which case \mathcal{R}_{∞} is the maximal *RAF* set for \mathcal{L} . Furthermore, there exists a polynomial-time algorithm for computing \mathcal{R}_{∞} and, when this is non-empty, finding an irreducible *RAF* set for \mathcal{L} .*

2.2 The *RAF* algorithm by Hordijk and Steel

In Hordijk and Steel (2004) a polynomial-time algorithm for computing \mathcal{R}_{∞} was presented, i.e. the algorithm determines if a catalytic reaction system contains an *RAF* set. The algorithm is based on two observations; (1) for a set of reactions \mathcal{R}' removing a reaction $r \in \mathcal{R}'$ that is not catalysed by an element $x \in \text{supp}(\mathcal{R}')$ will not change the maximal RA of the system, (2) removing a molecule $x \in X$ that cannot be generated from the food set will not change the closure of the food set relative to \mathcal{R}' . So we can iteratively remove such reactions and molecules until no more exist, and this will generate the maximal *RAF* set.

We now assume the existence of a catalytic reaction system $\mathcal{L} = (X, \mathcal{R}, C)$, a food set F and a set W representing the closure of the food set relative to \mathcal{R} . The *RAF* algorithm by Hordijk and Steel (2004) consists of three separate subroutines that are called sequentially and repeatedly. It assumes X , C and F to be fixed and will change the contents of \mathcal{R} and W . The first subroutine implements the $\gamma(\mathcal{R})$ function, that is it applies the reduction rule (R1) until no more reductions can be made, and thereby reduces a given reaction set \mathcal{R} to an RA set. The second subroutine computes the closure of F relative to the current set of reactions \mathcal{R} . Finally, the third subroutine implements the $\delta(\mathcal{R})$ function, that is it applies the reduction rule (R2) until no more reductions can be made, and thereby reduces a given reaction set \mathcal{R} to a subset that is F -generated.

The analysis of the algorithm yields a worst case complexity of $\mathcal{O}(|X||\mathcal{R}|^3)$. However, the algorithm essentially starts from scratch in each iteration when identifying reactions and molecules that should be removed; reactions are removed when there are no more molecules catalysing them and molecules are removed when there are no more reactions producing them. It is therefore reasonable to assume that an algorithm based on reference counting can improve the running time of the algorithm. This would then significantly expand the size of systems that can be analysed. Reference counting tracks for each object a count of the number of references to it held by other objects. If an object’s reference count reaches zero (or possibly some other number, depending on the context) the object is disregarded, and if this happens all object’s referenced by that object also have their reference counts decreased and so on. The task now is to come up with a pseudocode for an improved algorithm (in terms of running time) based on reference counting where we will keep track of the support of the current set of reactions \mathcal{R}' and the closure F relative to the current set of reactions \mathcal{R}' .

2.3 A new algorithm based on reference counting

The pseudocode is given below in Algorithm 1. The implementation can be found as Implementation 1 in Appendix.

Algorithm 1

```

/* Initialization */
 $\mathcal{R}' = \mathcal{R}, \forall x \in X : \mathbf{s}[x] = 0, \forall r \in \mathcal{R}' : \mathbf{c}[r] = 0, D = \emptyset$ 
for all  $r \in \mathcal{R}'$  do
  for all  $x \in \rho(r) \cup \pi(r)$  do
     $\mathbf{s}[x] = \mathbf{s}[x] + 1$  /*  $\mathbf{s}[x]$  is the number of reactions involving  $x$  */
  end for
end for
for all  $c = (x, r) \in C$  do
  if  $\mathbf{s}[x] > 0$  then
     $\mathbf{c}[r] = \mathbf{c}[r] + 1$  /*  $\mathbf{c}[r]$  is the number of compounds catalysing  $r$  */
  end if
end for
for all  $r \in \mathcal{R}'$  do
  if  $\mathbf{c}[r] = 0$  then
     $D = D \cup r$  /*  $D$  is the set of uncatalysed reactions */
  end if
end for
repeat
  /* Reduce to RA */
  while  $D \neq \emptyset$  do
     $D = D \setminus r, \mathcal{R}' = \mathcal{R}' \setminus r$ 
    for all  $x \in \rho(r) \cup \pi(r)$  do
       $\mathbf{s}[x] = \mathbf{s}[x] - 1$ 
      if  $\mathbf{s}[x] = 0$  then

```

```

    for all  $c = (x, s) \in C$  do
      if  $s \notin \mathcal{R}'$  then
         $C = C \setminus c$ 
      else
         $\mathbf{c}[s] = \mathbf{c}[s] - 1$ 
        if  $\mathbf{c}[s] = 0$  then
           $D = D \cup s$ 
        end if
      end if
    end for
  end if
end for
end while
/* Reduce to  $F$ -generated */
 $N = F, \forall r \in \mathcal{R}' : \mathbf{r}[r] = 0, \mathcal{S} = \emptyset$ 
while  $N \neq \emptyset$  do
   $N = N \setminus x$ 
  for all  $r \in \mathcal{R}'$  with  $x \in \rho(r)$  do
     $\mathbf{r}[r] = \mathbf{r}[r] + 1$ 
    if  $\mathbf{r}[r] = |\rho(r)|$  then
       $\mathcal{S} = \mathcal{S} \cup r$ 
      for all  $y \in \pi(r)$  with  $y$  not previously produced do
         $N = N \cup y$ 
      end for
    end if
  end for
end while
 $D = \mathcal{R}' \setminus \mathcal{S}$ 
until  $D = \emptyset$ 

```

In the implementation of the algorithm we have chosen our data structure to be a list of strings for the molecules, two lists of lists of integers for the reactions corresponding to a list for each reaction containing the molecules that are reactants for that particular reaction and a list for each reaction containing the molecules that are products for that particular reaction, respectively, and a list of lists of integers for the catalysations corresponding to a list for each molecule containing the reactions catalysed by that particular molecule. The assumptions that we are working under when analysing the running time are that the number of products and the number of reactants for a given reaction and the food set F are all of some constant size.

In the **repeat** loop the algorithm first reduces the set \mathcal{R} to an RA set, and then reduces \mathcal{R} even further so that only reactions remain of which the reactants can be generated from the food set F . However, at this point we cannot be sure that \mathcal{R} is still an RA set, so the **while** loops and the update of D at the end of the **repeat** loop are executed in sequence again, and again, until either no reactions

are left or no more reductions can be made. It follows that the total number of executions is $\mathcal{O}(|\mathcal{R}|)$. However, we can also observe that the set of molecules that are F -generated has to be strictly decreasing for each iteration of the **repeat** loop. Hence the number of executions of the **repeat** loop is also $\mathcal{O}(|X|)$. Furthermore, it does seem reasonable from a biological point of view to assume that the number of molecules is less than the number of reactions. We will use these observations in the following.

We will start by taking a look at the first **while** loop. Combining the **repeat** loop with the first **while** loop yields an overall running time of the first **while** loop of $\mathcal{O}(|C| |\mathcal{R}|)$. The number of executions of the **repeat** loop and the first **while** loop is bounded by $|\mathcal{R}|$ since at least one reaction will be removed in each iteration. But since the first **while** loop is inside the **repeat** loop many iterations of the **while** loop implies few iterations of the **repeat** loop and vice versa. So in each iteration of the **while** loop we remove exactly one reaction, and this can then happen at most $|\mathcal{R}|$ times. Removing a reaction from D and \mathcal{R}' takes constant time in our implementation, so the first line will take time at most $\mathcal{O}(|\mathcal{R}|)$. Because of the assumption, that for a given reaction the number of reactants and the number of products are all of some constant size, going through the list of reactants and the list of products in line two takes constant time in our implementation, so the first two lines will take time at most $\mathcal{O}(|\mathcal{R}|)$. The update of the array \mathbf{s} and checking for zero-entries takes constant time, and so the really interesting part is when going through the set of catalysations C in line five. The important thing to notice at this point is, that $\mathbf{s}[x] = 0$ will happen at most once for each molecule x , so every time we reach this part we will go through a specific list corresponding to the list of reactions catalysed by a particular molecule x , which we will therefore only have to run through once, and this can happen at most $|C|$ times. The update of the array \mathbf{c} , checking for zero-entries and updating D takes constant time in our implementation, but since it might happen that all the molecules catalyses all the reactions we may have to remove a catalyses from an array list containing the catalysations for each reaction and the worst case complexity of the innermost **for** loop will therefore be $\mathcal{O}(|C| |\mathcal{R}|)$, which is also the overall running of the first **while** loop.

Next we take a look at the second **while** loop. The second **while** loop is easily analyzed and details will therefore be skipped. The outer **for** loop can be done in $\mathcal{O}(|\mathcal{R}|)$ time, so the **while** loop can be done in $\mathcal{O}(|X| |\mathcal{R}|)$ time.

Therefore the overall running time for executing the repeat loop is $\mathcal{O}(|C| |\mathcal{R}|) + \mathcal{O}(|X|^2 |\mathcal{R}|) + \mathcal{O}(|X| |\mathcal{R}|) = \mathcal{O}(|C| |\mathcal{R}|) + \mathcal{O}(|X|^2 |\mathcal{R}|)$, and since $|C|$ is bounded by $|X| |\mathcal{R}|$ and $|X|$ is less than $|\mathcal{R}|$ our analysis yields an overall running time of $\mathcal{O}(|X| |\mathcal{R}|^2) + \mathcal{O}(|X|^2 |\mathcal{R}|) = \mathcal{O}(|X| |\mathcal{R}|^2)$ which is significantly better than $\mathcal{O}(|X| |\mathcal{R}|^3)$ obtained by the *RAF* algorithm in Hordijk and Steel (2004).

Next step is to implement a more efficient way of handling the reactions. The first implementation of the algorithm above assumes the representation of the reactions to be a boolean array returning true if the reaction is not a part of an *RAF* set and false if it might be, and when the algorithm terminates it will contain exactly the reactions that make up an *RAF* set (if all reactions return true no *RAF* set exists). So when going through the reactions we are not only looking at reactions that still might be part of an *RAF* set, but also reactions that are already disregarded, and

it is not advantageous to spend time on these reactions.

Also, in each iteration the closure of F is computed from scratch, building it from F using the reactions remaining, so introducing a count of the number of reactions remaining that produce a compound one could attempt the reverse approach of peeling off parts that are no longer F -generated. The complicating factor is that some parts that are no longer F -generated would still have a non-zero count due to cyclic structures. In particular, using reference counting on the closure of F , i.e. counting how many reactions still generate a compound from F , we would only have to recompute the closure of F whenever all the compounds no longer F -generated are all part of cyclic structures causing a non-zero reference count. So we still need to recompute the closure of F from time to time, but the repeat statement would run fewer times. These improvements will not make the worst case complexity any better, but it would definitely improve the average running time.

2.4 A more efficient way of handling the reactions

The implementation can be found as Implementation 2 in Appendix. First a change of list type for the set of catalysations C improved on the overall running time of the first **while** loop. When changing from array list to linked list for the set of catalysations removing elements can be done in constant time, so instead of having an overall running time of $\mathcal{O}(|C||\mathcal{R}|)$ for the first **while** loop we now have $\mathcal{O}(|C| + |\mathcal{R}|)$. What we would like to improve on now is the second **while** loop since the second **while** loop takes at least the same amount of time as the first **while** loop. We want a more efficient way of handling the reactions. Fortunately the choice of data structure made it quite easy to rewrite some of the input into a more useful (in some situations) representation and thereby an improvement of the running time was possible. Instead of having a list of lists of integers for reactions corresponding to a list for each reaction containing the molecules that are reactants for that particular reaction we would like to have a list for each molecule containing the reactions in which that particular molecule is a reactant. This representation of data allows us to access only the reactions of interest, and we will not have to go through all reactions in the outer **for** loop in the second **while** loop as it was the case in the first implementation. So instead of having an overall running time of $\mathcal{O}(|X|^2|\mathcal{R}|)$ for the second **while** loop we now have $\mathcal{O}(|X|(|X| + |\mathcal{R}|)) = \mathcal{O}(|X||\mathcal{R}|)$. Again, since $|C|$ is bounded by $|X||\mathcal{R}|$ our analysis yields an overall running time for executing the repeat loop of $\mathcal{O}(|X||\mathcal{R}|) + \mathcal{O}(|X||\mathcal{R}|) + \mathcal{O}(|X||\mathcal{R}|) = \mathcal{O}(|X||\mathcal{R}|)$, which is better than $\mathcal{O}(|X||\mathcal{R}|^2)$, obtained in the first implementation, by a factor $|\mathcal{R}|$.

2.5 A more efficient way of handling the closure of F

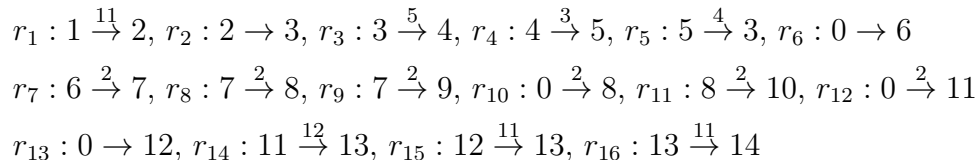
The implementation can be found as Implementation 3 in Appendix. In the implementation we sort of combine the 'reduce to RA' part and the 'reduce to F -generated' part. First we introduce a count of the number of reactions remaining that produce a compound given by an array \mathbf{t} . When removing an uncatalysed reaction we now count down $\mathbf{t}[x]$ for the molecules being produced by the reaction, and when the count reaches zero that particular molecule is no longer produced by any

reaction. Next we add all the reactions missing a reactant, i.e. reactions for which the count of one of the reactants has reached zero, to the set of reactions that should be removed. As already mentioned a complicating factor is that some parts that are no longer F -generated would still have a non-zero count due to cyclic structures. As an example of a cyclic structure see the example in Section 2.6. Figure 3 shows a catalytic reaction system, and when taking a look at the blue part of the catalytic reaction graph we recognize a cycle consisting of the molecules $\{3, 4, 5\}$ since after removing reaction r_2 , that is not catalysed, the molecules $\{3, 4, 5\}$ are still produced and catalysed by one another but they cannot be produced from the food set. The consequence of this is that we still have to recompute the closure of F whenever all the compounds no longer F -generated are all part of cyclic structures causing non-zero count, but hopefully one might have fewer iterations of the **repeat** loop. As already mentioned this will not make the worst case complexity $\mathcal{O}(|X| |\mathcal{R}|)$, obtained in the second implementation, any better, but we expect it to improve the average running time.

2.6 Example

In this section we will start by introducing an example illustrating several scenarios of possible graph structures that are taken into account in the implementation of the algorithm, and thereby give a better understanding of how the algorithm works.

As an example of a catalytic reaction system we will consider the following sixteen reactions:



Let the food set be $F = \{0, 1\}$. Then $\mathcal{L} = (X, \mathcal{R}, C)$ given by $X = \{0, 1, 2, \dots, 14\}$, $\mathcal{R} = \{r_1, r_2, r_3, \dots, r_{16}\}$ and $C = \{(11, r_1), (5, r_3), (3, r_4), \dots, (11, r_{16})\}$ is a catalytic reaction system over F . This CRS can be represented by the catalytic reaction graph shown in Figure 3. In this graph molecules (the set X) are represented as solid nodes, reactions (the set \mathcal{R}) are represented as open nodes, solid arrows represent reaction arcs and dashed arrows represent catalytic arcs. In the implementation of the algorithm we have as already mentioned chosen our data structure to be a list of strings for the molecules, two lists of lists of integers for the reactions corresponding to a list for each reaction containing the molecules that are reactants for that particular reaction and a list for each reaction containing the molecules that are products for that particular reaction, respectively, and a list of lists of integers for the catalysations corresponding to a list for each molecule containing the reactions

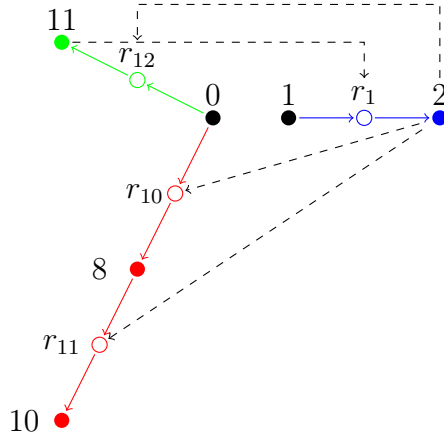


Figure 4: An example of an *RAF* set with four reactions $\{r_1, r_{10}, r_{11}, r_{12}\}$ for $F = \{0, 1\}$.

molecules involved in \mathcal{R}' are $\{0, 1, 2, 8, 10, 11\}$, reaction r_1 is catalysed by 11 and reactions r_{10} , r_{11} and r_{12} are catalysed by 2, so \mathcal{R}' is RA, (2) the reactants in \mathcal{R}' are either already in the food set (0 and 1) or can be constructed from it through r_1 , r_{10} , r_{11} and r_{12} (2, 8, 10 and 11), so \mathcal{R}' is also F -generated. In this example our algorithm would return the lists:

```

molecules = {0, 1, 2, 8, 10, 11}
reactants = {{1}, {0}, {3}, {0}}
products = {{2}, {3}, {4}, {5}}
catReactions = {∅, ∅, {1, 2, 3}, ∅, ∅, {0}}
foodset = {0, 1}

```

2.7 The average running time of our implementations

In this section we will investigate the average running time of the three implementations of our algorithm on Kauffman's model, and compare them with the average running time obtained using the *RAF* algorithm in Hordijk and Steel (2004).

The average running time of the three implementations is depicted in Figure 6. Implementations 1, 2 and 3 are depicted from the top and downwards. In each plot the black dots represent the actual running time of the implementation on Kauffman's model in CPU seconds for generating $p(n) |\mathcal{R}(n)|$ (averaged over 100 random instances) for values of $p(n)$ for which $Pr[RAF]$ is around 0.5 for $n = 5, 6, \dots, 10$ (see Section 3.1 for Kauffman's model). As the figure shows, the data in each case follows a straight line quite nicely.

The average running time of the implementation of the *RAF* algorithm from Hordijk and Steel (2004) is depicted in Figure 5. The solid line represents a fitted power law: $\tau = 0.0002 |\mathcal{R}|^{1.43}$, and the solid lines in the plots in Figure 6 represent

the fitted power laws:

$$\begin{aligned} \text{Implementation 1: } \tau &= 0.000000021 |\mathcal{R}|^{1.931} \\ \text{Implementation 2: } \tau &= 0.000000293 |\mathcal{R}|^{1.368} \\ \text{Implementation 3: } \tau &= 0.000000408 |\mathcal{R}|^{1.342} \end{aligned}$$

Our implementations have a sub-quadratic average running time with a very small constant, and the running times are getting better with each new implementation. It is worth mentioning that the third implementation has a better average running time than the average running time obtained by the implementation of the *RAF* algorithm from Hordijk and Steel (2004), and that the analysis of the theoretical running times in previous sections matches the empirical one.

When comparing the average running time for Implementation 2 and Implementation 3 applied on the Kauffman model there does not seem to be much of an improvement. If a catalytic reaction system contain many cycles we may have to compute the closure of F quite often, and so the running time of Implementation 3 will be something close to the running time of Implementation 2. Whether there are many cycles in a system will depend on the model being investigated, and so the last two implementations will roughly be equally good in the case with many cycles. However, if a system contains only few cycles the third implementation should be better than the second implementation.

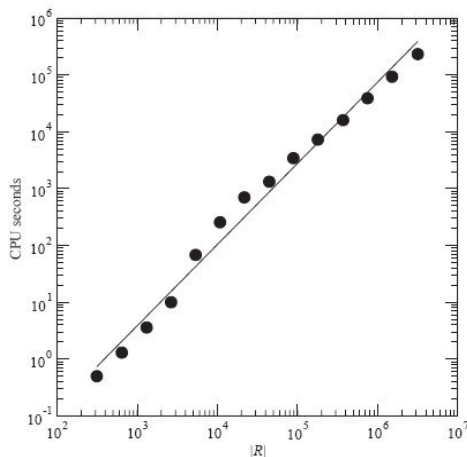


Figure 5: Actual running time of the implementation of the *RAF* algorithm from Hordijk and Steel (2004) on Kauffman’s model (Hordijk and Steel (2004), Fig. 7).

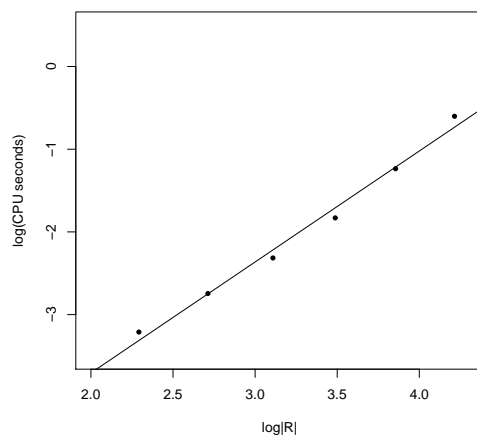
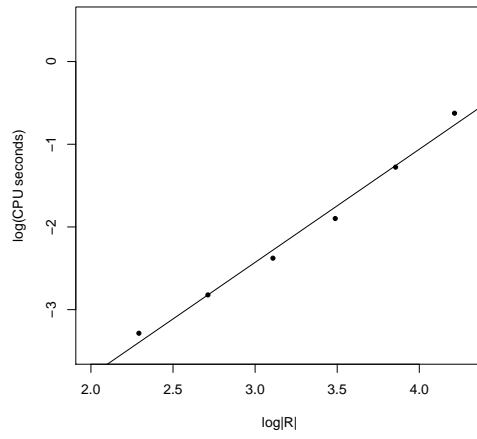
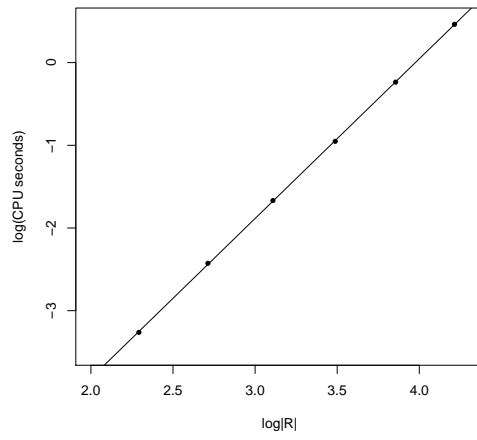


Figure 6: Actual running time of Implementation 1 (top), Implementation 2 (middle) and Implementation 3 (bottom) on Kauffman's model.

3 Random catalytic reaction systems

We will now consider catalytic reaction systems specified in different models. The first model is a model introduced by Kauffman. The other models are extensions of this model in an attempt to find more realistic models. One common thing for all these reaction systems is that they are made from a fixed molecule set, food set and reaction set, while the set of catalysation is randomly generated, using probabilities that are specified in the models.

3.1 The Kauffman model

The original model behind the formalized CRS by Steel (2000) was introduced and analysed by Kauffman (Kauffman, 1986, 1993). The model was a simple abstract origin of life model based on "protein-like polymers" randomly catalysing concatenations and cleavages of "protein-like polymers" and is described in Hordijk and Steel (2004). With the formalized notation of a CRS, the set of molecules $X = X(n)$ is the set of all possible sequences up to a given length n over a k -letter alphabet $\{0, 1, \dots, k - 1\}$. Only the case $k = 2$ was considered in details by Kauffman, while the general case was considered e.g. in Hordijk and Steel (2004) and Mossel and Steel (2005). The molecules are regarded as oriented, i.e. we distinguish between the molecules 001 and 100. The food set $F = X(t)$ consist of all sequences up to a length t , for a small and fixed value of $t < n$, representing simple, free available molecules. The set $\mathcal{R} = \mathcal{R}(n)$ of allowable reactions is the set of ligation reactions (ligation of two, possibly the same, molecules) and cleavage reactions, where reactants and products are in $X(n)$. As it emerges from Hordijk and Steel (2004), the reactions should be regarded as bi-directional, i.e. a reaction $r \in \mathcal{R}(n)$ represents both the ligation and cleavage reaction. For $k = 2$ and $n = 5$ we have for example the reactions

- A ligation reaction between 11 and 0/A cleavage reaction of 110
 $11 + 0 \longleftrightarrow 110, \quad 11, 0, 110 \in X(n).$
- A ligation reaction between 10 and 00/A cleavage reaction of 1000
 $10 + 00 \longleftrightarrow 1000, \quad 10, 00, 1000 \in X(n)$

The number of molecules and number of reactions in Kauffman's model, is given by

$$|X(n)| = \sum_{i=1}^n k^i, \quad |\mathcal{R}(n)| = \sum_{i=2}^n (i-1)k^i \quad (\text{Steel (2000)}),$$

so we distinguish between the reactions

$$10 + 1010 \longleftrightarrow 101010 \quad \text{and} \quad 1010 + 10 \longleftrightarrow 101010,$$

although the reactions have the same set of reactants and the same set of products.

The set of catalysations C is randomly generated, by assigning elements (x, r) of $X(n) \times \mathcal{R}(n)$ independently with the same probability $p(n)$. Notice that the

probability does not depend on either x or r , which makes this model very simple. Notice also that a molecule is allowed to catalyse a reaction, where the molecule itself is present in either the set of reactants or the set of products. In short notation we have described the following model.

Definition 3.1. *In the **Kauffman model**, a catalytic reaction system $\mathcal{L}(n) = (X(n), \mathcal{R}(n), C)$ over F satisfies*

$$X(n) = \{0, 1, \dots, k - 1\}^{\leq n}.$$

$$F = X(t), \text{ for a small value } t < n.$$

$$\mathcal{R}(n) = \text{The set of ligation and cleavage reactions.}$$

$$C \subseteq X(n) \times \mathcal{R}(n) : \forall x \in X(n), \forall r \in \mathcal{R}(n), Pr[(x, r) \in C] = p(n).$$

Figure 7 is an example of an *RAF* set in the Kauffman model, for $\mathcal{L}(5)$ with $k = 2$ and $t = 2$. The black dots are representing molecules, and the white dots are representing reactions. The two-way arrows shows which molecules are present in the reactions, and the dashed arrows shows which molecules catalyses the reactions.

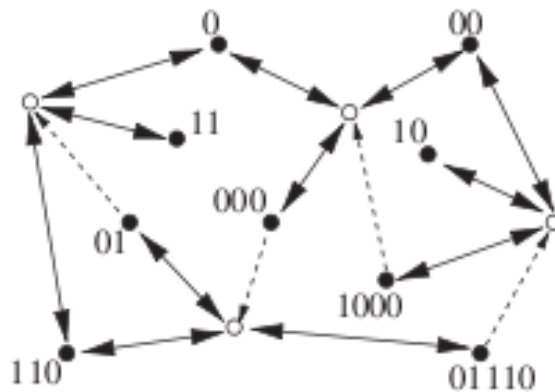


Figure 7: An *RAF* set for $\mathcal{L}(5)$ with $k = 2$ and $t = 2$ (Hordijk and Steel (2004), Fig. 3.)

We will now let P_n denote the probability that a catalytic reaction system $\mathcal{L}(n)$ has an *RAF* set, $P_\infty = \lim_{n \rightarrow \infty} P_n$ denote the limit probability and $\mu_n = p(n) \times |\mathcal{R}(n)|$ denote the expected number of reactions that each molecule catalyses.

Kauffman claimed that life-like subsystems (*RAF* sets) must spontaneously arise with high probability once the number of molecules becomes sufficiently large, i.e. $P_\infty = 1$, a conclusion which afterwards has been criticized and examined. A critic of this claim was made by Lifson, while Steel (Steel (2000)) solved the discussion by looking at their different assumptions. Kauffman actually assumed that $p(n) = c$ was a constant, which causes μ_n to increase unnaturally rapidly with n ($\mu_n \propto 2^n$). Lifson's interpretation was that $p(n)$ was proportional to $1/|\mathcal{R}(n)|$, i.e. $\mu_n = c$, which is much more realistic (Steel (2000)). The following theorem (Theorem 3.1) has proven them both right. Under the strong assumption in Kauffman's model, his claim holds, and under Lifson's assumption, it is not certain whether an *RAF* set will arise, and for low values of $p(n)$ it will almost certainly not.

Theorem 3.1. (Steel 2000)

1. If $\mu_n \geq cn^2$ where $c > \log_e(k)$, then $P_\infty = 1$.
2. If $\mu_n < \frac{1}{3}e^{-1}$, then $P_\infty = 0$.

If we want to use CRS's and *RAF* sets to explain the origin of life, we need some results that ensure that an *RAF* set will arise almost certainly with a low degree of catalysation, so that it can actually happen in nature. Theorem 3.1 shows that the degree of catalysation required is much less than Kauffman suggested, and more than the model considered by Lifson requires. A degree of catalysation of the order $\mu_n \propto n^2$ is still quite a lot compared to systems of molecules in nature, so Theorem 3.1 does not entirely solve the important question on whether CRS's and *RAF* sets can be used to describe the origin of life.

The main question in Hordijk and Steel (2004) is therefore what happens in between the two extreme cases described above, i.e. they are interested in the transition phase for the existence of an *RAF* set from being highly unlikely to being almost certain. Figure 8 and 9 shows results from their simulation study.

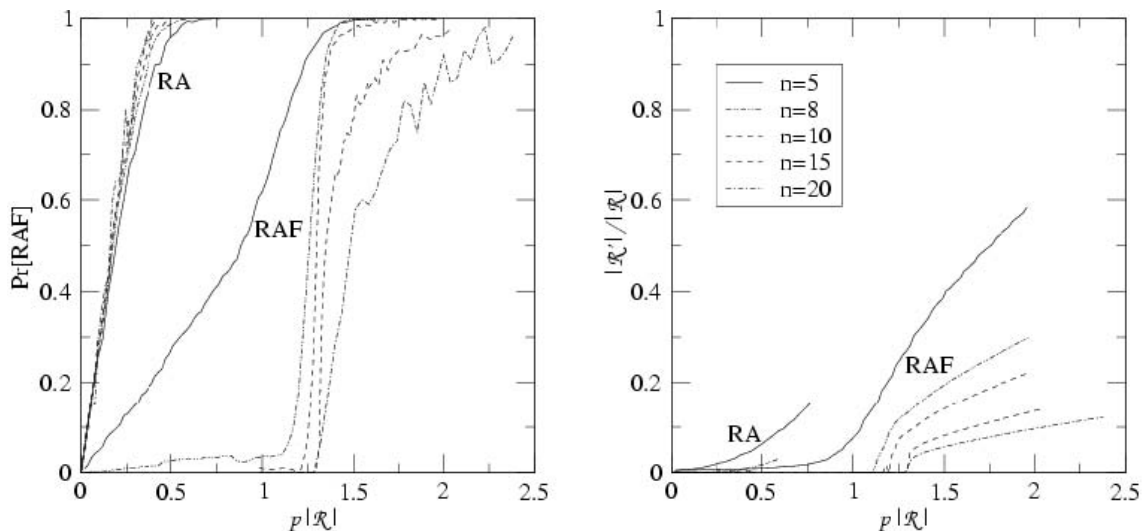


Figure 8: The probability of occurrence and the relative size of *RAF* sets in Kauffman's model (Hordijk and Steel (2004), Fig. 4).

In Figure 8 the observed probabilities $Pr[RAF]$ and average relative sizes of *RAF* sets $|\mathcal{R}'|/|\mathcal{R}|$ are plotted against μ_n for a range of $p(n)$ values for some values of $n \leq 20$, with $k = 2$ and $t = 2$ (Average over 100 to 10,000 simulations). For small values of $p(n)$, where $\mu_n < 1$ and $n > 5$, we can see that the $Pr[RAF]$ is negligible, after which there is a sharp transition between $1.25 \leq \mu_n \leq 1.5$. Eventually the $Pr[RAF]$ converges to 1. This result is in line with Theorem 3.1. We can also see that the average size of an *RAF* set increases with increasing $p(n)$, but except for the transition phase, it only increases slowly.

Figure 9 shows how the transition point scales with n . The black circles represent the average values of μ_n over 1000 simulations, for values of $p(n)$ for which the probability of finding an *RAF* set is around 0.5, for $n = 7, 8, \dots, 20$. A linear fit

was made ($l(n) = 1.107 + 0.081n$), so at least up to $n = 20$, the transition seems to scale linearly with n , which is a much better result, than Theorem 3.1.

A simulation study similar to this will be carried out in the two more realistic models described below. The simulations and the results are presented in Section 4.

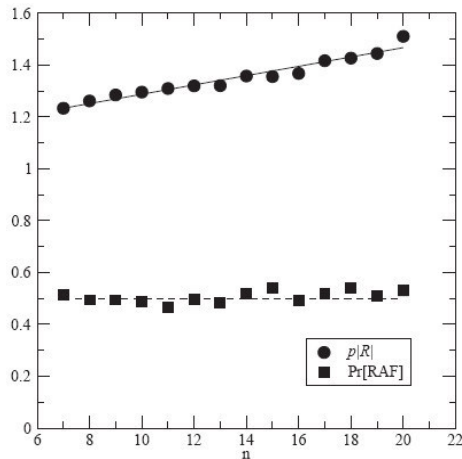


Figure 9: The transition points as a function of n (Hordijk and Steel (2004), Fig. 6).

3.2 Random sequence based model

In Mossel and Steel (2005) a more general model is investigated. The set of molecules $X(n)$ and the food set $F = X(t)$ is consistent with the Kauffman model, and the set of reactions $\mathcal{R}(n)$ still consists of ligation and cleavage reactions. A generalisation from the Kauffman model is that a ligation reaction and the corresponding cleavage reaction, is now considered as two separate reactions. We denote the set of ligation reactions with $\mathcal{R}_+(n)$, and the set of cleavage reactions with $\mathcal{R}_-(n)$. Another generalisation is that they allow the probability of a catalysation to depend on the molecule which catalyse the reaction. Still the probability does not depend on the reaction, so a given molecule from $X(n)$ can still randomly catalyse any given reaction from $\mathcal{R}(n)$. They formally write the model as

Definition 3.2. *In the Random sequence based model, a catalytic reaction system $\mathcal{L}(n) = (X(n), \mathcal{R}(n), C)$ over F satisfies*

$$X(n) = \{0, 1, \dots, k-1\}^{\leq n}.$$

$$F = X(t), \text{ where } t < n \text{ is a small value.}$$

$$\mathcal{R}(n) = \mathcal{R}_+(n) \cup \mathcal{R}_-(n).$$

$C \subseteq X(n) \times \mathcal{R}(n)$ is a random assignment of catalysation with requirements:

(R1) *The events $((x, r) \in C : x \in X(n), r \in \mathcal{R}_+(n))$ are independent.*

(R2) *For each $x \in X(n)$ and $r \in \mathcal{R}_+(n)$, $Pr[(x, r) \in C]$ can depend on x .*

In this model the total number of reactions is

$$|\mathcal{R}(n)| = |\mathcal{R}_+(n)| + |\mathcal{R}_-(n)| = 2 \sum_{i=2}^n (i-1)k^i .$$

This model is clearly more general, e.g. it allows different catalysation probabilities for ligation and cleavage, and the catalysation ability of a molecule can depend on the structure of the molecule. From the requirements we get that the expected number of reactions in $\mathcal{R}_+(n)$ that molecule x catalyses, is given by

$$\mu_n(x) = Pr[(x, r) \in C] \times |\mathcal{R}_+(n)|, \quad \text{for any } r \in \mathcal{R}_+(n).$$

The following results, from Mossel and Steel (2005), provide some upper and lower bounds on how much catalysation there is needed in the Random sequence based model, for the occurrence of an *RAF* set. In Section 5 we will extend the results to the Template based model described below.

Theorem 3.2. (*Mossel and Steel (2005)*) Consider a catalytic reaction system $\mathcal{L}(n)$ satisfying (R1) and (R2) and with $F = X(t)$, $t < n$. Let $\lambda \geq 0$.

1. Suppose that $\mu_n(x) \leq \lambda n$ for all $x \in X(n)$. Then $\mathcal{P}_n \leq 1 - \exp(-2\lambda x_t^2(1 + \mathcal{O}(\frac{1}{n})))$ ($\rightarrow 0$ as $\lambda \rightarrow 0$)
2. Suppose that $\mu_n(x) \geq \lambda n$ for all $x \in X(n)$, or that $\mu_n(x) \geq \lambda \theta_n |x|$ for all $x \in X(n)$, where $\lambda > \log_e(k)$ and where $\theta_n = \frac{1}{k}(1 + \frac{nk^{n+1}}{r_n}) \sim 1$. Then $\mathcal{P}_n \geq 1 - \frac{k(ke^{-\lambda})^t}{1 - ke^{-\lambda}}$ ($\rightarrow 1$ as $\lambda \rightarrow \infty$)

Corollary 3.3. (*Mossel and Steel (2005)*) Consider a random catalytic reaction system $\mathcal{L}(n)$ satisfying (R1) and (R2) and with $F = X(t)$, $t \leq n$.

1. If $\max_{x \in X(n)} \frac{\mu_n(x)}{n} \rightarrow 0$ as $n \rightarrow \infty$, then $\lim_{n \rightarrow \infty} \mathcal{P}_n = 0$.
2. If $\min_{x \in X(n)} \frac{\mu_n(x)}{|x|} \rightarrow \infty$ as $n \rightarrow \infty$, then $\lim_{n \rightarrow \infty} \mathcal{P}_n = 1$.

3.3 Template based model

In Kauffman's model and in the Random sequence based model catalysts are assigned randomly according to some probability, which either is fixed (Kauffman's model) or only depends on the catalysts (The Random sequence based model).

A more realistic model would then be a model, where the ability and the probability of a molecule to catalyse a reaction depends on both the reaction and the catalyst. In this project we want to investigate the specific case where catalysation is driven by complementarities. The idea comes from the reactions among RNA-like molecule types where catalysation is driven by the base-pair complementarities $A - U$ and $C - G$.

For example, in the ligation reaction $AUCG + AGU \rightarrow AUCGAGU$, which is also shown in Figure 10, $GCUC$ is allowed to catalyse the reaction, and in the cleavage reaction $CGGUAU \rightarrow CGGU + AU$, AU is allowed to catalyse the reaction. The important thing is that the catalyst is able to stick to both sides of the ligation/cleavage.

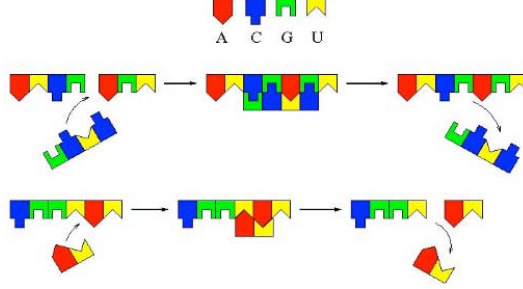


Figure 10: Catalysation in the template based model (Hein 2010, Fig. 2).

Let $X(n)$, F and $\mathcal{R}(n)$ be consistent with the Random sequence based model. Let k be an even number, and let elements from $\{0, \dots, k-1\}$ be paired, according to a chosen system, for example $\{0, 1\}, \dots, \{k-2, k-1\}$. We will refer to this, as the base-pair complementarity. We can then write a random ligation reaction $r \in \mathcal{R}_+(n)$ as

$$(r_u, \dots, r_1) + (r'_1, \dots, r'_v) \longrightarrow (r_u, \dots, r_1, r'_1, \dots, r'_v),$$

where $r_i, r'_j \in \{0, \dots, k-1\}$, for $i = 1, \dots, u, j = 1, \dots, v$ and $2 \leq u+v \leq n$. We will now use the notation r_{k+1} , when we want to change the element $r_k \in \{0, \dots, k-1\}$, to its complementarity (For example 0 to 1 or 1 to 0).

Definition 3.3. A molecule $x \in X(n)$ satisfies the **base-pair complementarity** for $r \in \mathcal{R}(n)_+$ if x is obtained by

- adding elements from $\{0, \dots, k-1\}$ to the sequence $(r_u + 1, \dots, r_1 + 1, r'_1 + 1, \dots, r'_v + 1)$.
- truncating the sequence $(r_u + 1, \dots, r_1 + 1, r'_1 + 1, \dots, r'_v + 1)$, while containing the subsequence $(r_1 + 1, r'_1 + 1)$.
- adding elements from $\{0, \dots, k-1\}$ to one of the ends of the sequence $(r_u + 1, \dots, r_1 + 1, r'_1 + 1, \dots, r'_v + 1)$, and truncating the sequence in the other end, while containing the subsequence $(r_1 + 1, r'_1 + 1)$.

A molecule $x \in X(n)$ satisfies the base-pair complementarity for $r \in \mathcal{R}_-(n)$, if and only if it satisfies the base-pair complementarity for the corresponding $r \in \mathcal{R}_+(n)$.

For example, when $n = 4$ and $k = 2$, the reaction $10+11 \rightarrow 011$, can be catalysed by $x \in \{10, 010, 100, 0100, 0010, 1000, 1001, 1010\}$. Now that the complementarity is well-defined, we can define the model that we want to investigate.

Definition 3.4. In the **Template based model**, a catalytic reaction system $\mathcal{L}(n) = (X(n), \mathcal{R}(n), C)$ over F satisfies

$$X(n) = \{0, 1, \dots, k-1\}^{\leq n}.$$

$F = X(t)$, where $t < n$ is a small value.

$$\mathcal{R}(n) = \mathcal{R}_+(n) \cup \mathcal{R}_-(n).$$

$C \subseteq X(n) \times \mathcal{R}(n)$ is a random assignment of catalysation with requirements:

(R1) The events $((x, r) \in C : x \in X(n), r \in \mathcal{R}(n))$ are independent.

(R2) For $x \in X(n)$ and $r \in \mathcal{R}(n)$, $Pr[(x, r) \in C]$ can depend on both x and r .

$Pr[(x, r) \in C] > 0$ if and only if x satisfies the complementarity of r .

The main thing is now to investigate how this model differ from the previous models regarding the probability of finding an *RAF* set and how much catalysation there is needed. In the light of the work by Hordijk and Steel (2004) and Mossel and Steel (2005), we will try to come up with similar results both analytically and by simulation. It is no longer easy to find $\mu_n(x)$, like it was in the Random sequence model, therefore we will take a look at the average number of reactions, that an average molecule catalyses, which can be determined from the formula

$$\mu_n = \sum_{x \in X(n)} \sum_{r \in \mathcal{R}(x)} Pr[(x, r) \in C] / |X(n)| \sim |C| / |X(n)|,$$

where $\mathcal{R}(x)$ is the set of allowable reactions, for the molecule $x \in X(n)$ to catalyse, and $|C|$ is the average size of the set C , determined from simulations. In the Kauffman model, and in the Random sequence based model with $Pr[(x, r) \in C] = p(n)$, we have that

$$\mu_n = p(n) |\mathcal{R}(n)| \sim |C| / |X(n)|.$$

In order to simulate from the model above, we have to choose the probability of a catalysation, for every $x \in X(n)$ and every $r \in \mathcal{R}(n)$. To measure just what the template assumption does to the model, we will first let $Pr[(x, r) \in C] = p(n) > 0$. To add another element towards a more realistic model, we also turn towards the thermodynamic world for inspiration, which is described below.

3.4 Thermodynamic models

The Thermodynamic model is based on the field of nucleic acid sequence structure formation. In this model we will assume that the probability of $x \in X(n)$ catalysing a reaction $r \in \mathcal{R}(n)$, if x fulfill the base-pair complementarity, to be

$$Pr[(x, r) \in C] = a^s b^t p,$$

where $a < 1$, $b > 1$ and $p > 0$ are parameters, and $s \in \{1, 2\}$ is defined to be the number of reactants and $t \in \{2, \dots, n\}$ is defined to be the length of the maximum overlap between the catalyst and the reactants.

For example the ligation reaction $10 + 11 \rightarrow 011$, can be catalysed by $x = 1000$, where $s = 2$ and $t = 3$.

This model tends to favour catalysis of reactions with long reactants by long catalysts, and also tends to favor cleavage over ligation. So with this model, given the same overall average probability of a molecule catalysing a reaction as for the non-template model, it would be more difficult for this model to 'get off the ground', as the initial molecules that can be generated from F are short molecules.

We can get rid of one of the parameters, if we solve the equation

$$\sum_{c=(x,r) \in C \subseteq X(n) \times \mathcal{R}(n)} a^s b^t = 1,$$

which means that we assume the average probability of catalysis is p . Because $s = 1$ or 2 , this equation expands to become a quadratic equation of a , and can be solved numerically. On the other hand, this quadratic equation can be solved by the following formula:

For a given value of $b > 1$

$$a = \frac{\sqrt{8 \frac{\sum_{l=2}^n \sum_{n_1=1}^{l-1} \sum_{t=2}^l \{2^{l(g(n_1,l,t)+h(n,n_1,l,t))}\}}{\sum_{l=2}^n \sum_{n_1=1}^{l-1} \sum_{t=2}^l \{2^{l(g(n_1,l,t)+h(n,n_1,l,t))} b^t\}}} + 1 - 1}{2},$$

where

$$g(n_1, l, t) = \begin{cases} 1 + l - t & \text{if } t > \max(n_1, l - n_1) \\ \min(n_1, l - n_1, t - 1) & \text{otherwise} \end{cases}$$

$$h(n, n_1, l, t) = \begin{cases} 0 & \text{if } t \leq \min(n_1, n_2) \\ 2^{n-l+1} - 2 & \text{else if } t \leq \max(n_1, n_2) \\ 2 * (2^{n-l+1} - 2) & \text{else if } t < l \\ (n - l) * 2^{n-l+1} & \text{else} \end{cases}$$

Here when the length of overlap is t , the coefficient functions f and g are taken for the case when there is overhang and when there is not, respectively. Here in this formula, some catalisations are counted repeatedly; for example, the reaction $r: 11 + 11 \rightarrow 1111$ with catalysis 0000 is counted at least once for exact matching and three times for overhang. However, since we are only seeking for a simplification of the formula and we are just trying to find the 'average', we can define this as our formula for finding a .

4 Simulation

We will investigate data, from simulations according to the following 4 models.

- **Simulation 1** The Kauffman model, with $Pr[(x, r) \in C] = p(n)$.
- **Simulation 2** The Random sequence based model, with $Pr[(x, r) \in C] = p(n)$.
- **Simulation 3** The Template based model, with $Pr[(x, r) \in C] = p(n)$, if x satisfies the base-pair complementarity for r .
- **Simulation 4** The Thermodynamic model, with $Pr[(x, r) \in C] = a^s b^t p$, if x satisfies the base-pair complementarity for r , where a, b, s and t is defined according to the Thermodynamic model.

We will use Simulation 1 to check if the implementations of our algorithm produces the right *RAF* sets, i.e. we will try to get results similar to the simulations, made by Hordijk and Steel (2004). In addition to that, we have already used this simulation in Section 2.7, to investigate the running time and compare it to the time found by Hordijk and Steel (2004). Notice that the model in Simulation 2 is similar to the model in Simulation 1, and only differs in the way we look at the ligation and the cleavage reactions. We will use the results produced in Simulation 2 to compare with the results produced in Simulation 3. In these two simulations we have the same assumptions about how we look at ligation and cleavage reaction (In contrast to Simulation 1) and in both cases the probability does not depend on neither the catalyst nor the reaction, and therefore we will be able to investigate exactly what the assumption about base-pair complementarity does. In Simulation 4 we try to investigate the even more realistic model, according to the Thermodynamic model. In all simulations, we will use $k = 2$ (binary sequences) and $t = 2$ (The food set $F = \{0, 1, 00, 01, 10, 11\}$). We then want to simulate for different values of n , p and b . For each value of n , p and b we will generate 100 – 500 CRS's, and with the help of the algorithm we will register the following:

- The observed probability of finding an *RAF* set, $Pr[RAF]$.
- The average size of the *RAF* sets, when they are present, $|\mathcal{R}'|$.
- The average size of the set of catalysations, $|C|$.
- The average running time, τ .

We will use Implementation 3 in all the simulations, apart from the simulations where we investigate the running time for the different implementations.

4.1 The Java-program

In this section, we will give a brief description of the simulation program.

4.1.1 The set of molecules and the food set

First we construct the set of molecules, which is easily recognized as binary numbers, where zero's are added from the left. There are 2^i elements of length i , corresponding to the first 2^i 'th binary numbers with zero's added. The food set is easily constructed afterwards, but instead of the binary sequences, we write the corresponding numbers from the set of molecules. This is common for all four models.

$$X(n) = \{0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, \dots\}.$$
$$F = X(t) = \{0, 1, 2, 3, 4, 5\}.$$

4.1.2 The set of reactions

Next we construct the reactions, by making two lists of lists, corresponding to the reactants and products. Again we represent the molecules with integers. The ligation reactions are constructed by running through the set of molecules two times, and if the length of the two sequences combined is smaller than or equal to n , the two numbers of the molecules make a list of reactants. The number of the molecule from the ligation will make the list of products, so for example we get that the reaction $0 + 1 \rightarrow 01$, will be represented by reactants $\{0, 1\}$ and a product $\{3\}$. The corresponding cleavage reaction is obtained by switching the two lists, so in the example the reactant is $\{3\}$ and the products are $\{0, 1\}$.

$$\text{Reactants} = \{\{0, 0\}, \{2\}, \{0, 1\}, \{3\}, \{0, 2\}, \{6\}, \{0, 3\}, \{7\}, \{0, 4\}, \{8\}, \dots\}.$$
$$\text{Products} = \{\{2\}, \{0, 0\}, \{3\}, \{0, 1\}, \{6\}, \{0, 2\}, \{7\}, \{0, 3\}, \{8\}, \{0, 4\}, \dots\}.$$

We will use this representation of the reactants and the products, when we construct the set of catalysations, but in the end, when data is running through the algorithm, we will ensure, that if the reactants (or products) in one list are the same, only one representation will be in the list, for example $\{0, 0\} \rightarrow \{2\}$, will be changed to $\{0\} \rightarrow \{2\}$. This has to do with the implementations of the algorithm. This part is also common for all four models.

4.1.3 The set of catalysations

The set of catalysations is represented by a list of lists of integers, so the i 'th list consists of the numbers of the reactions, that molecule i catalysis.

$$\text{Catalysations} = \{\{\}, \{\}, \{\}, \{4, 5\}, \{2\}, \{\}, \{\}, \{7\}, \{\}, \{\}, \{0, 5\}, \dots\}.$$

The part of the program, where we construct these lists, differ in the four simulations.

- **Simulation 1** In this model, we have to run through all molecules and all ligation reactions, and with the given probability $p(n)$, adding both the ligation reaction and the cleavage reaction to the list belonging to the molecule (These are only counted as one reaction).
- **Simulation 2** In this model we have to run through all molecules and all reactions, and with the given probability $p(n)$, adding the reaction to the list belonging to the molecule.

- **Simulation 3 and 4** In these models, we have to run through all reactions and then construct all the molecules, that are allowed to catalyse the given reaction, according to the base-pair complementarity. A molecule can catalyse a cleavage reaction, if and only if it can catalyse the corresponding ligation reaction, so we just have to run through the ligation reactions. When the set of allowable catalysts has been made for a given reaction, we will add the reaction independently to each of the elements in the set of allowable catalysts with the probability specified in the model, and the same thing will happen to the corresponding cleavage reaction. The table below shows how we construct the allowable catalysts, for the reaction $01 + 100 \rightarrow 01100$.

01	100
0	0
0	01
0	011
0	011 x_1
10	0
10	01
10	011
10	011 x_1
x_2 10	0
x_2 10	01
x_2 10	011
x_2 10	011 x_1

We build the molecules character by character, and for every step, adding the molecule to a list. The characters in the part of the molecules, overlapping the reactants are determined by the complementarity. At the ends we can add all molecules x_1 and x_2 from $X(n)$, up to a length, where the total length of the molecule will be less then or equal to n . In Simulation 4 we make sure that we save the length of the overlap, which is needed to find the probabilities. Sometimes the same molecule can be attached to a reaction in more than one way, for example if we look at the reaction $1011+1 \rightarrow 10111$, the molecule 000 can be attached in two ways, either with an overlap of two or three characters. The value of the overlap should then be the maximum overlap value.

The program is not suited to simulate systems where n is larger than 10. There may be some improvement of the program, which can make it run faster, but the limit factor is in the part of the program where the set of catalysations are constructed. In Simulation 2, where we run through all molecules and all reactions, when we are constructing the set of catalysations, there is for $n = 10$ roughly $|X(n)||\mathcal{R}(n)| \sim 67,000000$ elements. In Simulations 3 and 4 there are a lot less allowable catalysations (For $n = 10$ roughly 15,000000), but there is a lot of work in the construction of these, which described above differ from Simulation 2. One may consider a totally different way of simulating from these models, where not all the possible elements have to be constructed at first. It may be possible to simulate some smaller sets of molecules and

reactions, which are subsets of the sets constructed above, and from these subsets simulate catalysations.

4.2 Results

In this section we will present the results obtained from the simulations and point out what these results tell us about the models.

4.2.1 Simulation 1

Figure 11 shows the observed probabilities of *RAF* sets for various n against simulated values of $\mu_n \sim |C|/|X(n)|$, in the Kauffman model. For each $n = 5, \dots, 10$, 30 $p(n)$ -values nearby the transition phase was chosen to be used in the simulation and spline-functions were used to fit lines through the observations. If we compare the graph in Figure 11 to the left graph in Figure 8, we can see that except from the stochastic variation, the graphs seems similar for $n = 5, 8$ and 10 , which are the only values of n , that we can compare. It seems as if our algorithm finds all the *RAF* sets (and not too many), i.e. hopefully we have taken all special cases such as cycles into account. We have much less repeated simulations than Hordijk and Steel (2004), so a more precise result could be obtained, with more simulations. From Figure 11, we were able to determine roughly the values of $p(n)$, for which $Pr[RAF] = 0.5$. This information was used in Section 2.7, in the study of the running time.

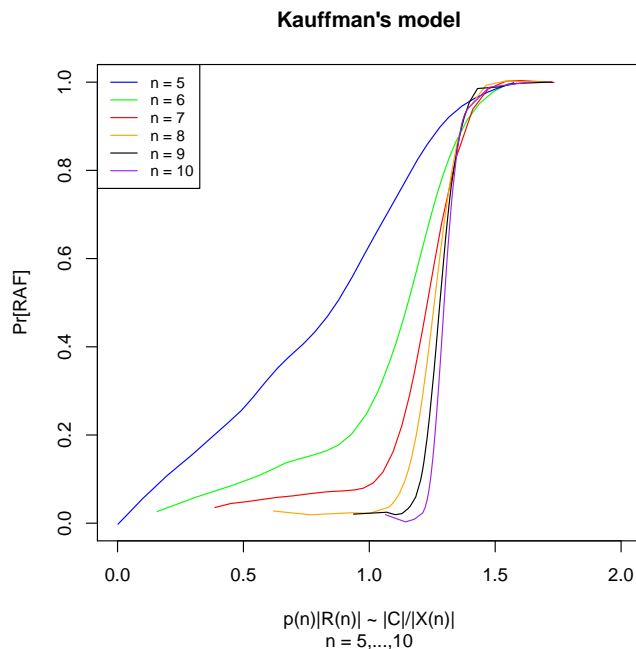


Figure 11: The probability of occurrence of *RAF* sets in Kauffman's model (Simulation 1).

4.2.2 Simulation 2 and Simulation 3

Figure 12 and 13 shows the observed probabilities and the average relative size of *RAF* sets for various n against simulated values of $\mu_n \sim |C|/|X(n)|$, in the two models from Simulation 2 and Simulation 3. Like in the Kauffman model, we chose for each model and each $n = 5, \dots, 10$, 30 $p(n)$ -values nearby the transition phase to be used in the simulation and spline-functions were used to fit lines through the observations.

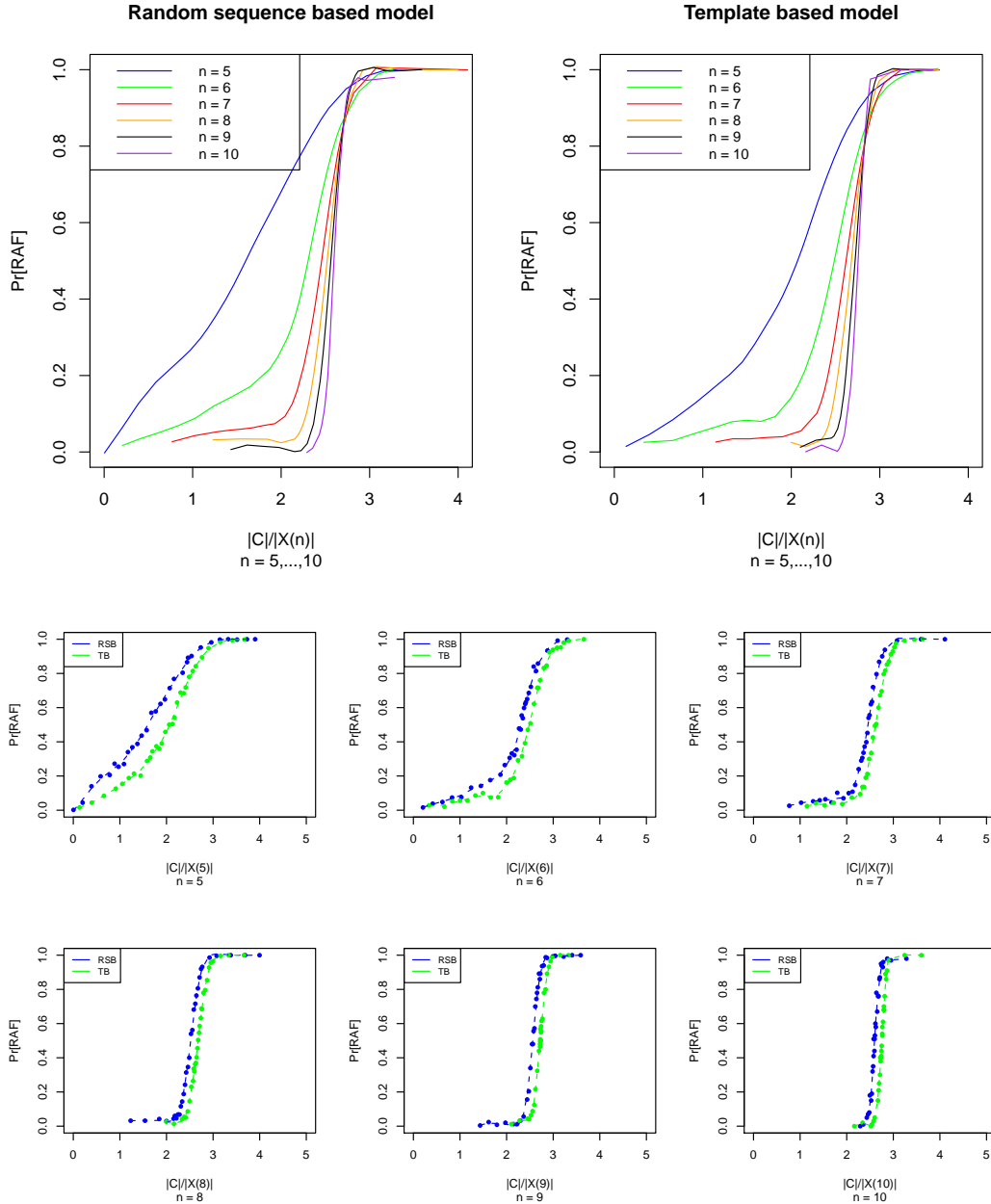


Figure 12: The probability of occurrence of *RAF* sets in the Random sequence based model and in the Template based model (Simulation 2 and Simulation 3).

The two graphs in the top of Figure 12, clearly shows the same pattern as Figure 11, e.g. to begin with the probability of an *RAF* set occurring is negligible, and then suddenly there is a sharp transition. But while the transition in the Kauffman model is roughly $0.8 \leq \mu_n \leq 1.3$ (In Figure 11 where we consider $n = 5, \dots, 10$), the transition phase in these models are roughly between $1.6 \leq \mu_n \leq 2.6$, i.e. twice as high. As described, we only count a ligation reaction and the corresponding cleavage reaction as one reaction in the Kauffman model and we always add the reactions simultaneously, i.e. either both of them or non of them are in the set of catalysation.

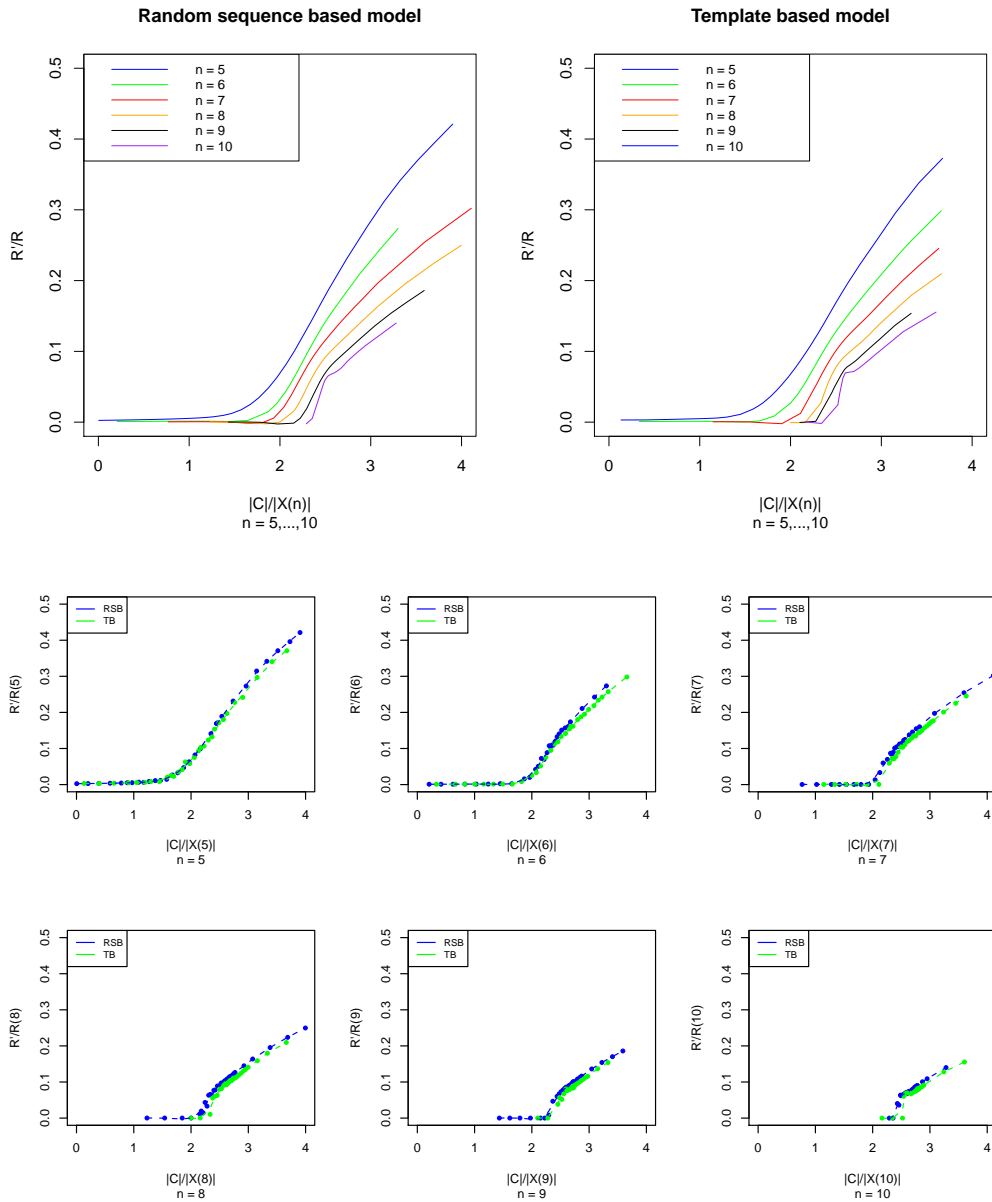


Figure 13: The relative size of *RAF* sets in the Random sequence based model and in the Template based model (Simulation 2 and Simulation 3).

If we instead count the reactions as two reactions, but still add them simultaneously, we would just remove the transition phase in Figure 11 to the right, and it actually seems like we would get the exact same result as in Figure 12. Thus it makes no difference whether we simulate from the model in Simulation 1 or the model in Simulation 2, the only difference is how we count the reaction. One might have expected the subsets created in the Random sequence based model to be different than the ones from the Kauffman model, and thereby change the probability of finding an *RAF* set, but it does not seem to be the case.

In the six graphs in the bottom of Figure 12 we compare the probabilities in the two models in Simulation 2 and Simulation 3, one graph for each n . The blue dots and lines correspond to the Random sequence based model from Simulation 2, and the green ones to the Template based model in Simulation 3. It is clear that we need a larger value of $p(n)$ in the Template based model, to get the same amount of catalysation as in the Random sequence based model, because we have a lot fewer allowable catalysations. For the same amount of catalysation, it seems that the assumption about base-pair complementarity does not change the probability considerably. However in all the graphs we see that the probability in the Template based model is a bit smaller than in the Random sequence based model, but it seems as if it follows the same pattern as in the Random sequence based model. It could be an idea to look at simulations where $k = 4$ (RNA-like molecules), where the base-pair complementarity may have a larger influence.

The two graphs in Figure 13 show the same pattern as Figure 8. The average relative size of the *RAF* sets increases very slowly with increasing $p(n)$ -values, and then when it reaches the transition phase, it grows quickly.

In the six graphs in the bottom of Figure 13 we see, that even though it was a bit more difficult to find an *RAF* set in the Template based model, the *RAF* sets which are produced, are a bit smaller, than the ones in the Random sequence based model.

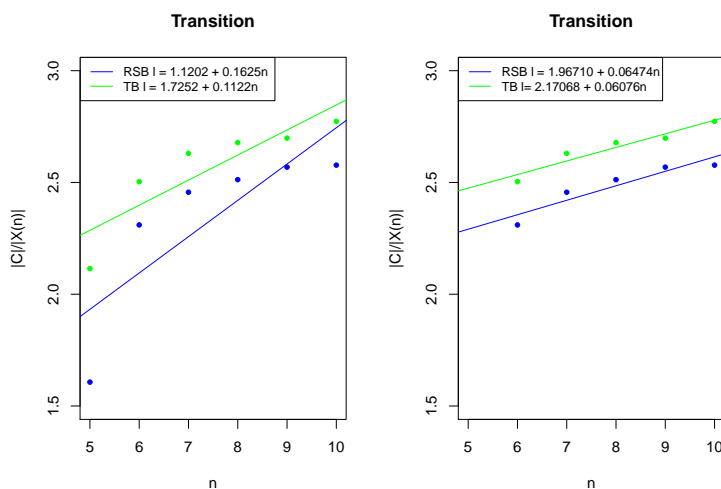


Figure 14: The transition points as functions of n in the Random sequence based model and in the Template based model (Simulation 2 and Simulation 3).

In Figure 14 we examine how the transition points scale with n in the simulations from before. The blue dots correspond to the Random sequence based model and the green dots to the Template based model. From the spline-functions used in Figure 12, we have for each $n = 5, \dots, 10$ and each model found the transition phase, i.e. the value of μ_n for which $Pr[RAF] \sim 0.5$. The graph to the right is the same as the graph to the left, except that we have removed the observations for $n = 5$. In each graph, we have made a linear fit. We will only consider the graph to the right. It seems as if the transition grows (at most) linear with n . Unfortunately we were not able to simulate data for n larger then 10, so with only 5 points, it is not possible to give a certain prediction of the transition. If it is linear, the two fitted lines are

- Random sequence based model: $l_1 = 1.96710 + 0.06474n$.
- Template based model: $l_2 = 2.17068 + 0.06076n$.

So at least it seems as if the transitions in the two models grow with the same rate, even though the Template based model has larger transition points. In the light of our results, the Template based model with a probability, which apart from the base-pair complementarity does not depend on neither the catalyst nor the reaction, is still a model, which is more realistic than previous models, and it is still worth considering in the question of the origin of life.

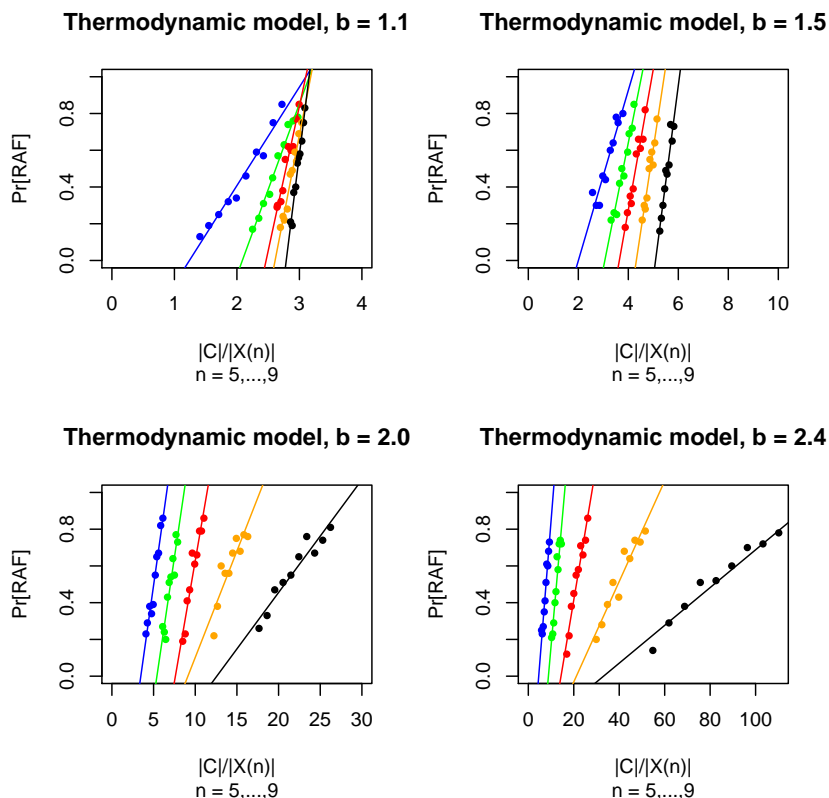


Figure 15: The probability of occurrence of RAF sets in the Thermodynamic model.

4.2.3 Simulation 4

In the Thermodynamic model we have two parameters, b and p . When $b = 1$ the model is identical to the Template based model, in which there only were a slight difference from the Random sequence based model (at least for values of $n \leq 10$). An interesting thing to investigate is, what happens when we enlarge the value of b . Figure 15 shows the observed probabilities of *RAF* sets for $n = 5, \dots, 9$ (5 = blue, 6 = green, 7 = red, 8 = orange and 9 = black), against the simulated values of $\mu_n \sim |C|/|X(n)|$, for four values of b ($b = 1.1, 1.5, 2.0$ and 2.4). We have only constructed data from the transition phase, i.e. 10 $p(n)$ -values, for which the probability of finding an *RAF* set roughly fulfil that $0.2 \leq Pr[RAF] \leq 0.8$. While the data where $b = 1.1$ does not differ that much from the Template based model, we see that as b grows it becomes more and more difficult to find an *RAF* set. This is an expected result, because the model favours more and more the long catalysts and the cleavage reactions with long reactants, as b becomes larger, so it becomes difficult to get started from the food set. The interesting thing is to see, how it effects the growth of catalysation needed to find *RAF* sets as a function of n , whether it still grows linear or we need an unnaturally high amount of catalysation.

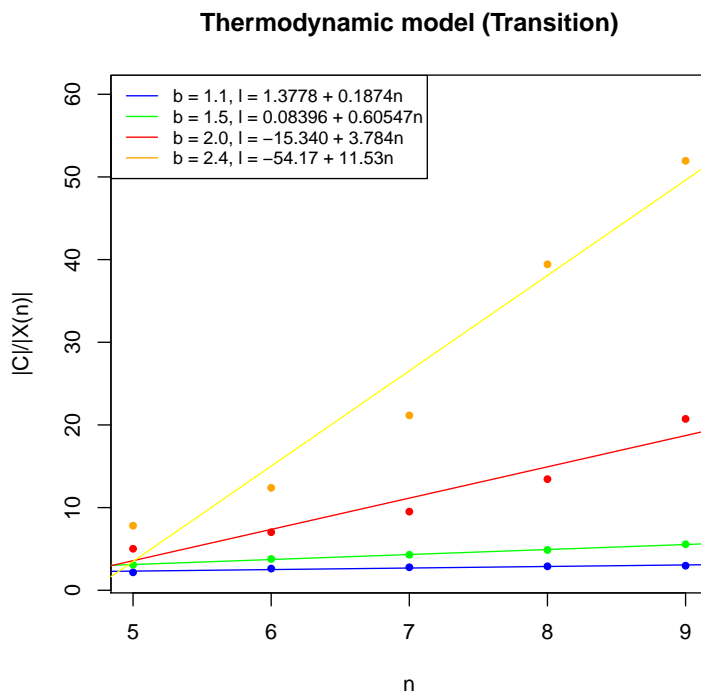


Figure 16: The transition points as functions of n .

In Figure 16 and 17, we investigate how the transition points grows with n , in the same way that we did with the Random sequence based model and the Template based model in Figure 14. In Figure 16 the transition points for $n = 5, \dots, 9$ and for $b = 1.1, 1.5, 2.0$ and 2.4 are shown in one graph. It is very clear that the size of b has a great influence a on how the transition grows with n .

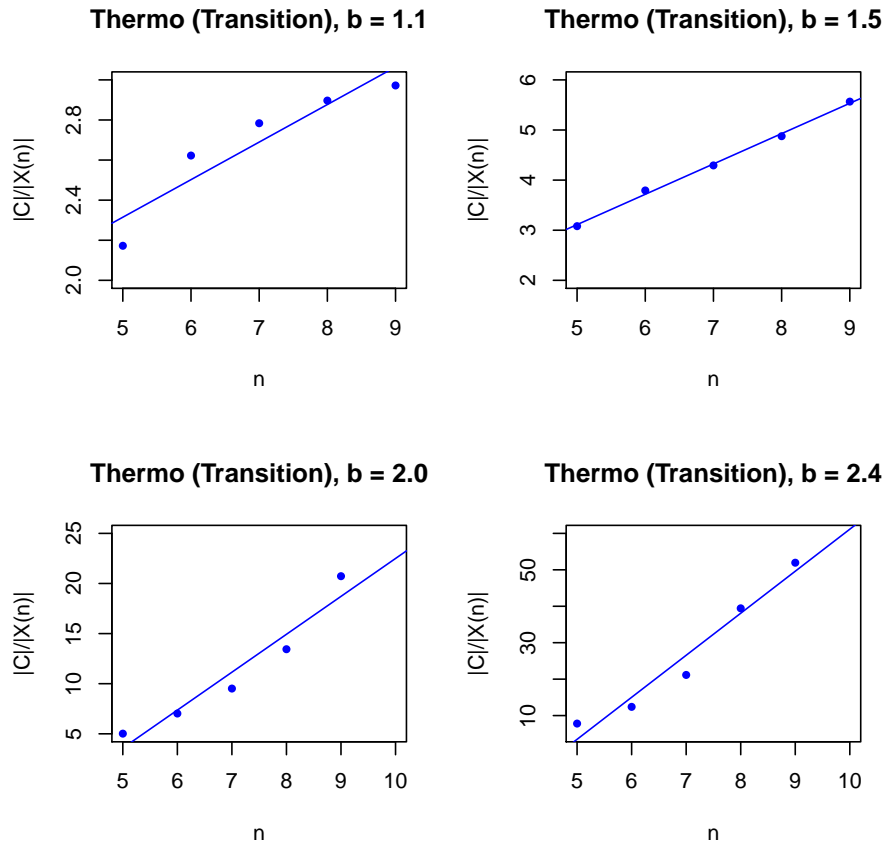


Figure 17: The transition points as functions of n .

Figure 17 shows one graph for each b -value. Again it is difficult to decide whether the transition grows linear or not, when we only have 5 observations. It seems like the transition grows at most linear for $b = 1.1$ and 1.5 , like in the Template based model, while for $b = 2.0$ and 2.4 it seems like the transition may grow faster than linear. If we in nature could observe $b = 2.0$ or 2.4 , it seems as if this model would not be the ideal model to describe the origin of life. The conclusions rely on few observations, so we need further work to give a more precise description.

5 Extended results in the Template based model

We now extend the results made by Mossel and Steel (2005) for the Random sequence based model to similar results in the case of the Template based model.

5.1 Notations

For a molecule $x \in X(n)$ and reaction $r \in \mathcal{R}(n)$, we will use the following notation.

$$\mathcal{R}(x) = \{r \in \mathcal{R}(n) \mid r \text{ and } x \text{ match}\}$$

$$X(r) = \{x \in X(n) \mid r \text{ and } x \text{ match}\}$$

$$x_r = |X(r)|$$

$$q_*(r) = \text{Pr}[r \text{ is not catalysed by any molecule in } X(n)]$$

(i.e. not Globally catalysed = GC)

d_n = The total number of pairs of allowable catalysations and reactions.

$$x_n = |X(n)|$$

$$r_n = |\mathcal{R}_+(n)| = \frac{1}{2}|\mathcal{R}(n)|$$

Lemma 5.1. *For any reaction $r \in \mathcal{R}(n)$ we have the following valuation.*

$$2^{n/2} \leq x_r \leq x_n.$$

Proof. (Just need the left inequality)

Let r be the reaction that involves two molecules x and y , where $|x| = n_1$, $|y| = n_2$, and $n_1 + n_2 \leq n$ (obviously).

Without loss of generality assume $n_2 \leq \lfloor \frac{n}{2} \rfloor$, and let α be the exact matching (i.e. reverse) of the substring that starts from the right most digit of x to the right most digit of y .

For example, let r be the reaction $010 + 100 \rightarrow 010100$, then α would be the exact matching of 0100 , i.e. $\alpha = 1011$.

Then any molecule in $X(n)$ that contains α of length $(n_2 + 1) \leq \lfloor \frac{n}{2} \rfloor + 1$ is a match with r .

Besides, any left substring of α is also matching r and has no repetition. Thus the total number of such molecules is

$$x_r \geq 1 + 2^1 + \dots + 2^{n-n_2-1} + n_2 - 1 = 2^{n-n_2} + n_2 - 2 \geq 2^{n/2}$$

□

Since $q_*(r) = (1 - p)^{x_r}$, we have

$$(1 - p)^{x_n} \leq q_*(r) \leq (1 - p)^{2^{n/2}}.$$

We denote $q_* = (1 - p)^{2^{n/2}}$.

Lemma 5.2. *Suppose $p > \frac{\lambda}{2^{n/2}}$, provided $\lambda > \log 2$, then*

$$\text{Pr}[RAF] \geq 1 - 2^{t+1}e^{-\lambda t}/(1 - 2e^{-\lambda}).$$

Proof. For any $s \geq t$ the probability that a molecule x with $|x| = s + 1$ is not generated by any forward GC reaction from $X(s)$ is given by

$$\prod_{r: \rho(r) \in X(s) \text{ and } \pi(r)=x} q_*(r) \leq q_*^s.$$

There are 2^{s+1} such molecules, so the probability that there is a molecule in $X(s + 1)$ that is not generated by a forward reaction from $X(s)$ is at most

$$1 - (1 - q_*^s)^{2^{s+1}} \leq 2^{s+1} q_*^s.$$

This in turn implies that the probability that all molecules in $X(n)$ are generated by forward GC reactions is at least $1 - 2^{t+1} q_*^t / (1 - 2q_*)$, provided $2q_* < 1$.

Then the probability that the system has an *RAF* set with $X(n) - F \subseteq \pi(\mathcal{R})$ is at least $1 - \frac{2^{t+1} q_*^t}{1 - 2q_*}$.

Hence when $p > \frac{\lambda}{2^{n/2}}$, $q_* \leq \left(1 - \frac{\lambda}{2^{n/2}}\right)^{2^{n/2}} < e^{-\lambda}$.

Finally we have shown that when $\lambda > \log 2$

$$Pr[RAF] \geq 1 - 2^{t+1} e^{-\lambda t} / (1 - 2e^{-\lambda}).$$

□

Lemma 5.3. *Suppose $p < \frac{\lambda n}{r_n}$, then*

$$Pr[RAF] \leq 1 - \exp(-\lambda x_t^2 (1 + \mathcal{O}(\frac{1}{n}))).$$

Proof. This is a direct consequence of Theorem 4.1(i) of the Mossel and Steel (2004), taking into account that $\mathcal{R}(x) \subseteq \mathcal{R}(n)$ for each molecule $x \in X(n)$.

To tighten this condition on p , one needs to consider how large x_r is compared to x_n , where r is taken over those reactions such that $\rho(r) \in X(t)$. In Lemma 5.2 it will be showed that $\frac{x_r}{x_n} \rightarrow 1$ as $n \rightarrow \infty$ for the reaction $r : 1 + 1 \rightarrow 11$.

Similarly the statement is true for all r with $\rho(r) \in X(t)$.

This idea is roughly stated as that given a specific string α , then for n large enough, almost all strings of length n contains α . □

Lemma 5.4. *Now we return to the $r : 1 + 1 \rightarrow 11$ case, then*

$$\frac{x_r}{x_n} \rightarrow 1 \text{ as } n \rightarrow \infty$$

Proof. $x \in X(n)$ catalyses r if and only if x contains the substring 00. Let a_i be the number of strings of length i that do not contain substring 00, then $x_r = \sum_{i=1, \dots, n} (2^i - a_i)$, whereas $x_n = \sum_{i=1, \dots, n} 2^i$.

For any string of length $i + 2$, it must start with 1 or with 01, followed by a substring that does not contain 00, of length $i + 1$ and i respectively, so $a_{i+2} = a_{i+1} + a_i$. The initial value can be obtained by simply counting that $a_1 = 2, a_2 = 3$.

Hence a_i is a Fibonacci Sequence with general solution

$$a_i = \frac{1}{\sqrt{5}} \left[\left(\frac{1+\sqrt{5}}{2} \right)^{i+2} - \left(\frac{1-\sqrt{5}}{2} \right)^{i+2} \right].$$

Obviously $\frac{a_i}{x_n} \rightarrow 0$ as $n \rightarrow \infty$, so $\frac{x_r}{x_n} \rightarrow 1$ as $n \rightarrow \infty$.

Therefore this bound on p cannot be very much improved on this end. \square

Lemma 5.5. $2^{\frac{n}{2}+1} r_n \leq d_n \leq 2x_n r_n$

Proof.

$$r_n = \sum_{r \in \mathcal{R}_+(n)} 1 \quad \text{and} \quad \frac{1}{2} d_n = \sum_{r \in \mathcal{R}_+(n)} x_r$$

Thus by Lemma 5.1, done. \square

Theorem 5.6. *Let μ_n be the expected average number of reactions each molecule in $X(n)$ catalyses, then $\mu_n = \frac{pd_n}{x_n}$. By the above lemmas, plus the fact that $x_n = 2^{n+1} - 2$, and $r_n = (n-2)2^{n+1} + 4$, we have the following results:*

Suppose $\mu_n > \lambda(n-2)2^{n/2+2}$, then $Pr[RAF] \geq 1 - k^{t+1}e^{-\lambda t}/(1 - ke^{-\lambda})$, provided $\lambda > \log k$.

Suppose $\mu_n < \lambda n 2^{-n/2-1}$, then $Pr[RAF] \leq 1 - \exp(-\lambda x_t^2(1 + \mathcal{O}(\frac{1}{n})))$.

5.2 Generalisation into k

We assume that each digit has exactly one matching digit, and this relation is symmetric.

Theorem 5.7.

Suppose $\mu_n > 2\lambda(n-2)k^{n/2+1}$, then $Pr[RAF] \geq 1 - k^{t+1}e^{-\lambda t}/(1 - ke^{-\lambda})$, provided $\lambda > \log k$.

Suppose $\mu_n < 2\lambda n k^{-n/2-1}$, then $Pr[RAF] \leq 1 - \exp(-\lambda x_t^2(1 + \mathcal{O}(\frac{1}{n})))$.

Proof. In this case, $x_n = \frac{k^{n+1}-k}{k-1} \geq \frac{k^{n+1}}{k-1}$ and $r_n = \frac{(n-1)k^{n+2} - nk^{n+1} + k^2}{(k-1)^2} \geq \frac{(n-2)k^{n+1}}{k-1}$.

Using exactly the same method, Lemma 5.1 becomes

$$k^{n/2}/(k-1) \leq x_r \leq x_n.$$

Consequently, Lemma 5.2 and 5.3 become

Suppose $p > \frac{\lambda(k-1)}{k^{n/2}}$, then $Pr[RAF] \geq 1 - k^{t+1}e^{-\lambda t}/(1 - ke^{-\lambda})$, provided $\lambda > \log k$

Suppose $p < \frac{\lambda n}{r_n}$, then $Pr[RAF] \leq 1 - \exp(-\lambda x_t^2(1 + \mathcal{O}(\frac{1}{n})))$

Lemma 5.5 becomes

$$\frac{2k^{n/2}}{k-1} r_n \leq d_n \leq 2x_n r_n$$

Then following exactly the same method, the theorem is proved. \square

6 Conclusions and further work

The ability of systems of molecular reactions to be simultaneously autocatalytic and sustained by some food source of simple molecules may have been an essential step in the origin of life. In this report we continued the work by Hordijk and Steel (2004) and Mossel and Steel (2005) of investigating the probability of autocatalytic sets arising.

We have described an alternative to the *RAF* algorithm in Hordijk and Steel (2004). The analysis of this new algorithm based on reference counting yields a worst case complexity of $\mathcal{O}(|X||\mathcal{R}|)$, which is significantly better than $\mathcal{O}(|X||\mathcal{R}|^3)$ obtained by the *RAF* algorithm in Hordijk and Steel (2004). We have demonstrated the use of the algorithm for several different models, and for the Kauffman model empirical results similar to the ones obtained in Hordijk and Steel (2004) indicate an average running time of $\mathcal{O}(|\mathcal{R}|^{1.342})$. Due to cyclic structures reference counting on the closure of F was not without difficulties; some parts that are no longer F -generated would still have a non-zero count. This is similar to the problem encountered using reference counting for garbage collection which could be explored further. Only, since the empirical results obtained are not as good as one initially could have hoped for, it is rather doubtful if this approach would improve anything.

The results in the simulation study indicate that it does not matter whether we consider the Kauffman model or the Random sequence based model, with a probability of catalysation, that not depends on either the catalyst or the reaction.

Furthermore the Template based model does not seem to differ that much from the Random sequence based model, as regards to the probability of finding an *RAF* set, at least not for $k = 2$, although the probability is a bit smaller in the Template based model. It is difficult to come up with an explanation on why that is the case, it would require further investigation of the model. It could be worth examining the case where $k = 4$ (RNA-like molecules), where the assumptions in the Template model about base-pair complementarities, may have a larger influence on the probability of finding an *RAF* set. The theorems in Section 5 imply that the expected number of reactions each molecule catalyses, which is needed for the existence of an *RAF* set, is not unnaturally high, and vice versa for non-existence of an *RAF* set. From the simulation of the Template based model we could see that the transition value is still small and in roughly linear growth with n , i.e. this model is worth considering in the question of the origin of life. Hence we can see the theorems are still far from tight. They can be improved if we could give a better estimation of d_n to replace Lemma 5.5.

In contrast to the Template based model, the Thermodynamic model is a lot different from the Random sequence based model. As expected, the value of b has a great influence on the probability of finding an *RAF* set. Specially for the two larger values of b that we have examined ($b = 2.0$ and 2.4) it seems like the growth of catalysation needed for the occurrence of *RAF* sets, may be faster than linear, which is not good, if we want the model to describe the origin of life. The conclusion rely on few observations, so an idea could be to try to examine this model for larger n , to give a more accurate description on how the transition grows, and for which values of b , the amount of catalysation needed is still small enough, for the model

to be considered as realistic. In order to delete parameters in the Thermodynamic model, we had a very long formula to calculate a from b . This idea of having average catalysation probability p is natural, but there may be other ways to delete parameters to avoid useless calculations. Then afterwards corresponding results similar to above might be found.

Another model to look at is to assume that molecules are represented by sequences of chemical groups, but where reactions would transfer a coherent group between two reactants, and then see if similar results can be obtained.

In our Template-based model we only studied the existence and relative size of *RAF* sets. We may also want to look at the pattern of *RAFs*, if any, i.e. we are also interested in what these *RAFs* look like under the base-pair complementarity condition. In a model with a more general k , e.g. $k = 4$, it could be interesting to examine if the molecules in the *RAF* sets only contain some of the base-pairs.

When adding molecules inhibition of reactions to the molecule, the problem of finding *RAFs* is proven NP complete in Mossel and Steel (2005). This does not necessarily the story, though. It is possible to develop heuristic approaches that find a solution for many instances of the problem.

A very naïve approach is simply to run the *RAF* algorithm ignoring inhibitions-if no *RAF* is found then there is no *RAF* when inhibitions are included, and if a *RAF* is found where none of the F -generated molecules inhibit any of the reactions then this proves the existence of a *RAF*. One can expand on this idea or develop novel approaches that will allow the question of the presence of *RAFs* to be settled for the most random systems. As long as the set of systems where no decision is reached is small, this will still allow a simulation based investigation into the likelihood of the emergence of *RAFs* under different parameter values.

A different approach to developing heuristic methods is to identify conditions under which it is no longer NP hard to find *RAFs* in the presence of inhibitions. In Mossel and Steel (2005) no assumptions were made about the nature of inhibitions. However, one can explore the effect of several natural restrictions of inhibitions. For example, inhibitions of reaction r by molecule x could be caused by x binding to the reactant molecules a of r . In such a case it could be natural to require that all reactions where a is a reactant are inhibited by x . Alternatively, the inhibition could be a simple matter of kinetics. If r and r are two reactions utilizing the same reactant molecule x but r running much faster than r then inhibition could be a case of r depleting x . Hence it could be more a case of r inhibiting r , i.e. molecules could only inhibit neighbouring reactions.

Acknowledgements

We wish to thank Jotun Hein and Rune Lyngsø for their valuable guidance and discussions throughout the project. A special thanks goes out to Adam Novak for helpful programming assistance. Some of us were supported by Associate Professor Asger Hobolth from Department of Mathematical Sciences at Aarhus University, and for that we are grateful. Finally we would like to thank the rest of the summer school students for giving us an instructive and wonderful summer.

Appendix

Implementation 1

```
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Deque;
import java.util.List;

public class RafAlgorithm {

    private int [] s;
    private int [] c;
    private Deque<Integer> uncatReactions;
    private List<List<Integer>> catReactions2;
    private boolean [] reactionR;
    private boolean [] inUncat;

    private void metode(int x) {
        s[x]--;
        if(s[x]==0)
            for(int reac = 0; reac < catReactions2.get(x).size(); reac++) {
                int reac2 = catReactions2.get(x).get(reac);
                if(reactionR[reac2]==true) {
                    catReactions2.get(x).remove(reac);
                    reac--;
                }
                else {
                    c[reac2]--;
                    if(c[reac2]==0 && inUncat[reac2]==false) {
                        inUncat[reac2]=true;
                        uncatReactions.add(reac2);
                    }
                }
            }
    }

    public ReactionSystem start(ReactionSystem input) {

        List<List<Integer>> reactants = input.reactants;
        List<List<Integer>> products = input.products;
        List<List<Integer>> catReactions = input.catReactions;
        int numberMolecules = input.molecules.size();
        List<Integer> foodset = input.foodset;

        s = new int[numberMolecules];

        for(List<Integer> list : reactants)
            for(int x : list)
                s[x]++;
        for(List<Integer> list : products)
            for(int x : list)
                s[x]++;

        c = new int[reactants.size()];
        inUncat = new boolean[c.length];

        for(int cat = 0; cat < catReactions.size(); cat++)
            if(s[cat] > 0)
                for(int reac : catReactions.get(cat))
                    c[reac]++;

        uncatReactions = new ArrayDeque<Integer>();

        for(int r = 0; r < c.length; r++)
            if(c[r] == 0) {
                inUncat[r]=true;
                uncatReactions.add(r);
            }

        catReactions2 = new ArrayList<List<Integer>>();

        for(List<Integer> list : catReactions)
            catReactions2.add(new ArrayList<Integer>(list));

        reactionR = new boolean[c.length];

        do {
            while(uncatReactions.size() > 0) {
                int reacToRemove = uncatReactions.pollFirst();
                reactionR[reacToRemove] = true;
                for(int x : reactants.get(reacToRemove))
                    metode(x);
                for(int x : products.get(reacToRemove))
                    metode(x);
            }

            Deque<Integer> nset = new ArrayDeque<Integer>(foodset);
            boolean notFgenReactions [] = new boolean[c.length];
            int r [] = new int[c.length];
```

```

boolean prodMol[] = new boolean[numberMolecules];

for(int i : foodset)
    prodMol[i]=true;

while(nset.size() > 0) {
    int comToRemove = nset.pollFirst();
    for(int reac = 0; reac < reactionR.length; reac++)
        if(reactionR[reac] == false && reactants.get(reac).contains(comToRemove)) {
            r[reac]++;
            if(r[reac]==reactants.get(reac).size()) {
                notFgenReactions[reac]=true;
                for(int prod : products.get(reac))
                    if(prodMol[prod]==false) {
                        prodMol[prod]=true;
                        nset.add(prod);
                    }
            }
        }
}

for(int i = 0; i < c.length; i++)
    if(reactionR[i]==false && notFgenReactions[i]==false && inUncat[i]==false) {
        inUncat[i]=true;
        uncatReactions.add(i);
    }
} while(uncatReactions.size()>0);

List<List<Integer>> reactantsNew = new ArrayList<List<Integer>>();
List<List<Integer>> productsNew = new ArrayList<List<Integer>>();
List<List<Integer>> catReactionsNew = new ArrayList<List<Integer>>();
List<String> moleculesNew = new ArrayList<String>();
List<Integer> foodsetNew = new ArrayList<Integer>();

int renumberR[] = new int[reactionR.length];
int renumberM[] = new int[numberMolecules];

for(int j = 0; j < s.length; j++)
    if(s[j]>0) {
        renumberM[j]=moleculesNew.size();
        moleculesNew.add(input.molecules.get(j));
    } else {
        renumberM[j]=-1;
    }

for(int j : foodset)
    if(s[j]>0)
        foodsetNew.add(renumberM[j]);

for(int i = 0; i < reactionR.length; i++)
    if(reactionR[i]==false) {
        renumberR[i]=reactantsNew.size();
        List<Integer> internalReactants = new ArrayList<Integer>();
        List<Integer> internalProducts = new ArrayList<Integer>();
        for(int k : reactants.get(i))
            internalReactants.add(renumberM[k]);
        reactantsNew.add(internalReactants);
        for(int l : products.get(i))
            internalProducts.add(renumberM[l]);
        productsNew.add(internalProducts);
    }

for(int n = 0; n < numberMolecules; n++)
    if(s[n] > 0) {
        List<Integer> internalCat = new ArrayList<Integer>();
        for(int m : catReactions2.get(n))
            if(reactionR[m]==false)
                internalCat.add(renumberR[m]);
        catReactionsNew.add(internalCat);
    }

ReactionSystem sys = new ReactionSystem();
sys.molecules = moleculesNew;
sys.foodset = foodsetNew;
sys.reactants = reactantsNew;
sys.products = productsNew;
sys.catReactions = catReactionsNew;
return sys;
}
}

```

Implementation 2

```
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Deque;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;

public class RafAlgorithm2 {

    private int [] s;
    private int [] c;
    private Deque<Integer> uncatReactions;
    private List<List<Integer>> catReactions2;
    private boolean [] reactionR;
    private boolean [] inUncat;

    private void metode(int x) {
        ListIterator<Integer> listIt = catReactions2.get(x).listIterator();
        s[x]--;
        if (s[x]==0)
            while (listIt.hasNext()) {
                int o = listIt.next();
                if (reactionR[o] == true)
                    listIt.remove();
                else {
                    c[o]--;
                    if (c[o] == 0 && inUncat[o]==false) {
                        inUncat[o]=true;
                        uncatReactions.add(o);
                    }
                }
            }
    }

    @SuppressWarnings("unchecked")
    public ReactionSystem start(ReactionSystem input) {

        List<List<Integer>> reactants = input.reactants;
        List<List<Integer>> products = input.products;
        List<List<Integer>> catReactions = input.catReactions;
        int numberMolecules = input.molecules.size();
        List<Integer> foodset = input.foodset;

        List<Integer> reactants2 [] = (ArrayList<Integer>[])new ArrayList[numberMolecules];

        for(int i = 0; i < reactants2.length; i++) {
            reactants2[i] = new ArrayList<Integer>();
        }

        for(int i = 0; i < reactants.size(); i++)
            for(int j = 0; j < reactants.get(i).size(); j++)
                reactants2[reactants.get(i).get(j)].add(i);

        s = new int [numberMolecules];

        for(List<Integer> list : reactants)
            for(int x : list)
                s[x]++;
        for(List<Integer> list : products)
            for(int x : list)
                s[x]++;

        c = new int [reactants.size()];
        inUncat = new boolean [c.length];

        for(int cat = 0; cat < catReactions.size(); cat++)
            if (s[cat] > 0)
                for(int reac : catReactions.get(cat))
                    c[reac]++;

        uncatReactions = new ArrayDeque<Integer>();

        for(int r = 0; r < c.length; r++)
            if (c[r] == 0) {
                inUncat[r]=true;
                uncatReactions.add(r);
            }

        catReactions2 = new LinkedList<List<Integer>>();

        for(List<Integer> list : catReactions)
            catReactions2.add(new LinkedList<Integer>(list));

        reactionR = new boolean [c.length];

        do {
            while(uncatReactions.size() > 0) {
                int reacToRemove = uncatReactions.pollFirst();
                reactionR[reacToRemove] = true;
                for(int x : reactants.get(reacToRemove))
                    metode(x);
            }
        }
    }
}
```

```

        for(int x : products.get(reactToRemove))
            metode(x);
    }

    Deque<Integer> nset = new ArrayDeque<Integer>(foodset);
    boolean notFgenReactions[] = new boolean[c.length];
    int r[] = new int[c.length];
    boolean prodMol[] = new boolean[numberMolecules];

    for(int i : foodset)
        prodMol[i]=true;

    while(nset.size() > 0) {
        int comToRemove = nset.pollFirst();
        for(int reac : reactants2[comToRemove])
            if(reactionR[reac] == false) {
                r[reac]++;
                if(r[reac]==reactants.get(reac).size()) {
                    notFgenReactions[reac]=true;
                    for(int prod : products.get(reac))
                        if(prodMol[prod]==false) {
                            prodMol[prod]=true;
                            nset.add(prod);
                        }
                }
            }
    }

    for(int i = 0; i < c.length; i++)
        if(reactionR[i]==false && notFgenReactions[i]==false && inUncat[i]==false) {
            inUncat[i]=true;
            uncatReactions.add(i);
        }

    } while(uncatReactions.size()>0);

    List<List<Integer>> reactantsNew = new ArrayList<List<Integer>>();
    List<List<Integer>> productsNew = new ArrayList<List<Integer>>();
    List<List<Integer>> catReactionsNew = new ArrayList<List<Integer>>();
    List<String> moleculesNew = new ArrayList<String>();
    List<Integer> foodsetNew = new ArrayList<Integer>();

    int renumberR[] = new int[reactionR.length];
    int renumberM[] = new int[numberMolecules];

    for(int j = 0; j < s.length; j++)
        if(s[j]>0) {
            renumberM[j]=moleculesNew.size();
            moleculesNew.add(input.molecules.get(j));
        } else {
            renumberM[j]=-1;
        }

    for(int j : foodset)
        if(s[j]>0)
            foodsetNew.add(renumberM[j]);

    for(int i = 0; i < reactionR.length; i++)
        if(reactionR[i]==false) {
            renumberR[i]=reactantsNew.size();
            List<Integer> internalReactants = new ArrayList<Integer>();
            List<Integer> internalProducts = new ArrayList<Integer>();
            for(int k : reactants.get(i))
                internalReactants.add(renumberM[k]);
            reactantsNew.add(internalReactants);
            for(int l : products.get(i))
                internalProducts.add(renumberM[l]);
            productsNew.add(internalProducts);
        }

    for(int n = 0; n < numberMolecules; n++)
        if(s[n] > 0) {
            List<Integer> internalCat = new ArrayList<Integer>();
            for(int m : catReactions2.get(n))
                if(reactionR[m]==false)
                    internalCat.add(renumberR[m]);
            catReactionsNew.add(internalCat);
        }

    ReactionSystem sys = new ReactionSystem();
    sys.molecules = moleculesNew;
    sys.foodset = foodsetNew;
    sys.reactants = reactantsNew;
    sys.products = productsNew;
    sys.catReactions = catReactionsNew;
    return sys;
}
}

```

Implementation 3

```
import java.util.ArrayDeque;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Deque;
import java.util.LinkedList;
import java.util.List;
import java.util.ListIterator;

public class RafAlgorithm3 {

    private int [] t;
    private int [] s;
    private int [] c;
    private Deque<Integer> uncatReactions;
    private List<List<Integer>> catReactions2;
    private boolean [] reactionR;
    private boolean [] inUncat;

    private void metode(int x) {
        ListIterator<Integer> listIt = catReactions2.get(x).listIterator();
        s[x]--;
        if (s[x]==0)
            while (listIt.hasNext()) {
                int o = listIt.next();
                if (reactionR[o] == true)
                    listIt.remove();
                else {
                    c[o]--;
                    if (c[o] == 0 && inUncat[o]==false) {
                        inUncat[o]=true;
                        uncatReactions.add(o);
                    }
                }
            }
    }

    @SuppressWarnings("unchecked")
    public ReactionSystem start(ReactionSystem input) {

        List<List<Integer>> reactants = input.reactants;
        List<List<Integer>> products = input.products;
        List<List<Integer>> catReactions = input.catReactions;
        int numberMolecules = input.molecules.size();
        List<Integer> foodset = input.foodset;

        List<Integer> reactants2 [] = (ArrayList<Integer>[])new ArrayList [numberMolecules];

        for (int i = 0; i < reactants2.length; i++) {
            reactants2[i] = new ArrayList<Integer>();
        }

        for (int i = 0; i < reactants.size(); i++)
            for (int j = 0; j < reactants.get(i).size(); j++)
                reactants2[reactants.get(i).get(j)].add(i);

        s = new int [numberMolecules];

        for (List<Integer> list : reactants)
            for (int x : list)
                s[x]++;
        for (List<Integer> list : products)
            for (int x : list)
                s[x]++;

        c = new int [reactants.size()];
        inUncat = new boolean [c.length];

        for (int cat = 0; cat < catReactions.size(); cat++)
            if (s[cat] > 0)
                for (int reac : catReactions.get(cat))
                    c[reac]++;

        uncatReactions = new ArrayDeque<Integer>();

        for (int r = 0; r < c.length; r++)
            if (c[r] == 0) {
                inUncat[r]=true;
                uncatReactions.add(r);
            }

        catReactions2 = new LinkedList<List<Integer>>();

        for (List<Integer> list : catReactions)
            catReactions2.add(new LinkedList<Integer>(list));

        reactionR = new boolean [c.length];

        t = new int [numberMolecules];
        for (int i = 0; i < foodset.size(); i++)
            t[foodset.get(i)]++;
        for (int reac = 0; reac < products.size(); reac++)
            for (int mol = 0; mol < products.get(reac).size(); mol++)
```

```

        t[products.get(react).get(mol)]++;
do {
    while(uncatReactions.size() > 0) {
        int reacToRemove = uncatReactions.pollFirst();
        reactionR[reacToRemove] = true;
        for(int x : reactants.get(reacToRemove))
            metode(x);
        for(int x : products.get(reacToRemove))
            metode(x);
        for(int x : products.get(reacToRemove))
            if(t[x] > 0) {
                t[x]--;
                if(t[x]==0) {
                    for(int reac = 0; reac < reactants2[x].size(); reac++)
                        if(inUncat[reactants2[x].get(reac)]==false) {
                            inUncat[reactants2[x].get(reac)]=true;
                            uncatReactions.add(reactants2[x].get(reac));
                        }
                }
            }
    }

    Deque<Integer> nset = new ArrayDeque<Integer>(foodset);
    boolean notFgenReactions[] = new boolean[c.length];
    int r[] = new int[c.length];
    boolean prodMol[] = new boolean[numberMolecules];

    for(int i : foodset)
        prodMol[i]=true;

    while(nset.size() > 0) {
        int comToRemove = nset.pollFirst();
        for(int reac : reactants2[comToRemove])
            if(reactionR[reac] == false) {
                r[reac]++;
                if(r[reac]==reactants.get(reac).size()) {
                    notFgenReactions[reac]=true;
                    for(int prod : products.get(reac))
                        if(prodMol[prod]==false) {
                            prodMol[prod]=true;
                            nset.add(prod);
                        }
                }
            }
    }

    for(int i = 0; i < c.length; i++)
        if(reactionR[i]==false && notFgenReactions[i]==false && inUncat[i]==false) {
            inUncat[i]=true;
            uncatReactions.add(i);
        }
} while(uncatReactions.size()>0);

List<List<Integer>> reactantsNew = new ArrayList<List<Integer>>();
List<List<Integer>> productsNew = new ArrayList<List<Integer>>();
List<List<Integer>> catReactionsNew = new ArrayList<List<Integer>>();
List<String> moleculesNew = new ArrayList<String>();
List<Integer> foodsetNew = new ArrayList<Integer>();

int renumberR[] = new int[reactionR.length];
int renumberM[] = new int[numberMolecules];

for(int j = 0; j < s.length; j++)
    if(s[j]>0) {
        renumberM[j]=moleculesNew.size();
        moleculesNew.add(input.molecules.get(j));
    } else {
        renumberM[j]=-1;
    }

for(int j : foodset)
    if(s[j]>0)
        foodsetNew.add(renumberM[j]);

for(int i = 0; i < reactionR.length; i++)
    if(reactionR[i]==false) {
        renumberR[i]=reactantsNew.size();
        List<Integer> internalReactants = new ArrayList<Integer>();
        List<Integer> internalProducts = new ArrayList<Integer>();
        for(int k : reactants.get(i))
            internalReactants.add(renumberM[k]);
        reactantsNew.add(internalReactants);
        for(int l : products.get(i))
            internalProducts.add(renumberM[l]);
        productsNew.add(internalProducts);
    }

for(int n = 0; n < numberMolecules; n++)
    if(s[n] > 0) {
        List<Integer> internalCat = new ArrayList<Integer>();
        for(int m : catReactions2.get(n))
            if(reactionR[m]==false)
                internalCat.add(renumberR[m]);
    }

```

```
        catReactionsNew.add(internalCat);
    }

    ReactionSystem sys = new ReactionSystem();
    sys.molecules = moleculesNew;
    sys.foodset = foodsetNew;
    sys.reactants = reactantsNew;
    sys.products = productsNew;
    sys.catReactions = catReactionsNew;
    return sys;
}
}
```

References

- [1] Stuart A. Kauffman. Autocatalytic sets of proteins. *Journal of Theoretical Biology*, 119(1):1-24, 1986.
- [2] Mike Steel. The emergence of a self-catalysing structure in abstract origin-of-life models. *Applied Mathematics Letters*, 13(3):91-95, 2000.
- [3] Wim Hordijk and Mike Steel. Detecting autocatalytic, self-sustaining sets in chemical reaction systems. *Journal of Theoretical Biology*, 227(4):451-461, 2004.
- [4] Elchanan Mossel and Mike Steel. Random biochemical networks: the probability of self-sustaining autocatalysis. *Journal of Theoretical Biology*, 233(3):327-336, 2005.
- [5] Wim Hordijk, Jotun Hein and Mike Steel. Autocatalytic sets and the origin of life. *Entropy*, 2010.
- [6] Jotun Hein. Autocatalytic sets of self-replicating RNAs. http://www.stats.ox.ac.uk/__data/assets/pdf_file/0004/5377/Autocatalytic_sets_of_RNAs.pdf, 2010.
- [7] Jotun Hein and Rune Lyngsø. Search for Life in Catalytic Reaction Systems. *Project description for summer studentship*, 2010.