In: Proc. of CSL'97

Evolution as a computational engine*

Rolf Backofen and Peter Clote

Institut für Informatik
Ludwig-Maximilians-Universität München
Oettingenstraße 67, D-80538 München, Germany.
{backofen,clote}@informatik.uni-muenchen.de

Abstract. Given a finite set K of *lethal genes*, a starting genome x and a desired *target* genome y, is there a sequence of base insertions, deletions, and substitutions, which from x produces the desired genome y, and such that no intermediate genome contains a lethal gene. The main result of this paper is that, appropriately formulated, this problem is undecidable, even when the underlying alphabet has only 2 symbols (e.g. two of the bases A,C,G,T). We prove that asexual evolutionary systems form a universal computation engine, in the sense that for any reversible Turing machine M and input x, there is a finite set K of lethal genes, which can guide the computation, so that all intermediate genomes are prefixes of (encodings of) configurations in the computation of M. Since reversible Turing machines retain the history of their entire computation, and are essentially encoded by evolutionary systems, this suggests a encoding mechanism by which *phylogeny recapitulates ontogeny*.

1 Introduction

This paper studies the question of how computations can be executed in biological systems by using asexual evolution with pointwise mutations (as contrasted with sexual evolution with crossover). Our work was motivated from three directions: mathematical evolution theory, biocomputing and genetic algorithms.

Concerning evolution, it has been suggested that species in a closed environmental system are strange attractors, in the sense of mathematical dynamics. Can this remark somehow be formalized? One might try to define an evolution operator $E(x, y) = x'$, which, from the ecosystem x[1] and environment y yields the ecosystem of the next generation. Assume that u and v are genomes (i.e. words in $\{A, C, G, T\}^*$) whose Levenshtein distance[2] or other dissimilarity measure is small. Do both u and v have approximately the same number of descendents in each generation, when applying mutation and survival of the fittest under specified environmental conditions? Are the phylogeny trees below u and below v structurally similar? Is the evolution operator E continuous? What are the trajectories of E's iterates?

* Research supported by NSF CCR-9408090 and US-Czech Science and Technology Grant 93-205.

[1] An ecosystem is a collection of species, where species is a probability distribution on a gene pool.

[2] Levenshtein distance between sequences $u = u_1 \cdots u_n$ and $v = v_1 \cdots v_m$ is defined as $n+m-2\ell$, where ℓ is the length of the longest common subsequence of u, v; i.e. sequence alignment distance between u, v, with penalty 1 for a deletion or insertion and 2 for a mismatch. See chapter 9 of [14] for more on sequence alignment. See [4] for the estimated Levenshtein distance between two random sequences.

Our model of asexual evolution is admittedly a heretical biological fiction – it is known that certain genes are indispensible for the viability of an organism, but there is no evidence of lethal or harmful genes (indeed our use of the phrase 'lethal gene' is an oxymoron). Nevertheless, the simplicity of our model allows us to give a precise, mathematical formulation of the above mentioned questions which motivate our work. Our model provides us with certain interesting, highly nontrivial open problems, whose solution might help us to better understand phylogeny, genetic distance between species, and whether evolution displays chaotic behaviour, in the sense of mathematical dynamics.

Concerning biocomputing, L. Adleman [1] showed how to encode graph theoretic problems using nucleotide sequences, thus illustrating how biochemical techniques could be used to solve hard combinatorial optimization problems. Our main result proves that for every computation problem, an environment can be designed which *steers* the subsequent genome populations in a manner whereby the solution of the given computation problem will appear in the progeny. Specifically, we show that *asexual evolution*, i.e. selection in an environment allowing only pointwise mutations, forms a universal computational engine, in the sense that for any algorithm A there exists a finite set K of *lethal genes* which for any input x steers the computation of A on x. All progeny genomes, not containing any lethal gene, are exactly prefixes of (encodings of) intermediate configurations of the computation of A.

Concerning genetic algorithms, in [13] P. Pudlák introduced the model of genetic Turing machine, which roughly speaking performs a computation by performing context sensitive crossover (sexual evolution), and then applying a fitness function to an exponentially large in $p(n)$ population of computation strings. Pudlák's main result was that polynomial time genetic Turing machines compute exactly polynomial space predicates, a complexity theoretic analysis which justifies the strength of genetic and evolutionary programming methods. Independently, a similar result was obtained by Arora, Rabani and Vazirani [2]. By removing time and size constraints, Pudlák's construction implicitly yields a result of the form that nature, using sexual evolution with crossover and a fitness function, can simulate a universal computational device. Our main theorem, which does not follow from the work of [13, 2], shows an analogous result for asexual evolution. It is clear that one can characterize certain complexity classes by a resource bounded version of our evolutionary systems, but we have not investigated that issue here.

It should be mentioned that in [9, 10, 11] S. Yu. Maslov[3] defined and studied various *mutation calculi*, whose rules allowed context dependent insertion, deletion and substitution as well as possible polyploidy (duplication of words). Maslov obtained results concerning the decidability and undecidability of various calculi. Our main result, though related, does not directly follow from Maslov's work, since our main proof consists of defining an appropriate finite set K of lethal genes and showing how K can indirectly act as a context for context dependent insertion, substitution and substitution. Maslov's work appears not to be well-known in the DNA computing community. Recently, J. Dassow, V. Mitrana [5], L. Ilie, V. Mitrana [6] and G. Paun, A. Salomaa [12] have reported on various applications of formal language theory to computational biology (e.g. [5] concerns *evolutionary grammars*, which allow context sensitive insertion, deletion, and substitution rules). The main result of [5] is an immediate consequence of [9]. None of this work cannot be used to deduce our main results.

The results of this paper were obtained before the authors had knowledge of [7, 8]; an anonymous referee pointed out these articles and asked us to comment on

[3] Thanks to G. Mints, who brought Maslov's work to our attention.

the relation with our work. We were unable to obtain a copy of [8], so reserve our comments to [7]. There, a study is made of closure properties of the class of regular, context-free, context-sensitive and recursively enumerable languages under context-determined insertion and deletion operators. The only point of comparison with our work is Proposition 6.1 of [7]. We begin by some notation. The (x, y) contextual insertion of v in u, denoted $u \leftarrow_{(x,y)} v$, means that if word $u = u_1xyu_2$, where $u_1, u_2, x, y, v \in \Sigma$, then u is replaced by u_1xvyu_2. The (x, y) context deletion of v in u, denoted $u \rightarrow_{(x,y)} v$, means that if word $u = u_1xvyu_2$, where $u_1, u_2, x, y, v \in \Sigma$, then u is replaced by u_1xyu_2. Note that v, the string which is inserted or deleted, is a word, rather than a symbol of Σ. An indel system ID is stipulated by a finite set of such contextual insertion and deletion rules. Proposition 6.1 of [7] now states that if a language L is accepted by a Turing machine, then there exists an indel system ID which accepts the same language.

The proof of Proposition 6.1 is straightforward; and in fact can be simplified if one works with semi-Thue systems in place of Turing machines. Semi-Thue systems, or rewriting systems, have contextual substitution rules. The idea of the simulation is to extend the alphabet to allow introduction of new symbols, so that a sequence of appropriate contextual insertion and deletion rules can be used to simulate a substitution rule. Since the alphabet is finite, and one is inserting and deleting finite words (rather than a bit from the alphabet $\{0, 1\}$, as in our work), and since one applies a rule, provided that a context is satisfied (rather than being allowed to apply a rule, if no killer gene results), the result is technically much easier. It is clear that the work of [7] does not imply our main result. As well, our result concerns evolutionary systems, which allow symmetric rules; i.e. in a context if one can delete a bit, then that bit can be added back (this is reminiscent of lambda-calculus). This is different from the directionality of rewriting rules, and from the indel rules of [7]. Prima faciae, our work does not directly imply Proposition 6.1 of [7], since we work exclusively over the alphabet $\{0, 1\}$. It is clear, however, that our results extend immediately to any finite alphabet of cardinality at least 2, and it seems then possible that a derivation of Proposition 6.1 of [7] could be given. However, this would require a careful proof, indicating how exactly from a set K of lethal genes, one defines a set of indel rules. Certainly such a derivation would be much more complicated than the straightforward proof of Proposition 6.1 given in [7].

An important difference between our work and the work cited above is that we model a formal system of evolution, where the application of rules (for substitution, insertion and deletion) are controlled by a uniform "environment". Thus, the "environment" provides a mechanism for specifying the context of a rule application (in form of lethal genes) that is the same for every rule application. In the above cited work, the context of a valid rule application is attached to a rule. This implies that previous systems do not model any form of uniform "environment".

In our work, we study reversible systems, i.e., systems which, for every rule contain a symmetric rule. We choose this because we want to model (asexual) evolution, where mutation, insertion or deletion are also reversible operations. This is another difference with [7] and [5], which allow systems where some (or all) rules do not have symmetric counterparts. For this reason, we simulate Thue systems in proving undecidability instead of semi-Thue systems. Since semi-Thue systems have the advantage that one can simulate directly Turing machines (which are usually used for defining complexity classes), we consider the notion of reversible Turing machines in the appendix. The reversible Turing machines can be simulated by Thue system instead of semi-Thue system. This concludes our discussion of related work.

2 Statement of result

We wish to formalize the problem of asexual evolution in the presence of a finite set of lethal genes. For words u, v in finite alphabet Σ, write $u \subseteq v$ to mean that u is a convex subword of v, and write $u \Rrightarrow v$ if v can be obtained from u by the deletion, insertion, or substitution of one symbol. Write $u \Rrightarrow_r v$ if insertion and deletion can only occur at the end of u.

Consider the problem, given $x, y \in \Sigma^*$ and finite set $K \subset \Sigma^*$, whether there exists a sequence

$$x = z_0 \Rrightarrow z_1 \Rrightarrow \cdots \Rrightarrow z_n = y$$

for which w is not a convex subword of z_i for all $i \leq n$ and $w \in K$. This problem is trivially decidable, since there is such a sequence if and only if no lethal gene $w \in K$ is a convex subword of either x or y.[4] There are various manners to remedy this trivial formulation of the problem. In a biological setting, special proteins bind to ends of chromosomes to help prevent unravelling, so it seems reasonable to allow special *end markers* which are not part of the alphabet Σ. This is incorporated in the following definition.

Definition 2.1. *Let Σ be a finite alphabet, consisting of at least 2 symbols, and $\$, ¢ \notin \Sigma$. Let $x, y \in \$\Sigma^*¢$ and K be a finite subset of $(\Sigma \cup \{\$, ¢\})^*$. Write $x \Rrightarrow^*_{\Sigma,K} y$ to denote that for some natural number n*

$$x = x_0 \Rrightarrow x_1 \Rrightarrow \cdots \Rrightarrow x_n = y$$

and for all $i \leq n$ and $w \in K$, $w \not\subseteq x_i$. As well, write $x \Rrightarrow^*_{\Sigma,K,r} y$, when insertions and deletions can only occur at the right extremity. A tuple $E = (\Sigma, K)$ is called an evolutionary system, and the elements of K are called lethal genes. Given $x \in \Sigma^*$ and a set of lethal genes K, we say that a lethal gene $w \in K$ applies iff $w \subseteq x$.

Definition 2.2 (Future Genome Problem). *Fix finite alphabet Σ, $|\Sigma| \geq 2$, and $\$, ¢ \notin \Sigma$, and finite subset $K \subseteq (\Sigma \cup \{\$, ¢\})^*$. Given $x, y \in \$\Sigma^*¢$, does $x \Rrightarrow^*_{\Sigma,K} y$?*

Theorem 2.3. *For every Σ with $|\Sigma| \geq 2$, there exists finite K for which the Future Genome Problem is undecidable.*

The next section proves this theorem. We first introduce some preliminaries, before we consider substitution systems, which are a variant of Thue-systems more appropriate for our purposes. We then present an overview over the different simulation steps that are decribed in detail in the subsequent sections.

3 Thue Systems and Evolutionary Systems

For the remainder of this paper, Σ will denote a finite alphabet with $|\Sigma| \geq 2$, and $\$, ¢ \notin \Sigma$. For words $x, y \in \Sigma^*$, write xy to denote the concatenation of x, y. The empty word is denoted by λ. The word x is a prefix [resp. suffix] of the word y if $y = xu$ [resp. $y = ux$] for some word u. The word x is a convex subword of word y, denoted by $x \subseteq y$, if $y = uxv$ for words u, v. The length of word x is denoted by

[4] Personal communication, Jan Krajíček.

$|x|$; e.g. $|x_1 \cdots x_n| = n$, if $x_i \in \Sigma$. If $a \in \Sigma$ and $x \in \Sigma^*$, then $|x|_a$ is the number of occurrences of symbol a in x.

Given a word $u \in \Sigma$ and a number $1 \leq i \leq |u|$, we define u_i to be the symbol at position i of u. For $1 \leq i \leq j \leq |u|$, we write u_i^j to denote the substring of u between positions i and j (including i, j). For convenience, we define $u_i^j := u_1^j$ for $i < 1$, $u_i^j := u_i^{|u|}$ for $j > |u|$ and u_i^j to be the empty string for $j < 1$, $i > |u|$ or $i > j$.

As usual, we define a semi-Thue system to be a binary relation over Σ^*, and a Thue system G to be a binary, symmetric relation over Σ^*. We write $x \Rightarrow_{(u,v)} y$ if $x = wuw'$ and $y = wvw'$, and we define $\Rightarrow_{\{u,v\}}$ as the symmetric closure of $\Rightarrow_{(u,v)}$. We write $x \Rightarrow_G y$ if $x \Rightarrow_{(u,v)} y$ for some $(u, v) \in G$. A derivation in G (or G-derivation) is a sequence $x_0 \Rightarrow_{r_1} x_1, \cdots, x_{n-1} \Rightarrow_{r_n} x_n$ such that $\{r_1, \ldots, r_n\} \subseteq G$.

Definition 3.1 (Substitution System). *A substitution system P is a triple (Σ, δ, R) consisting of an alphabet Σ, a designated element $\delta \in \Sigma$ and a finite set of rules $R \subseteq \Sigma^* \times \Sigma^*$ of the form $\{u, v\}$ with $|u| = |v|$. The associated Thue system G_P is a Thue system over $(\Sigma \cup \{\$, ¢\})^*$ defined by*

$$G_P := R \cup \{\{¢, \delta¢\}\}.$$

We define $\Rightarrow_P$ to be $\Rightarrow_{G_P}$. For every $r = \{u, v\} \in R$ we define the length of r (written $\mathrm{len}(r)$) to be $|u|$. A substitution system is called simple if $\{u, v\} \in R$ implies that u and v differ in at most one symbol.

Lemma 3.2. *Every Thue system G over Σ^* can be simulated by a substitution system $P = (\Sigma \cup \{\delta\}, \delta, R)$ such that for all $x, y \in \Sigma^*$, $x \Rightarrow_G^* y$ if and only if $\$x¢ \Rightarrow_P^* \$y¢$.*

Proof. Let G be a fixed Thue system over Σ^*. Construct a substitution system P as follows. We write δ^k as an abbreviation for the word of length k consisting only of δ's. The set R is defined as

$$\begin{aligned} R &:= R_{trans}(G) \cup R_{exch}(G) \\ R_{trans}(G) &:= \{\{u, v\delta^k\} \mid \{u, v\} \in G \wedge |u| \geq |v| \wedge k = |u| - |v|\} \\ R_{exch}(G) &:= \{\{\delta\sigma, \sigma\delta\} \mid \sigma \in \Sigma\} \end{aligned} \tag{1}$$

The rules in (1) are called exchange rules. Note well that for different rules r, r', it may be that $\mathrm{len}(r) \neq \mathrm{len}(r')$.

Then for $x, y \in \Sigma^*$, we have $x \Rightarrow_G^* y$ if and only if $\$x¢ \Rightarrow_P^* \$y¢$:

$\rightarrow$ We simulate a rule application $z \Rightarrow_{(u,v)} z'$ with $(u, v) \in G$ and $|u| \geq |v|$ by a sequence of rule applications in P of the form

$$\$z¢ = z_0 \Rightarrow_{r_1} z_1 \ldots z_{n-1} \Rightarrow_{r_n} z_n = \$z'¢,$$

where r_1 is the rule $(u, v\delta^k) \in R_{trans}(G)$ with $k = |u| - |v|$, $r_2, \ldots, r_{n-k}$ are applications of the exchange rules in (1) transporting the δ's introduced by the application of r_1 to the right end, and $r_{n-k+1} \ldots r_n$ are applications of the right deletion/insertion rule $\{¢, \delta¢\}$ successively deleting the introduced δs. A rule application $z \Rightarrow_{(v,u)} z'$ with $(v, u) \in G$ and $|v| < |u|$ is simulated analogously.

$\leftarrow$ Let

$$\$x\not\in = z_0 \Rightarrow_{r_1} z_1 \ldots z_{n-1} \Rightarrow_{r_n} z_n = \$y\not\in$$

be a derivation in P such that $x, y \in \Sigma^*$. For every word $u \in (\Sigma \cup \{\delta, \$, \not\in\})^*$, let $u/\{\delta, \$, \not\in\}$ be the word in Σ obtained by deleting all occurrences of $\delta, \$$ and $\not\in$ in u. Now for every rule $r = (u, v\delta^k) \in R_{trans}(G)$ and $x \Rightarrow_r y$ we have $x/\{\delta, \$, \not\in\} \Rightarrow_{\{u,v\}} y/\{\delta, \$, \not\in\}$. For every $r \notin R_{trans}(G)$ and $x \Rightarrow_r y$ we have $x/\{\delta, \$, \not\in\} = y/\{\delta, \$, \not\in\}$. Hence, we can transform every derivation sequence in P into a valid sequence in G by deleting all δ's and all exchange and deletion/insertion rules, and by transforming every rule $R_{trans}(G)$ into a corresponding G rule.

□

In the following sections, we show that one can simulate arbitrary substitution systems P by evolutionary systems. The simulation is done by performing the following steps:

1. A substitution system $P = (\Sigma, \delta, R)$ is translated into a simple substitution system $P' = (\Sigma', \delta, R')$. The translation guarantees that every derivation in P can be simulated in P', and specific derivations in P' satisfying a given order can be translated back into P.
2. The simple substitution system P' is then simulated by a restricted evolutionary system $E_{restr} = (\Sigma', \delta, R_S, K)$ (i.e., an evolutionary system where not all substitution rules $\{\sigma, \sigma'\}$ are allowed but only those contained in a given set R_S, and where only the symbol δ may be inserted or deleted). In this system, the application of a rule in R' is simulated by the substitution of the one symbol that is changed. The set of lethal genes K will guarantee that
   (a) the substitution of a symbol is only allowed if it is in the right context
   (b) the substitutions can be reordered in a way such that the corresponding rule applications in P' can be translated back into P.
3. E_{restr} is then translated into an evolutionary system E (with alphabet $\{0, 1\}$).

As an example for the simulation of a substitution system by a simple substitution system indicated above in step 1, suppose that R contains only the rule

$$\{ab, ef\}. \tag{2}$$

Then this rule can be simulated in P' by a sequence of rules of the form

$$ab \Rightarrow_{r_1} \sigma^{r,1}b \Rightarrow_{r_2} \sigma^{r,1}\sigma^{r,2} \Rightarrow_{r_3} e\sigma^{r,2} \Rightarrow_{r_4} ef. \tag{3}$$

Here, the $\sigma^{r,i}$ are new symbols, namely for every rule and every position in the left (or right side) of the given rule in R.

For the simulation of P' by $E_{restr} = (\Sigma', \delta, R_S, K)$ as depicted in 2, the translation used in (3) is not sufficient because of the following considerations. For the simulation of P' by E_{restr}, we have to allow the substitution of a symbol σ which occurs in the position i of a rule r by the symbol $\sigma^{r,i}$. Thus, for our example (3) we have

$$R_S = \left\{\{a, \sigma^{r,1}\},\ \{b, \sigma^{r,2}\},\ \{e, \sigma^{r,1}\},\ \{f, \sigma^{r,2}\}\right\}.$$

Without knowing the exact definition of K, we know that K *cannot* contain the two words eb (since $ebab \Rightarrow_P ebef$) and $\sigma^{r,1}b$ (since $ab \Rightarrow_{P'} \sigma^{r,1}b$). Hence,

$$eb \Rightarrow \sigma^{r,1}b \Rightarrow \sigma^{r,1}\sigma^{r,2} \Rightarrow e\sigma^{r,2} \Rightarrow ef. \tag{4}$$

must be a valid derivation in E_{restr} which is not valid in P'.

These problems can be avoided if we simulate $\{ab, ef\} \in R$ by the following sequence:

$$\begin{aligned} ab &\Rightarrow_{r_1} \sigma^{r,1}b \quad \Rightarrow_{r_2} \sigma^{r,1}\sigma^{r,2} \\ &\Rightarrow_{r_3} \tau^{r,1}\sigma^{r,2} \Rightarrow_{r_4} \tau^{r,1}\tau^{r,2} \Rightarrow_{r_5} e\tau^{r,2} \Rightarrow_{r_6} ef. \end{aligned} \tag{5}$$

where the $\tau^{r,i}$ are again new symbols. This can be simulated by $E_{restr} = (\Sigma', \delta, R_S, K)$ with (using regular expression for the definition of K):

$$R_S = \left\{\{a, \sigma^{r,1}\},\ \{b, \sigma^{r,2}\},\ \{\sigma^{r,1}, \tau^{r,1}\},\ \{\sigma^{r,2}, \tau^{r,2}\},\ \{e, \tau^{r,1}\},\ \{f, \tau^{r,2}\}\right\}.$$
$$K = \sigma^{r,1}\overline{(b \cup \sigma^{r,2})} \ \cup\ \tau^{r,1}\overline{(\sigma^{r,2} \cup \tau^{r,2})} \ \cup\ \overline{(\sigma^{r,1} \cup \tau^{r,1})}\sigma^{r,2} \ \cup\ \overline{(e \cup \tau^{r,1})}\tau^{r,2}$$

For the last simulation step, we define an equivalent rewrite system in the alphabet $\{0, 1\}$ by encoding the symbols of the previous system's alphabet in unary. In other words, if the previous rewrite alphabet has m symbols, then encode the i-th symbol by $11100^{i-1}10^{m-i}0111$. Here the initial (resp. terminal) string 1110 (resp. 0111) acts as a buffer to allow concatenation of encodings of symbols. The rules of the rewrite system are encodings of the rules of the previous system. Unlike the previous system, this rewrite system no longer has rules which change only one symbol. Since these rules are no substitutions, we have to replace each rule

$$11100^{i-1}10^{m-i}0111 \Rightarrow 11100^{j-1}10^{m-j}0111$$

(this rule transforms the encoding of the i-th symbol to the encoding of the j-th symbol, where for specificity we assume $i < j$) by the two following rules

$$\begin{aligned} 11100^{i-1}10^{m-i}0111 &\Rightarrow 11100^{i-1}10^{j-i-1}10^{m-j}0111 \\ &\Rightarrow 11100^{j-1}10^{m-j}0111. \end{aligned} \tag{6}$$

Finally, one essentially defines K to be those strings of the form $1110w0111$ where $|w|_1 \notin \{1, 2\}$, or $|w|_1 = 2$ and which don't correspond to a transformation of the type illustrated above. Additional details must be applied to ensure insertion and deletion only at the right.

3.1 Simulating Substitution Systems by Simple Substitution Systems

For the simulation of P we define a simple substitution system $P' = (\Sigma', \delta, R')$ with

$$\begin{aligned} \Sigma' &:= \Sigma \cup \Sigma_{new} \\ \Sigma_{new} &:= \bigcup_{r=\{u,v\}\in R} (\{\sigma^{r,i} \mid 1 \le i \le |u|\} \cup \{\tau^{r,i} \mid 1 \le i \le |u|\}) \\ R' &:= \{r \in R \mid \mathrm{len}(r) = 1\} \cup \bigcup_{r\in R, \mathrm{len}(r)>1} \mathrm{trans}(r) \end{aligned}$$

where the $\sigma^{r,i}$ and $\tau^{r,i}$ are new symbols not occurring in Σ, and $\mathrm{trans}(r)$ is given as follows. For $r = \{u, v\}$ with $n = |u| > 1$ (where for specifity we assume that u is lexicographical smaller than v), we define $\mathrm{trans}(r)$ by

$$\begin{aligned} &\left\{ \left\{ \begin{array}{l} \sigma^{r,1} \dots \sigma^{r,i-1}\ u_i\ u_{i+1} \dots u_n, \\ \sigma^{r,1} \dots \sigma^{r,i-1}\sigma^{r,i}u_{i+1} \dots u_n \end{array} \right\} \;\middle|\; 1 \le i \le n \right\} \\ \cup &\left\{ \left\{ \begin{array}{l} \tau^{r,1} \dots \tau^{r,i-1}\sigma^{r,i}\sigma^{r,i+1} \dots \sigma^{r,n} \\ \tau^{r,1} \dots \tau^{r,i-1}\tau^{r,i}\sigma^{r,i+1} \dots \sigma^{r,n} \end{array} \right\} \;\middle|\; 1 \le i \le n \right\} \\ \cup &\left\{ \left\{ \begin{array}{l} v_1 \dots v_{i-1}\tau^{r,i}\tau^{r,i+1} \dots \tau^{r,n} \\ v_1 \dots v_{i-1}\ v_i\ \tau^{r,i+1} \dots \tau^{r,n} \end{array} \right\} \;\middle|\; 1 \le i \le n \right\} \end{aligned}$$

Next, we define the P'-derivations that can be translated back into P-derivations. A rule application $x \Rightarrow_r y$ in P' is called initial (resp. final) if $x = uvw$ and $y = uv'w$ such that $v \in (\Sigma \cup \{\$, ¢\})^*$ (resp. $v' \in (\Sigma \cup \{\$, ¢\})^*$). We say that a rule application $x \Rightarrow_r y$ is interior if it is neither initial nor final, i.e. v and v' contain an element of Σ_{new}. Note that the deletion/insertion rule is a rule whose application is both initial and final at the same time. Given two consecutive rule applications

$$x \Rightarrow_r y \Rightarrow_{r'} z$$

in P', we say that the applications of r and r' are P-consecutive iff the following conditions hold:

1. if the application of r is final, then the application of r' is initial,
2. if the application of r is not final, then $x = uvw \wedge y = uv'w \wedge z = uv''w$ such that $r = \{v, v'\} \wedge r' = \{v', v''\}$.

Note that in the second case it is allowed that r equals r'. Then we have $v = v''$ and hence $x = z$, i.e., the second application of r is just undoing the effects of the first application of r. Furthermore, note that v, v', v'' may be words in Σ^*, rather than symbols in Σ. We say that a derivation in P' is pure if every two consecutive rule applications are P-consecutive.

Proposition 3.3. *Let P and P' be given as described. Then every derivation in P can be translated into an equivalent derivation in P' which is pure. Conversely, every pure derivation in P' where both the initial word and the final word is in $\$\Sigma^*¢$ (i.e., are words over the alphabet of P) can be translated into an equivalent derivation in P.*

Hence, our next step should be to show that one can "reorder" every derivation in P' into a pure derivation. But this is not always possible. E.g., consider the substitution system P such that R contains the rules

$$r = \{ab, cd\} \quad \text{and} \quad r' = \{xc, yc\}.$$

Then

$$\begin{aligned}
\text{trans}(r) &:= \left\{ \begin{array}{l} \{ab, \sigma^{r,1}b\},\ \{\sigma^{r,1}b, \sigma^{r,1}\sigma^{r,2}\},\ \{\sigma^{r,1}\sigma^{r,2}, \tau^{r,1}\sigma^{r,2}\}, \\ \{\tau^{r,1}\sigma^{r,2}, \tau^{r,1}\tau^{r,2}\},\ \{\tau^{r,1}\tau^{r,2}, c\tau^{r,2}\},\ \{c\tau^{r,2}, cd\} \end{array} \right\} \\
\text{trans}(r') &:= \left\{ \begin{array}{l} \{xc, \sigma^{r',1}c\},\ \{\sigma^{r',1}c, \sigma^{r',1}\sigma^{r',2}\},\ \{\sigma^{r',1}\sigma^{r',2}, \tau^{r',1}\sigma^{r',2}\}, \\ \{\tau^{r',1}\sigma^{r',2}, \tau^{r',1}\tau^{r',2}\},\ \{\tau^{r',1}\tau^{r',2}, y\tau^{r',2}\},\ \{y\tau^{r',2}, yc\} \end{array} \right\}.
\end{aligned}$$

Hence, one possible P'-derivation is

$$\begin{array}{lll}
xab & \Rightarrow_{P'} x\sigma^{r,1}b & \Rightarrow_{P'} x\sigma^{r,1}\sigma^{r,2} \\
\Rightarrow_{P'} x\tau^{r,1}\sigma^{r,2} & \Rightarrow_{P'} x\tau^{r,1}\tau^{r,2} & \Rightarrow_{P'} xc\tau^{r,2} \\
\Rightarrow_{P'} \sigma^{r',1}c\tau^{r,2} & \Rightarrow_{P'} \sigma^{r',1}\sigma^{r',2}\tau^{r,2} & \Rightarrow_{P'} \tau^{r',1}\sigma^{r',2}\tau^{r,2} \\
\Rightarrow_{P'} \tau^{r',1}\tau^{r',2}\tau^{r,2} & \Rightarrow_{P'} y\tau^{r',2}\tau^{r,2} & \Rightarrow_{P'} yc\tau^{r,2} \\
\Rightarrow_{P'} y\tau^{r,1}\tau^{r,2} & \Rightarrow_{P'} y\tau^{r,1}\sigma^{r,2} & \Rightarrow_{P'} y\sigma^{r,1}\sigma^{r,2} \\
\Rightarrow_{P'} y\sigma^{r,1}b & \Rightarrow_{P'} yab &
\end{array} \tag{7}$$

Now this can not be reordered into a pure derivation.[5] The reason is just that we start in the third row with the simulation of rule r' without completing the

[5] In private correspondence, P. Pudlák pointed out a similar example leading to this problem.

simulation of rule r. Later in the fifth row, we undo the simulation of the rule r. Hence, there is no direct translation of this derivation into a P-derivation with two rule applications. Nevertheless, there is an equivalent P-derivation, namely

$$xab \Rightarrow_r xcd \Rightarrow_{r'} ycd \Rightarrow_r yab.$$

Hence, we conjecture that it is possible to "complete" every P'-derivation such that the completed derivation can be reordered into a pure derivation. However, we will not prove this, since it is easier to simulate P' in a restricted evolutionary system and then ensure using lethal genes that such derivations do not occur.

3.2 Simulating Substitution Systems by Restricted Evolutionary Systems

Definition 3.4 (Restricted Evolutionary System). *Let $\mathcal{P}_2(\Sigma)$ denote the set of all 2-element subsets of Σ. A restricted evolutionary system $E_{restr} = (\Sigma, \delta, R_S, K)$ is an evolutionary system, where $R_S \subseteq \mathcal{P}_2(\Sigma)$ and $\delta \in \Sigma$ is the only symbol that may be inserted or deleted. The associated Thue-system over the alphabet $(\Sigma \cup \{\$, ¢\})^*$ is defined as*

$$G_{E_{restr}} := R_S \cup \{\{¢, \delta¢\}\}$$

We say that a derivation $x_0 \Rightarrow_{G_{E_{restr}}} \ldots \Rightarrow_{G_{E_{restr}}} x_n$ is valid in E_{restr} (or a valid E_{restr}-derivation) if $\forall i \in 0..n, \forall w \in K : w \not\subseteq x_i$.

We simulate P' by a restricted evolutionary system $E_{restr} = (\Sigma', \delta, R_S, K)$, where we let

$$R_S := \bigcup_{r=\{u,v\} \in R} (\{\{u_i, \sigma^{r,i}\} \mid 1 \leq i \leq |u|\} \cup \{\{v_i, \tau^{r,i}\} \mid 1 \leq i \leq |v| = |u|\}$$

The set of lethal genes K will be defined later. Note that P' and E_{restr} use the same alphabet Σ', and the same symbol δ that may be deleted. In the following, we write $\mu^{r,i}$ to denote the symbol $\sigma^{r,i}$ or $\tau^{r,i}$. We say that a substitution $\{\sigma, \mu^{r,i}\}$ simulates a rule $\{w, w'\}$ if $w \Rightarrow_{\{\sigma, \mu^{r,i}\}} w'$.

Obviously, we can simulate every derivation in P' by the corresponding sequence of substitution, deletions or insertions. For the back translation, we have to ensure using a set K_{con} of lethal genes that a substitution is only applied if it is in the right context (i.e., if it corresponds to the application of one of the rules in some trans(r)).

A word is called admissible for a symbol $\mu^{r,i}$ if it is an element of

$$\mathrm{adm}(\mu^{r,i}) := \{w \in \Sigma'^* \mu^{r,i} \Sigma'^* \mid \exists w' : \{w, w'\} \in \mathrm{trans}(r)\}.$$

We define $\mathrm{adm}(r) := \bigcup_{1 \leq i \leq \mathrm{len}(r)} \mathrm{adm}(\mu^{r,i})$. Note that if w is admissible for $\mu^{r,i}$, then it has length $\mathrm{len}(r)$.

Now the first idea is to include in K_{con} for every rule r and for every $1 \leq i \leq \mathrm{len}(r)$ all words in $\Sigma'^{\mathrm{len}(r)} \backslash \mathrm{adm}(\sigma^{r,i})$. But this is not enough since we must forbid a symbol $\sigma^{r,i}$ to occur too close to the beginning or end of a word produced by the evolutionary

system. Thus, we define

$$K_{con}^{r,i} := \{w\mu^{r,i}w' \mid w, w' \in \Sigma'^* \wedge |w| = i-1 \wedge |w'| = \mathrm{len}(r) - i \wedge w\mu^{r,i}w' \notin \mathrm{adm}(\mu^{r,i})\}$$
$$\cup\ \{\$w\mu^{r,i} \mid w \in \Sigma'^* \wedge |w| < i-1\}$$
$$\cup\ \{\mu^{r,i}w\not\$ \mid w \in \Sigma'^* \wedge |w| < \mathrm{len}(r) - i\}$$
$$K_{con} := \bigcup_{r\in R} \bigcup_{1\le i\le \mathrm{len}(r)} K_{con}^{r,i}.$$

Proposition 3.5. *Let $x = w\mu^{r,i}w'$ such that $w, w' \in \Sigma'^*$, $|w| = i-1$ and $|w'| = \mathrm{len}(r) - i$. Then either $x \in adm(r)$, or $x \in K_{con}$.*

Proof. By definition of K_{con}. □

Lemma 3.6. *Let $x \in \$\Sigma'^*\not\$$, and let $x \Rightarrow_s y$ with s being the substitution $\{\sigma, \mu^{r,i}\} \in R_S$. If neither x nor y is excluded by the lethal genes K_{con}, then there is a unique $r' \in \mathrm{trans}(r)$ in R' simulated by $\{\sigma, \mu^{r,i}\}$ such that $x \Rightarrow_{r'} y$.*

Proof. Let x, y be given as described, let $s = \{\sigma, \mu^{r,i}\}$, where $r = \{u, v\}$ and $1 \le i \le \mathrm{len}(r)$. By the definition of $\mathrm{trans}(r)$, we know that there is exactly one rule $r' \in \mathrm{trans}(r)$ that is simulated by s. Furthermore, we know (again by the definition of $\mathrm{trans}(r)$) that if $x \Rightarrow_{r'} y$, then $r' = \{z, z'\}$ with

$$z = x_{k-i+1}^{k+n-i} \quad \text{and} \quad z' = y_{k-i+1}^{k+n-i},$$

where k is the position in x where the substitution has taken place and $n = \mathrm{len}(r)$. Without loss of generality, we can assume that x_k is $\mu^{r,i}$.

The first case is that s is $\{u_i, \sigma^{r,i}\}$. Then z must be of the forms $z = \sigma^{r,1} \dots \sigma^{r,j} u_{j+1} \dots u_n$ with $j \ge i$ or $z = \tau^{r,1} \dots \tau^{r,j-1}\sigma^{r,j} \dots \sigma^{r,n}$ with $j \le i$, since otherwise $z \in K_{con}$ by Proposition 3.5. But then z' must be $\sigma^{r,1} \dots \sigma^{r,i-1} u_i \dots u_n$, since otherwise $z' \in K_{con}$. This implies $\{z, z'\} \in \mathrm{trans}(r)$.

The other cases for s are treated analogous. □

The final observation is that we can re-order every derivation D in the evolutionary system $E'_{restr} = (\Sigma', \delta, R_S, K_{con})$ such that it can be translated back into a pure P'-derivation. For this purpose, we need another set of lethal genes. These lethal genes will guarantee that different applications of substitution interact within the corresponding context if and only if they are P-consecutive rule applications. Let

$$K_{inter} := \{\mu^{r,i}w\mu^{r',j} \mid w \in \Sigma'^* \wedge 1 \le |w| \le \max(\mathrm{len}(r), \mathrm{len}(r)) - i + j - 1\}$$

In the remaining part of the simulation, we set $E_{restr} = (\Sigma', \delta, R_S, K_{con} \cup K_{inter})$.

Lemma 3.7. *Every pure P'-derivation has an equivalent derivation in E_{restr}. Conversely, every derivation in E_{restr} has an equivalent derivation in P'.*

Proof (sketch). The second claim is a corollary of Lemma 3.6. For the first claim, we only have to show that in pure P'-derivations, no lethal gene in $K_{con} \cup K_{inter}$ can be applied. Now assume that some lethal gene in K_{con} applies in some pure

P'-derivation D. Then we know that there is a $\mu^{r,i}$ in some word x used in D such that

$$x_{k-i+1}^{k+\mathrm{len}(r)-i} \in K_{con} \text{ or } x_{k-i+1}^{i} \in K_{con} \text{ or } x_{i}^{k+\mathrm{len}(r)-i} \in K_{con}$$

By induction on length of pure derivations we can show that $\mu^{r,i}$ must be contained in the context of the last rule application. Hence, it occurs in the i^{th} position of this context, which implies that $p > i-1$ and $|x| - p > \mathrm{len}(r) - i$. Furthermore, this implies that $x_{k-i+1}^{k+\mathrm{len}(r)-i} \in \mathrm{adm}(r)$ for some $r \in R$, which contradicts our assumption that $x_{k-i+1}^{k+\mathrm{len}(r)-i} \in K_{con}$.

To show that no lethal genes of K_{inter} applies, it is sufficient to show that whenever two symbols $\mu^{r,i}$ and $\mu^{r',j}$ occur in a word in the derivation D, they are consecutive (since K_{inter} contains no word having this property). This can be done by induction on length of pure derivations. □

Note that not all P' derivation can be translated into E_{restr}. Especially, derivations of the form depicted in (7) cannot be translated. Given a pure derivation in P', we denote with $\mathrm{sim}_{evol}(D)$ the equivalent derivation that is valid in E_{restr}. Conversely, given a derivation D valid in E_{restr}, we denote with $\mathrm{sim}_{subst}(D)$ the unique derivation in P' which is equivalent to D.

Lemma 3.8. *Let D be a valid E_{restr}-derivation with $\mathrm{sim}_{subst}(D) = x \Rightarrow_{r_1} y \Rightarrow_{r_2} z$ such that the applications of r_1 and r_2 in $\mathrm{sim}_{subst}(D)$ are not P-consecutive. Then there is some y' and an valid E_{restr}-derivation D' such that $\mathrm{sim}_{subst}(D') = x \Rightarrow_{r_2} y' \Rightarrow_{r_1} z$.*

The key observation is that since the applications of r_1 and r_2 are not P-consecutive, K_{inter} guarantees that either $x = usvtw$, $y = us'vtw$ and $z = us'vt'w$ such that $r_1 = \{s, s'\}$ and $r_2 = \{t, t'\}$, or $x = usvtw$, $y = usvt'w$ and $z = us'vt'w$ such that $r_1 = \{t, t'\}$ and $r_2 = \{s, s'\}$.

Lemma 3.9 (Reordering). *Let $x, y \in \$\Sigma^*\not\$$ and D be a derivation valid in E_{restr}, and let $\mathrm{sim}_{subst}(D)$ be $x = x_0 \Rightarrow_{r_1} x_1 \dots x_{n-1} \Rightarrow_{r_n} x_n = y$. Then there is a permutation π of $\{1, \dots, n\}$ such that $x = x'_0 \Rightarrow_{r_{\pi(1)}} x'_1 \cdots x'_{n-1} \Rightarrow_{r_{\pi(n)}} x'_n = y$ is a pure P'-derivation.*

Lemma 3.10. *For every derivation in P there is an equivalent derivation in E_{restr} and vice versa.*

3.3 Simulating Restricted Evolutionary Systems

Finally, we simulate the restricted evolutionary system E_{restr} by an evolutionary system $E = (\{0,1\}, K^{\{0,1\}})$ over the alphabet $\{0,1\}$. Thus, we show that there is an evolutionary system $E = (\Sigma, K)$ such that $\Rightarrow^*_{\Sigma,K,r}$ is undecidable. The undecidability of $\Rightarrow^*_{\Sigma,K}$ will be shown to be an immediate consequence of the undecidability of $\Rightarrow^*_{\Sigma,K,r}$. This completes the proof of theorem 2.3, since the claim of this theorem is an immediate consequence of the undecidability $\Rightarrow^*_{\Sigma,K}$.

We start with an injective mapping $\phi : \Sigma' \cup \{\$, \not\$\} \to \{0,1\}^* \cup \{\$, \not\$\}$. Let $\{\sigma_1, \dots, \sigma_m\}$ be an enumeration of Σ'. Then we define $\phi(\$) := \$$, $\phi(\not\$) := \not\$$ and

$$\phi(\sigma_i) := 11100^{i-1}10^{m-i}0111$$

As usual, we use the symbol ϕ also for the homomorphic extension of ϕ to $(\Sigma' \cup \{\$, ¢\})^*$. Furthermore, we define a mapping γ of rules into words in $\{0,1\}^*$ by

$$\gamma(\{\sigma_i, \sigma_j\}) := \phi(\sigma_i) \vee \phi(\sigma_j),$$

where $\vee$ is the bitwise logical or.

The idea is to translate every word in $\$\Sigma'^*¢$ into $\$\{0,1\}^*¢$ using ϕ, and to simulate an application of a rule $\{\sigma_i, \sigma_j\}$ with $i < j$ by a sequence of substitutions as described in example (6) on page 7. Note that the middle word in (6) is just $\gamma(\{\sigma_i, \sigma_j\})$. Then for simulating E_{restr}, it is clear that $K^{\{0,1\}}$ must at least contain the translation of the lethal genes of E_{restr}, namely the set

$$\begin{aligned} K^{\{0,1\}}_{trans} := &\{\phi(w) \mid w \in K_{con} \cup K_{inter}\} \\ &\cup \{\gamma(r) \mid r \in \mathcal{P}_2(\Sigma) \wedge r \notin R_S\}. \end{aligned}$$

Additionally, we need a set of lethal genes $K^{\{0,1\}}_{code}$ excluding all words that are not elements of $L_{code} := \$L^*_{trans}L_{pre}(\delta)¢$, where

$$\begin{aligned} L_{trans} &:= \{w \in \{0,1\}^* \mid w \in \mathrm{img}(\phi) \cup \mathrm{img}(\gamma)\} \\ L_{pre}(\delta) &:= \{w \mid w \text{ prefix of } \phi(\delta) \text{ and } w \neq \phi(\delta)\ \} \end{aligned}$$

Note that all words in L_{trans} have the same length, namely $m + 8$. We need the $L_{pre}(\delta)$ part in L_{code} since the deletion of a δ in E_{restr} must be simulated by $m+8$ deletions in E. In the following, we say that a word w is a pre-codon if $w = 0111$ or $w = \$$. These are the words directly before the encoding of a $\sigma \in \Sigma'$ in a word in L_{code}.

$$\begin{aligned} K^{\{0,1\}}_{code} := &\{w_p w \mid w_p \text{ pre-codon } \wedge\ w \in \{0,1\}^* \wedge |w| = m + 8\ \wedge\ w \notin L_{trans}\} \\ &\cup \{w_p w¢ \mid w_p \text{ pre-codon } \wedge\ |w| < m + 8\ \wedge\ w \notin L_{pre}(\delta)\} \end{aligned}$$

Note that words having more than six consecutive 1's contain an element of K_{code} as a convex subword. As in the simulation of substitution system by restricted evol. systems, we need to define an order on rule application for the back translation, for which we need an additional set of lethal genes $K^{\{0,1\}}_{inter}$. To motivate $K^{\{0,1\}}_{inter}$, consider the system $E_{restr} = (\{a, b\}, \delta, \{\{a, b\}\}, \{ab, ba\})$. Then we have $\phi(a) = 1110100111$, $\phi(b) = 1110010111$ and $\gamma(\{a, b\}) = 1110110111$. A valid derivation in $E' = (\{0,1\}, K^{\{0,1\}}_{trans} \cup K^{\{0,1\}}_{code})$ is

$$\begin{aligned} \$|1110\underline{10}0111|1110\underline{10}0111|¢ &\Rightarrow_r \$|1110\underline{11}0111|1110\underline{10}0111|¢ \\ &\Rightarrow_r \$|1110\underline{11}0111|1110\underline{11}0111|¢ \\ &\Rightarrow_r \$|1110\underline{01}0111|1110\underline{11}0111|¢ \\ &\Rightarrow_r \$|1110\underline{01}0111|1110\underline{01}0111|¢, \end{aligned} \tag{8}$$

which cannot be translated back into a valid E_{restr}-derivation. The problem with this derivation is that we started the simulation of two rule-applications in (8) in parallel, thus avoiding that the word ba becomes "visible". This problem can be fixed by defining

$$\begin{aligned} K^{\{0,1\}}_{inter} := &\{\gamma(r)w\gamma(r') \mid w \in L^*_{trans} \wedge r, r' \in R_S \wedge w \leq l_{\max}\} \\ &\cup \{\gamma(r)w¢ \mid w \in L^*_{trans}L_{pre}(\delta) \wedge r \in R_S \wedge w \leq l_{\max},\} \end{aligned}$$

where $l_{\max}$ is $\max\{|w| \mid w \in K^{\{0,1\}}_{trans}\}$. We define the evolutionary system E simulating E_{restr} by $E = (\{0,1\}, K^{\{0,1\}})$, where $K^{\{0,1\}} = K^{\{0,1\}}_{trans} \cup K^{\{0,1\}}_{code} \cup K^{\{0,1\}}_{inter}$. Then we can again reorder the derivations in E appropriately, thus getting the following lemma.

Lemma 3.11. *For every E_{restr}-derivation D there is an equivalent E-derivation D' and vice versa.*

Theorem 3.12. *There is an evolutionary system $E = (\Sigma, K)$ such that the relations $\Rightarrow^*_{\Sigma,K,r}$ and $\Rightarrow^*_{\Sigma,K}$ are undecidable.*

Proof. The first claim follows from the undecidability of Thue-systems together with Lemmas 3.2, 3.10 and 3.11. The second claim follows from the fact that given a word $w \in L^*_{trans}$, then deleting symbol in w generates a word $w' \notin L_{trans}$. Since all words $w' \notin L_{trans}$ are excluded by $K^{\{0,1\}}_{code}$, this implies that $\Rightarrow^*_{\Sigma,K,r} = \Rightarrow^*_{\Sigma,K}$. □

Theorem 2.3 is an immediate consequence of the last theorem.

4 Concluding remarks

In this paper, we have defined and analyzed a simple model for asexual evolution. Despite the extreme simplicity of the model, we have shown that one cannot answer questions such as whether given genomes x, y are in the same ergocity class modulo finite set K of lethal genes – i.e. whether there is a sequence of descendents from x to y.

We also wrote a simulation program in C to generate the set $N_k(x, K)$ of descendents (or neighbors) y within k generations of x, modulo a given set K of lethal genes. Our intent was to investigate the structure of the phylogeny tree below x, and to determine the relation between the Levenshtein distance from x to K (also the density or clustering of K) and the size of $N_k(x, K)$ as a function of k. This latter problem seems to be related to the earlier cited work of Chvátal-Sankoff [4], and appears to be quite difficult, as illustrated by the following example furnished by J. Sgall (personal communication). Let x be the empty word, $K_1 = \{01\}$ and $K_2 = \{00\}$. Then

$$
\begin{aligned}
N_k(x, K_1) &= \{0^i, 1^i, 1^r0^s \mid i \le k, r + s \le k\} \quad \text{and} \\
N_k(x, K_2) &= \{1^{n_1}01^{n_2}0\cdots01^{n_i} \mid n_j > 0 \text{ and } i - 1 + n_1 + \ldots n_i \le k\} \\
&\quad \cup \{01^{n_1}01^{n_2}0\cdots01^{n_i}, 1^{n_1}01^{n_2}0\cdots01^{n_i}0 \mid n_j > 0 \text{ and } i + n_1 + \ldots n_i \le k\} \\
&\quad \cup \{01^{n_1}01^{n_2}0\cdots01^{n_i}0 \mid n_j > 0 \text{ and } i + 1 + n_1 + \ldots n_i \le k\}
\end{aligned}
$$

It follows that the number of elements of $N_k(x, K_1)$ is linear in k, whereas that of $N_k(x, K_2)$ is exponential in k. However, under any sort of dissimilarity criterion, x is the same distance from K_1 as from K_2.

Acknowledgments Special thanks go to Pavel Pudlák for several key discussions. We are indebted to him in particular for having explained his technique [13] for reducing to a rewrite system whose rules change at most one symbol, and for finding an error in a working draft of this paper. Thanks to K. Ramsey for several conversations and for suggesting a variant of the problem investigated in this article, to G. Mints for the references from S. Yu. Maslov, and D. Kerski who had suggested that undecidability results should be implicit in biological systems. Finally, thanks to Christa Mauer for some help with typing the manuscript.

References

1. L. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 266:1021–1024, 1994.
2. S. Arora, Y. Rabani, and U. Vazirani. Simulating quadratic dynamical systems is $PSPACE$-complete (preliminary version). In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing*, pages 459–467, 1994.
3. C.H. Bennett. Time/space trade-offs for reversible computation. *SIAM Journal on Computing*, 18:766–776, 1989.
4. V. Chvátal and D. Sankoff. Longest common subsequences of two random sequences. *Journal of Applied Probability*, 12:306–315, 1975.
5. J. Dassow and V. Mitrana. Evolutionary grammars: a grammatical model for genome evolution. In *Computer Science and Biology*, pages pp. 87–92, 1996. Proceedings of the German Conference on Bioinformatics (GCB'96).
6. L. Ilie and V. Mitrana. Crossing-over on languages: A formal representation of the recombination of genes in a chromosome. Technical Report 16, Turku Centre for Computer Science, June 1996.
7. Lila Kari and Gabriel Thierrin. Contextual insertion/deletions and computability. *Information and Computation*, 131(1):47–61, 1996.
8. L.Kari, G.Paun, G.Thierrin, and S.Yu. At the crossroads of DNA computing and formal languages: Characterizing recursively enumerable languages by insertion-deletion systems. In *Proceedings of the* 3^{rd} *DIMACS Workshop on DNA-based Computers*, pages 318–333, Philadelphia, June 1997.
9. S. Yu. Maslov. The mutation calculi. *Journal of Soviet Mathematics*, 10:495–517, 1978.
10. S. Yu. Maslov. Absorption relation on regular sets. *Journal of Soviet Mathematics*, 14:1468–1475, 1980.
11. S. Yu. Maslov. *Theory of Deductive Systems and its Applications*. MIT Press (Cambridge), 1987.
12. G. Paun and A. Salomaa. From DNA recombination to DNA computing, via formal languages. In *Computer Science and Biology*, pages pp. 93–98, 1996. Proceedings of the German Conference on Bioinformatics (GCB'96).
13. P. Pudlák. Complexity theory and genetics. In *Proceedings of 9th Annual IEEE Conference on Structure in Complexity Theory*, 1994.
14. M. S. Waterman. *Introduction to Computational Biology - Maps, sequences and genomes*. Chapman and Hall, 1995.

A Simulation of Turing Machine by Asexual Evolution

In future work, we intend to characterize complexity classes via bounded evolutionary systems. Since complexity classes are usually defined by Turing machines, in order to capture complexity classes by evolutionary systems (a reversible notion), one thus needs to consider "reversible" Turing machines. C. Bennett [3] has shown how to simulate 1-tape Turing machines by 3-tape reversible Turing machines.[6] Our construction of an evolutionary system then would require something like a 1-tape reversible TM. Since it is not clear that reversibility can be preserved in tape reduction, we preferred simply to define a "special" TM, and show how to simulate a 1-tape special TM.

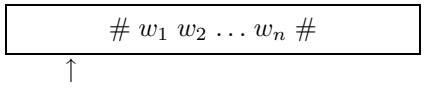
Definition A.1. *A 1-tape Turing machine (abbreviated TM) is given by* $M = (Q, \Gamma, \delta, s)$*, where* Q *is a finite set of states containing the halt state* h*,* Γ *is a finite alphabet not*

[6] It is slightly tricky to define the notion of reversible TM. The intuition is to simple reverse the 4-tuples of the form (q, a, q', b), (q, a, q', R) and (q, a, q', L); however it is slightly problematic to reverse transitions where the head moves.

containing the blank symbol #, $s \in Q$ is the start state, and the transition function δ maps

$$(Q - \{h\}) \times (\Gamma \times \{\#\}) \to Q \times (\Gamma \cup \{\#, L, R\}).$$

The TM M begins its computation on a 1-way infinite tape with input $w = w_1 \cdots w_n \in \Gamma^*$ in state s, with the tape head on the leftmost symbol.

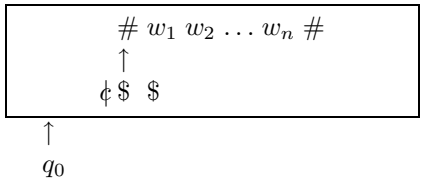


A configuration α in the computation of TM M on input x is a word $[uqv]$, where $q \in Q$, $u, v \in (\Gamma \cup \{\#\})^*$, $|u|, |v| \geq 1$, the brackets ']', '[' do not belong to $\Gamma \cup \{\#\}$), and where ## is neither a prefix of u nor suffix of v. The graph Δ of δ is the set

$$\{(q, a, p, b) : \delta(q, a) = (p, b)\}$$

As usual, if α, β are configurations of M, then $\alpha \vdash_M \beta$ means that β is the configuration immediately following α, using the transition function of M. As well, $\vdash_M^*$ is the reflexive, transitive closure of $\vdash_M$.

Let $M = (Q, \Sigma, \delta, s)$ be a 1-tape TM. Define the 1-tape TM $\widetilde{M}$ which has three tracks on its single tape, and begins its computation as follows:



The TM $\widetilde{M}$ simulates the movements of M and additionally keeps the history of the rules applied. Starting at the far left end of its tape, $\widetilde{M}$ sweeps to the right in a state similar to M's current state until the rightmost '$' is encountered on the third track. In proceeding right, $\widetilde{M}$ determines what character M would have read, and performs the actions M would have on its second track. At the right end, $\widetilde{M}$ erases the '$', writes the $r \in \Delta$ which M applied, and prints a '$' to the right. Then $\widetilde{M}$ sweeps back to the leftmost '$', and begins again. In summary, to simulate one of M's steps, $\widetilde{M}$ requires a sweep from the far left to far right, and back again. It follows that if M uses s tape cells and takes t steps in its computation on w, then $\widetilde{M}$ uses $3 + \max(s, t)$ tape cells and takes time $O(t^2)$.

In reading the transition rules of $\widetilde{\delta}$ below, it may help to know that state names have been chosen to indicate either current action or future action to be performed. For instance, $q_{j,R}^{r,R,\uparrow}$ is the state of $\widetilde{M}$ which simulates the state q_j of M, where the next action to be taken when in this state is to move right (indicated by superscript 'R'), then afterwards to write $\uparrow$, while $\widetilde{M}$ is simulating M's state q_j in its initial sweep to the right.

Formally, $\widetilde{M} = (\widetilde{Q}, \widetilde{\Sigma}, \widetilde{\delta}, s)$ where $\widetilde{Q}$ is

$$\left\{ \begin{array}{l} q_0, q_{i,R}, q_{i,R}^r, p_{i,R}^{r,L,\uparrow}, q_{i,R}^{r,R,\uparrow}, p_{i,R}^{r,\uparrow}, q_{i,R}^{r,\uparrow}, \\ p_{i,R}^r, q_{i,R}^{r,\$}, p_{i,R}^{r,\$}, p_{i,R}^{\$,L}, q_{i,R}^{\$,L}, p_{i,L}^L, q_{i,L}^L : 0 \leq i \leq n, r \in \Delta \end{array} \right\},$$

$\widetilde{\Sigma}$ is

$$(\Sigma \cup \{\#\}) \times \{\uparrow, \square\} \times \{\$, \square, ¢, r : r \in \Delta\},$$

and where $\widetilde{\delta}$ is described as follows.

1. q_0 $\boxed{\begin{matrix} \\ \\ \not\vdash \end{matrix}}$ $q_{0,R}$ R

2. $q_{i,R}$ $\boxed{\begin{matrix} \sigma \\ \neq\uparrow \\ \neq \\ \not\vdash \end{matrix}}$ $q_{i,R}$ R

3. $q_{i,R}$ $\boxed{\begin{matrix} \sigma \\ \uparrow \\ \gamma \end{matrix}}$ $q^r_{j,R}$ $\boxed{\begin{matrix} \sigma' \\ \uparrow \\ \gamma \end{matrix}}$, if $r = q_i \sigma q_j \sigma' \in \Delta$

4. $q_{i,R}$ $\boxed{\begin{matrix} \sigma \\ \uparrow \\ \gamma \end{matrix}}$ $p^{r,L,\uparrow}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \gamma \end{matrix}}$, if $r = q_i \sigma q_j L \in \Delta$

5. $q_{i,R}$ $\boxed{\begin{matrix} \sigma \\ \uparrow \\ \gamma \end{matrix}}$ $q^{r,R,\uparrow}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \gamma \end{matrix}}$, if $r = q_i \sigma q_j R \in \Delta$

6. $p^{r,L,\uparrow}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \gamma \end{matrix}}$ $p^{r,\uparrow}_{j,R}$ L

7. $q^{r,R,\uparrow}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \gamma \end{matrix}}$ $q^{r,\uparrow}_{j,R}$ R

8. $p^{r,\uparrow}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \gamma \end{matrix}}$ $p^{r,R}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \uparrow \\ \gamma \end{matrix}}$

9. $q^{r,\uparrow}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \gamma \end{matrix}}$ $q^r_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \uparrow \\ \gamma \end{matrix}}$

10. $q^r_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \gamma \\ \neq \$ \end{matrix}}$ $q^r_{j,R}$ R

11. $q^r_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \gamma \\ \$ \end{matrix}}$ $q^{r,\$}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \gamma \\ r \end{matrix}}$

12. $q^{r,\$}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \gamma \\ r \end{matrix}}$ $q^{\$,L}_{j,R}$ R

13. $q^{\$,L}_{j,R}$ $\boxed{\begin{matrix} \sigma \\ \\ \end{matrix}}$ $q^L_{j,L}$ $\boxed{\begin{matrix} \sigma \\ \\ \$ \end{matrix}}$

14. $q^L_{j,L}$ $\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$ $q_{j,L}$ L

15. $q_{j,L}$ $\boxed{\begin{array}{c}\sigma\\ \gamma\\ \neq \$\end{array}}$ $q_{j,L}$ L

16. $q_{j,L}$ $\boxed{\begin{array}{c}\sigma\\ \gamma\\ \$\end{array}}$ $q_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \gamma\\ \$\end{array}}$

17. $p^{r,R}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \uparrow\\ \gamma\end{array}}$ $p^r_{j,R}$ R

18. $p^r_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \neq \$\end{array}}$ $p^r_{j,R}$ R

19. $p^r_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$ $p^{r,\$}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \gamma\\ r\end{array}}$

20. $p^{r,\$}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \gamma\\ r\end{array}}$ $p^{\$,L}_{j,R}$ R

21. $p^{\$,L}_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \end{array}}$ $p^L_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$

22. $p^L_{j,R}$ $\boxed{\begin{array}{c}\sigma\\ \\ \$\end{array}}$ $q_{j,L}$ L

We assume $\widetilde{M}$ starts on the following configuration.

$$\boxed{\begin{array}{c}\#\ w_1\ w_2\ \ldots\ w_n\ \#\\ \uparrow\\ \not\$\ \ \$\end{array}}$$
$$\uparrow$$
$$q_0$$

Definition A.2. *If $M = (Q, \Gamma, \delta, q_0)$ is a TM, and $x \in \Gamma$, then $\alpha_{0,x}$ denotes the initial configuration of M on input x. The TM M is injective or 1-1 on $x \in \Sigma^*$ if for all $\alpha, \beta, \gamma \in \{\theta : \alpha_{0,x} \vdash^*_M \theta\}$ $\alpha \vdash_M \gamma$ and $\beta \vdash_M \gamma$ implies $\alpha = \beta$.*

The 1-tape TM M is special if it is 1-1 on all inputs x in Γ^* and for all configurations α, β, $\beta = (\beta_1, \ldots, \beta_m)$ in $\{\theta : \alpha_{0,x} \vdash^*_M \theta\}$, if $\alpha \vdash_M \beta$, i is the position of M's head on β, and q is the state of M on β, then knowledge of $q, \beta_{i-1}, \beta_i, \beta_{i+1}$ uniquely determines the r of M used in the transition $\alpha \vdash_M \beta$.

Our notion of *special* 1-tape TM is a technical variant of Bennett's notion of *reversible* 3-tape TM.

Claim. $\widetilde{M}$ is special.

PROOF: The proof is by meticulous inspection of the transition rules of $\widetilde{\delta}$, using the encoded rule (e.g. rule r in $q^r_{j,R}$), or an adjacent symbol in β_{i-1}, β_i, β_{i+1} to disambiguate which rule was applied.

We summarize the considerations that must be made. Assume that a rule has just been performed, and that $\widetilde{M}$ is now in a new state q, and that β_{i-1}, β_i, β_{i+1} are the 3 symbols on $\widetilde{M}$'s tape in the immediate vicinity of $\widetilde{M}$'s head.

CASE 1: q is $q_{j,R}$.
Rule 1,2 or 16 must have been used. Rule 1 can be distinguished from the other two by whether β_{i-1} has $\not\in$ on the third track. Rule 2 cannot be applied when the head is on the cell containing the rightmost '\$' on its third track; the head can be in this position only when applying rule 16.

CASE 2: q is $q^r_{j,R}$.
Rule 3,9 or 10 must have been used. Rules 3,9 can be distinguished by r. Note that after application of rule 10, '↑' cannot be in the middle track, since $q^r_{j,R}$ is a state entered only after M's rule has been successfully simulated. This distinguishes rule 10 from rules 3,9.

CASE 3: q is $p^r_{j,R}$.
Rule 17 or 18 must have been used. These rules can be distinguished by the presence of '↑' on the second track in the vicinity.

CASE 4: q is $q_{j,L}$.
Rule 14,15 or 22 must have been used. These rules can be distinguished by looking at a history element (a rule) on the third track in the vicinity.

CASE 5: q is not of the above.
Then q is of the form $p^{r,L,\uparrow}_{j,R}$, $q^{r,R,\uparrow}_{j,R}$, $p^{r,\uparrow}_{j,R}$, $q^{r,\uparrow}_{j,R}$, $p^{r,R}_{j,R}$, $q^{r,\$}_{j,r}$, $q^{\$,L}_{j,R}$, $q^{L}_{j,L}$, $p^{r,\$}_{j,R}$, $p^{\$,L}_{j,R}$, or $p^{L}_{j,R}$. Each of these was obtained by a unique rule. □

Since $\widetilde{M}$ is special, one can simulate $\widetilde{M}$ in a straightforward manner by a Thue system, and hence by an evolutionary system. Since the special Turing machine $\widetilde{M}$ simulating 1-tape Turing machine M keeps the history of M's past, the evolutionary system E encodes M's past. In this sense, we have indicated a formal mechanism how the history of development of life could be encoded by individuals of a species.