Mutation Systems

Dana Angluin James Aspnes Raonne Barbosa Vargas

Department of Computer Science
Yale University

LATA 2011

Introduction

- Biological evolution proceeds by variation and selection
- A model of evolution of strings

  Variation:
  A function μ mapping a string to possible mutations
  Selection:
  A function f deciding whether a string is fit

- Evolvability:
  Can s evolve to t via stepwise mutations to fit strings?

Mutation Systems

A **mutation system** (Σ, μ, f) has

- an alphabet Σ
- a mutator $\mu : \Sigma^* \rightarrow 2^{\Sigma^*}$
  μ maps a string to the set of its mutations
- a fitness function $f : \Sigma^* \rightarrow \{0, 1\}$
  f decides whether a string is fit (1) or not (0)

Mutation and Evolution: Reachability

Let $S = (\Sigma, \mu, f)$ be a mutation system

- $s \to_\mu t$ if $t \in \mu(s)$
  s can mutate to t in one step
- $s \to_S t$ if $s \to_\mu t$ and $f(s) = f(t) = 1$
  s can evolve to t in one step
  (both s and t must be fit)

s can mutate to t if $s \to_\mu^* t$
s can evolve to t if $s \to_S^* t$

An Example Mutation System $S = (\Sigma, \mu, f)$

Example S:

- $\Sigma = \{a, b, c\}$
- $\mu(s)$ = strings obtained by swapping adjacent symbols in s
- $f(s) = 1$ if no two adjacent symbols are equal

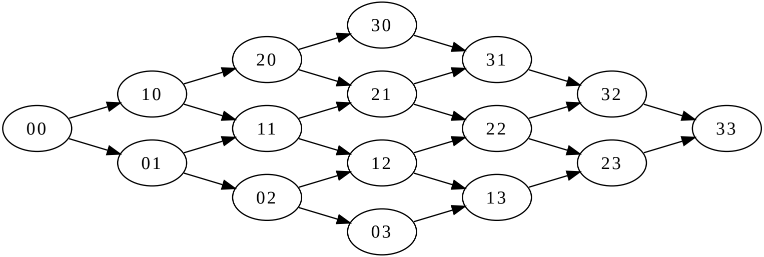
Mutation steps:

$$abcbc \rightarrow_{\mu} abccb \rightarrow_{\mu} acbcb \rightarrow_{\mu} cabcb$$

But *abccb* is not fit!
Alternative evolution steps:

$$abcbc \rightarrow_{S} bacbc \rightarrow_{S} bcabc \rightarrow_{S} bcacb \rightarrow_{S} cbacb \rightarrow_{S} cabcb$$

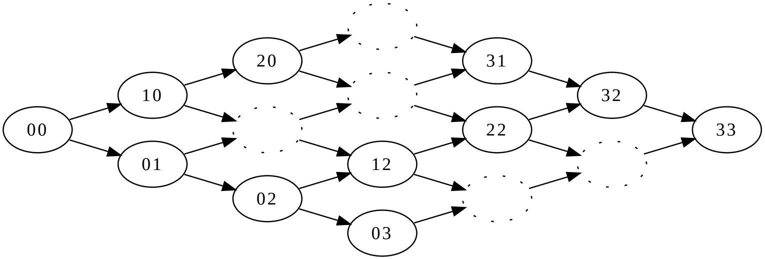
The Mutation Graph: $(\Sigma^*, \rightarrow_\mu)$



Nodes: all strings of length 2 over $\{0, 1, 2, 3\}$

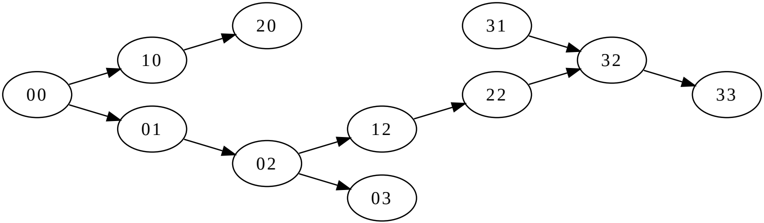
Directed edges: $\rightarrow_\mu$

Unfit Strings are Removed



Remove **unfit** strings to get evolvability graph

The Evolvability Graph: $(f^{-1}(1), \rightarrow_S)$



Nodes: **fit** strings of length 2 over $\{0, 1, 2, 3\}$

Directed edges: $\rightarrow_S$

Deciding Evolvability?

The problem

Input: A mutation system and two strings s and t

Output: Can s evolve to t?

is undecidable for

$\mu =$ point mutations

$f =$ a strictly 2-testable predicate

Point Mutations

A point mutation of s is obtained by

- replacing one symbol in s
- or deleting one symbol from s
- or inserting one symbol in s

So $\mu(bcb)$ contains

- acb, ccb, bab, bbb, bca, bcc
- cb, bb, bc
- $abcb$, $bbcb$, $cbcb$, $bacb$, $bccb$, $bcab$, $bcbb$, $bcba$, $bcbc$

Point mutations are reversible: the mutation graph is undirected

Strictly k-Testable Fitness Functions

A strictly k-testable L given by (PRE, MID, SUF)

- PRE contains strings of length $k-1$
- MID contains strings of length k
- SUF contains strings of length $k-1$

L contains all strings s such that

- length $k-1$ prefix of s in PRE
- every length k substring of s in MID
- length $k-1$ suffix of s in SUF

Fitness function $f_L(s) = 1$ iff $s \in L$

Example: a Strictly 2-Testable Fitness Function

Fitness function f with

$$\begin{aligned} \mathrm{PRE} &= \{a, b\} \\ \mathrm{MID} &= \{aa, ac, bb, bd, cc, dd\} \\ \mathrm{SUF} &= \{c, d\} \end{aligned}$$

has fit strings $a^+c^+ + b^+d^+$

Symbol Duplication

How to control point mutations?

- Duplication map $d(s)$ replaces each symbol x by $x_1 x_2$
- Define a fitness function:

  PRE contains all symbols x_1
  MID contains all pairs of symbols $x_1 x_2$ and $y_2 x_1$
  SUF contains all symbols x_2

- Fit strings are $d(s)$
- Point mutations of fit strings are unfit

Simulating a FSM

Evolvability $\leftrightarrow$ computational reachability

- Issue of reversibility? Use computation histories
- Annotate symbol read with state (x if unread)

Example M

- $\Sigma = \{a, b\}$
- $\delta(s) =$ parity of a's

Histories of M on input $abaa$

- Initial history: $a_x b_x a_x a_x$
- History after first step: $a_1 b_x a_x a_x$
- ...
- Final history: $a_1 b_1 a_0 a_1$

Mutations Simulating a FSM

Duplicate history symbols: a_q^1, a_q^2

PRE: index 1, unread or correct transition from q_0

MID: indices 1,2

- main input symbols equal
- both unread or both read and states equal
- first read and second unread

MID: indices 2,1

- both unread
- first read and second unread
- both read and state transition to second correct

SUF: index 2, unread or read

Example of Mutations Simulating M on $abaa$

Initial history, all unread

$$a_x^1 \quad a_x^2 \quad b_x^1 \quad b_x^2 \quad a_x^1 \quad a_x^2 \quad a_x^1 \quad a_x^2$$

First symbol read

$$a_1^1 \quad a_x^2 \quad b_x^1 \quad b_x^2 \quad a_x^1 \quad a_x^2 \quad a_x^1 \quad a_x^2$$

Duplicate of first symbol updated

$$a_1^1 \quad a_1^2 \quad b_x^1 \quad b_x^2 \quad a_x^1 \quad a_x^2 \quad a_x^1 \quad a_x^2$$

Second symbol read

$$a_1^1 \quad a_1^2 \quad b_1^1 \quad b_x^2 \quad a_x^1 \quad a_x^2 \quad a_x^1 \quad a_x^2$$

Duplicate of second symbol updated

$$a_1^1 \quad a_1^2 \quad b_1^1 \quad b_1^2 \quad a_x^1 \quad a_x^2 \quad a_x^1 \quad a_x^2$$

Reversible Cellular Automata

A 1-dimensional reversible asynchronous cellular automaton:

- An alphabet Σ
- Transition rules
  - Substitutions: $axb \leftrightarrow ayb$
  - Insertions/Deletions: $axb \leftrightarrow ab$

Example:

- Rules $\{abc \leftrightarrow adc, dce \leftrightarrow dfe, fe \leftrightarrow fge\}$
- Reachable from $abce$ are $\{abce, adce, adfe, adfge\}$

Simulating a Cellular Automaton

From a cellular automaton C to a mutation system S:

- Symbols ${}_u a^i_v$
  - main symbol component a from Σ
  - index i from $\{1, 2, *\}$
  - left neighbor information u
  - right neighbor information v
- Rules from substitution and deletion/insertion rules of C

Application of a rule of C becomes a sequence of mutations:

- Symbol "locks" its neighbors
- Symbol then changes
- Symbol "unlocks" its neighbors
- Up to 14 mutations for 1 rule application

Example of Locking

Starting with $d(abcde)$:

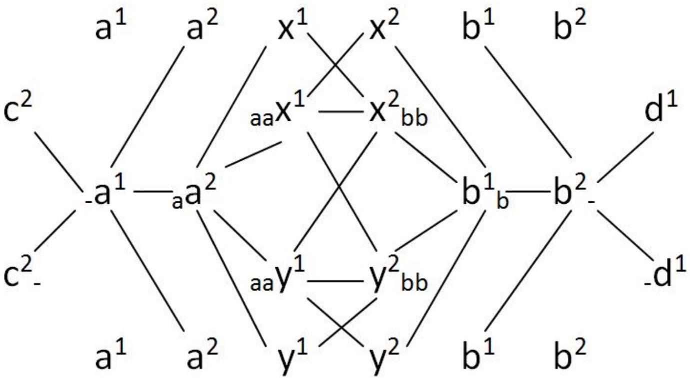
$$a^1 \cdot a^2 \cdot b^1 \cdot b^2 \cdot c^1 \cdot c^2 \cdot d^1 \cdot d^2 \cdot e^1 \cdot e^2$$

After several mutations:

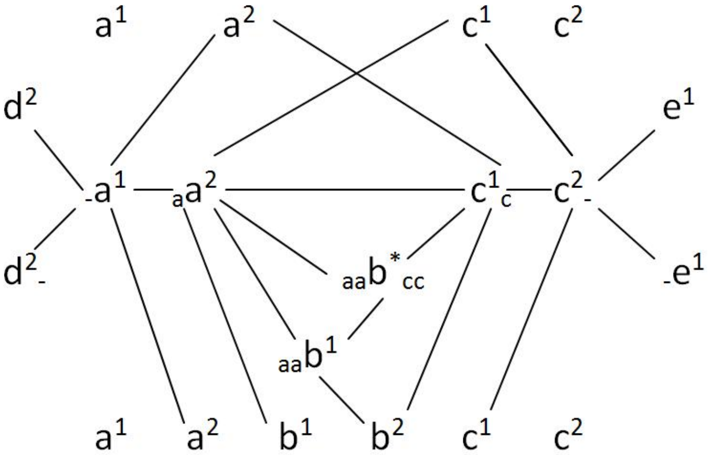
$$a^1 \cdot a^2 \cdot {}_{\_}b^1 \cdot {}_{b}b^2 \cdot {}_{bb}c^1 \cdot c^2_{dd} \cdot d^1_{d} \cdot d^2_{\_} \cdot e^1 \cdot e^2$$

symbol c has locked its left and right neighbors
and is prepared for a rule application

Fitness Pairs for $axb \leftrightarrow ayb$



Fitness Pairs for $abc \leftrightarrow ac$



Deciding Evolvability?

Thus the problem

Input: A mutation system and two strings s and t

Output: Can s evolve to t?

is undecidable for

$\mu =$ point mutations

$f =$ a strictly 2-testable predicate

Other Questions

Random point mutations?

- FSM simulation becomes a random walk: $O(n^3)$ steps
- Can be biased forward: $O(n^2)$ steps
- More generally?

Learnability?

- Mutation process known & fitness function unknown?
- k-testable languages POS limit learnable [GV 1990]
- Also concatenations of k-testable languages [KY 1994]
- Stochastic results?

Thank you!

(Research supported by NSF Grant CCF-0916389)