Genome pattern matching using regular expressions

Simon Nicolai Lefoli Maibom - xvm226

Arinbjörn Brandsson - hkt789

Martin Simon Haugaard - cdl966
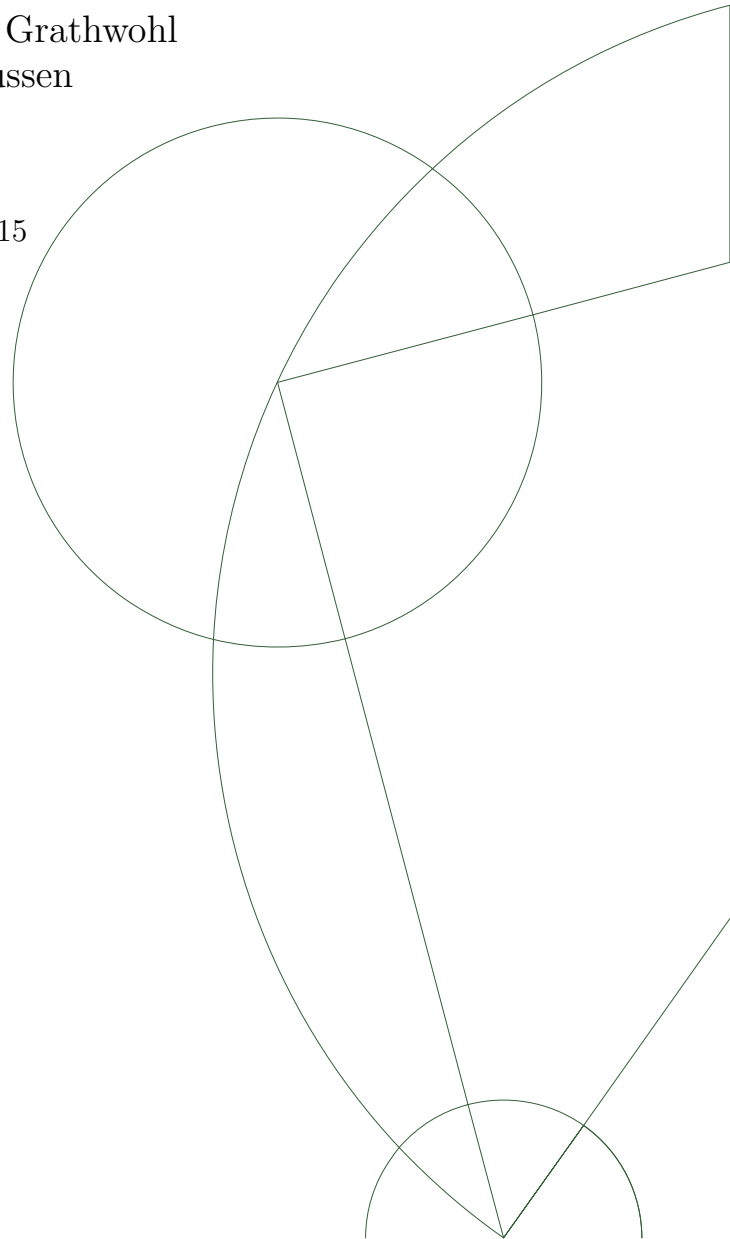
Supervisors
Rasmus Fonseca

Niels Bjørn Bugge Grathwohl

Ulrik Rasmussen

May 20, 2015

Contents

1 Nucleic Acids

((TODO: TELL WHY NUCLIEC ACIDS))

1.1 Ribonucleic acid

((TODO: ADD FIGURES, TALK ABOUT DIFFERENT KINDS OF RNA))

Ribonucleic acid (RNA) is a large molecule composed of nitrogenous bases nested on a ribose-phosphate backbone. The possible nitrogenous bases, or bases for short, that can be nested on the backbone are guanine (G), adenine (A), uracil (U) and cytosine (C). In nature, the predominant form of RNA are as a singlestranded chain that can fold in on itself, bundled with other chains to form a structure. This flexibility of the backbone that allows for the chain to fold in on itself is possible because the RNA uses a sugar called ribose in its backbone, which is unstable compared to its other form, deoxyribose, used in deoxyribonucleic acid (DNA), but is more flexible, allowing the RNA chain to bend in ways that DNA can not.

The bases found in RNA can form hydrogen bonds with each other, though not all bases can form bonds with each other. Bases that can bond with each other are guanine with cytosine, and adenine with uracil. These bonds are complementary of each other, and forms the structure of each RNA. When two bases bond with each other, they will stick to each other which changes the form of the RNA chain. However sometimes a base will have no complementary base to bond with, causing the base to stick out, giving rise to certain characteristic forms. This will be elaborated on later.

1.2 Deoxyribonucleic acid

((TODO: ADD FIGURES, TELL MORE ABOUT DNA))

Deoxyribonucleic acid (DNA) is a large molecule composed of nitrogenous bases nested on a deoxyribose-phosphate. DNA is mostly found in nature as helixes, where two strands has bonded. Similarly to RNA, DNA has four nitrogenous bases, and shares three of the four that RNA have, (guanine, adenine and cytosine). However instead of uracil, the fourth base is thymine (T).

1.3 Secondary Structures of Nucleic Acids

The secondary structure of a nucleic acid describes how the bases of the nucleic acid has bonded. The secondary structure of nucleic acids can change if the nucleic acid is damaged or has mutated, causing it to gain or lose bases. When two bases bonds they hold onto each other, causing bases that have no complementary base to bond with to stick out. Below are examples of three common secondary structures as well as a brief analysis on how to identify them.

Bulge

A bulge occurs when one or more bases have no base to bond with and these bases are surrounded by bases that have bonded. This causes the bases to get pushed out slightly, resembling a bulging growth. This type of structure occurs when one or more bases has been inserted or deleted, since if a base has been inserted then it will have no base to bond with, and if a base has been deleted then the previously-bonded base will have no base to bond with. Below is an example of a bulge:

> **Example 1.** *The RNA sequence* `AGCAGGCUAGCCGCU`. *Note the bulging* `A`.

Since we know that a bulge occurs when a base is inserted or deleted, we can define a bulge as follows:
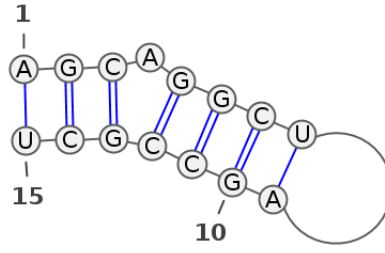
Figure 1: Example of a bulging A

Definition 1. *Let S be a bulge. Let E be a sequence of characters. Let Σ be an alphabet. Let $a \in \Sigma$. Let 0 be the empty string. Let Λ be a map that holds the complement to each base described in the alphabet. Let E^{-1} denote the reverse of the sequence. Then we can define a stem loop as follows;*

$$S = E' \ f(E^{-1}) \mid E \ f(E^{-1'})$$
$$E = \{a\} \ E \mid 0$$
$$E' = E \ E' \mid E$$
$$f(E) = \Lambda(E)$$

((NOTE: ABOVE DEFINITION NEEDS WORK, ISN'T CORRECT))

Interior Loop

An interior loop is when two or more opposing bases aren't complementary and can't bond, causing them both to bulge. This occurs when one or more consecutive bases mutate to another base. Below is an example of an interior loop:
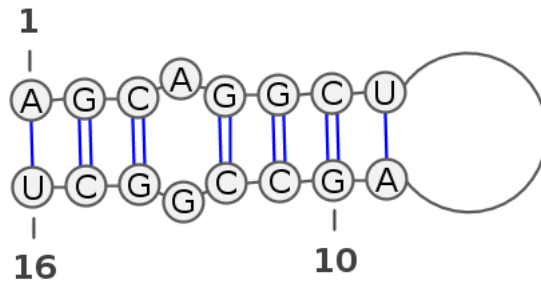

Figure 2: Example of an Interior Loop A-G

Example 2. *The RNA sequence* `AGCAGGCUAGCCGGCU`. *Note the bulging* `A` *and* `G` *creating a loop inside the bonded strand.*

These interior loops vary in size, and can have differing amount of bases on either side of the strands, making them hard to generalize.

((TODO: ADD A BETTER EXPLANATION AND A DEFINITION OF AN INTERIOR LOOP))

Stem Loop

A stem loop, also known as a hairpin loop, occurs when a strand bonds with itself, but leaves a sequence of bases sticking out that doesn't bond with anything. This kind of loop occurs typically in RNA as they are single-stranded, but may happen in DNA if the two strands of the DNA has been separated. Below is an example of a stem loop:
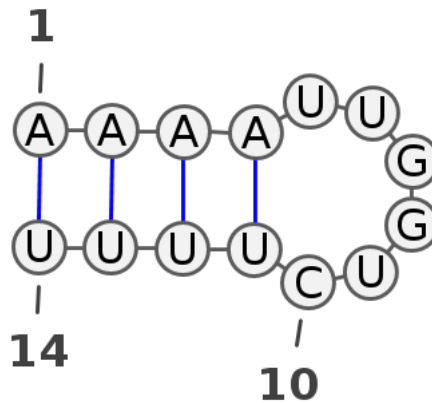

Figure 3: Example of a Stem Loop, UUGGUC

Example 3. *A stem loop of the RNA sequence* `AAAAUUGGUCUUUU`.

An important thing to take note of is how the sequence can be seen as one long strand that starts from the adenine bases that binds with the uracil bases, loops around without binding to anything and finally become the uracil bases that the adenine bases from the start binds with. This means that the stem loop can be written as one continuous sequence of bases; `AAAAUUGGUCUUUU`. This specific kind of loop adheres to a specific pattern that reflects from the example, that can be defined as follows:

Definition 2. *Let S be a stem loop. Let E_n be a sequence of characters, where if in E_n and E_m $m = n$, then $E_n = E_m$. Let Σ be an alphabet. Let $a \in \Sigma$. Let Λ be a map that holds the complement to each base described in the alphabet. Let E_n^{-1} denote the reverse of the sequence. Then we can define a stem loop as follows;*

$$S = E_0 E_1 f(E_0^{-1})$$
$$E_n = \{a\}$$
$$f(E_n) = \Lambda(E_n)$$

Since we can define a stem loop we can, with the right tools, search through a file documenting the bases of a nucleic acid and find all stem loops the nucleic acid has.

2 Scan_For_Matches

Scan_for_matches is a string-searching tool created by Ross Overbeek, David Joerg and Morgan Price in C which searches through text files. Users specify what they wish to search for by defining a pattern, and scan_for_matches returns all matches that corresponds to the specified pattern.

Definition 3. *Let* `E` *be any pattern that's in the alphabet Σ as defined in Example 4. Let 0 be the empty string. Let* `A` *be a string that we are processing to see if the pattern is valid. A pattern may then be constructed as such:*

Example 4. *Let Σ denote an alphabet. Then we can define a pattern as follows:*

`h`	*Match the sequence h, where $h \in \Sigma$*
`n...m`	*Match n to m characters where $n \leq m$*
`x=n...m`	*Match n to m characters, and call the sequence x*
`x	y`	*Match either x or y*
`x[n,m,l]`	*Match x, allowing for n mismatches, m deletions and insertions where $n,m,l \geq 0$*
`length(x+y) < n`	*The length of patterns x+y<n where $n > 0$*
`z={hl, lh}`	*Create a pattern rule where h is the complement of l, and l is the complement of h, where $h \in \Sigma$, $l \in \Sigma$, and call it z*
`<x`	*Match the reverse of pattern x*
`~x`	*Match the reverse complement of pattern x using the G-C, C-G, A-T and T-A pairing rule*
`z~x`	*Match the reverse complement of pattern x using pattern rule z*
`^ x`	*Match only pattern x if it is at the start of a string*
`x $`	*Match only pattern x if it is at the end of a string*

Definition 3 states that a pattern may be any combination of the alphabet defined in example 4. Using these patterns, it is possible to make very specific or very broad searches in a text file.

Example 5. *Say we want to write a pattern that finds the sequence* `GUUC`*, allowing one mismatch, followed by a random sequence which has a length between 3 and 5, followed by the reverse complement of the first sequence that we found. We can then write this as*

$$p1=\texttt{GUUC[1,0,0] 3...5 \textasciitilde p1}$$

Example 5 will match a stem loop as described in section 1.3. Note that if we wanted to find all stem loops in a file where the bonded bases are of length 4, we would replace `GUUC[1,0,0]` with an arbitrary sequence of characters by writing `p1=4..4 3..5 ~p1`.

3 Regular expressions

To explain what a regular expression is, we must first introduce languages and alphabets. All literals will be written using the typewriter font, to distinguish between literals and other text.

Definition 4. *An alphabet Σ is a finite set of letters.*

Definition 5. *A language is a subset of all strings, formed over Σ : $L \subseteq \Sigma^*$*

Example 6. *If we have a DNA sequence string, the alphabet Σ consists of the literals $\{\texttt{t}, \texttt{g}, \texttt{c}, \texttt{a}\}$. and the language contains strings formed by the literals from this alphabet. For example "gtcaaa" or "gtcaaat".*

Definition 6. *A regular expression is described by the following grammar:*

$$E ::= a|0|1|E_1 + E_2|E_1 E_2|E^*$$

where E_1 and E_2 are RE's and $a \in \Sigma$

Definition 7. *The language interpation L(E) of a regular expression is:*

$$L(0) = \emptyset$$
$$L(1) = \{\epsilon\}$$
$$L(\mathtt{a}) = \{\mathtt{a}\}$$
$$L(E_0 + E_1) = L(E_0) \cup L(E_1)$$
$$L(E_1 E_2) = \{w_1 w_2 | w_1 \in L(E_1), w_2 \in L(E_2)\} = L(E_1)L(E_2)$$
$$L(E^*) = \bigcup_{n=0}^{\infty} L(E)^n$$
$$L(E)^0 = \{\epsilon\}$$
$$L(E)^N = \underbrace{L(E)L(E)...L(E)}_{n \ times}.$$

[1, p.5 def. 3]

With definition 7, we can now form regular languages.

Example 7. *Natural numbers can be described as a regular expression. Natural numbers have the alphabet $\Sigma = \{\mathtt{0,1,2,3,4,5,6,7,8,9}\}$, the regular expression for natural numbers would look like:*

$$E_{nat} = (1+2+3+4+5+6+7+8+9)(0+1+2+3+4+5+6+7+8+9)^*$$

References

[1] Niels Bjørn Bugge Grathwohl& Ulrik Terp Rasmussen Fritz Henlein. A crash-course in regular expression parsing ans regular expressions as types.

[2] Kolman, Busby, and Ross. *Discrete Mathematical Structures, sixth edition.* Pearson, 2009.

[3] Gunnar Forst. Noter til kombinatorik og grafteori. 2, 2006.

[4] William Cherowitzo. Graph theory lecture notes 6. `http://www-math.ucdenver.edu/~wcherowi/courses/m4408/gtln6.htm`, January 2015.

[5] Wikipedia. Chromatic polynomial. `http://en.wikipedia.org/wiki/Chromatic_polynomial`, January 2015.

[6] Wikipedia. Ribonucleic acid. `http://en.wikipedia.org/wiki/RNA`, May 2015.

[7] Wikipedia. Deoxyribonucleic acid. `http://en.wikipedia.org/wiki/DNA`, May 2015.

[8] Wikipedia. Nucleic acid secondary structure. `http://en.wikipedia.org/wiki/Nucleic_acid_secondary_structure`, May 2015.

[9] Ville Laurikari. Efficient submatch addressing for regular expressions. November 2001.

[10] Torben Ægidius Mogensen. *An Introduction to Compiler Design.* Springer, 2001.

[11] Dan Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press.

[12] Ville Laurikar. Nfas with tagged transitions, their conversion to deterministic automata and application to regular expressions. 2000.

[13] R. A. Baeza-Yates & G. H. Gonnet. Fast text searching for regular expressions or automaton searching on tries. *ACM, vol. 43*, 1996.

[14] Pang Ko & Sromovas Aluru. Suffix tree applications in computational biology.

[15] Wikipedia. Levenshtein automaton. `http://en.wikipedia.org/wiki/Levenshtein_automaton`, April 2015.

[16] Russ Cox. Regular expression matching can be simple and fast (but is slow in java, perl, php, python, ruby, ...). January 2007.

[17] Ville Laurikari. Tre documentation. `http://laurikari.net/tre/documentation`.

[18] Mohammadreza Ghodsi. Approximate string matching using backtracking over suffix arrays. 2009.