Appendix 1  Formal Model

Here we present a formal model of elementary repeats, in preparation for a correctness proof for the phRAIDER algorithm.

Definitions and Terminology

Sequence Descriptors

Definition 1 A sequence descriptor is a string over $\Sigma = \{A, C, G, T, *\}$; a sequence descriptor of length $l$ will be referred to as an $l$-mer.

The four characters represent nucleotides, while the $*$ represents a single wild-card character.

Definition 2 A sequence descriptor $r$ and a nucleotide sequence $g$ of the same length are consistent if, for any position $i$ in which they differ, $r_i = *$.

Definition 3 The frequency of a sequence descriptor $r$ in genomic sequence $G$, denoted $\nu_G(r)$, is the number of positions $i$ such that the substring $G_{i:i+|r|}$ is consistent with $r$.

Sequence Descriptors and Spaced Seeds

We define our spaced seeds to be equivalent to those used in PatternHunter, then extend our definition of elementary repeats to incorporate them [1].

Definition 4 Given a spaced seed $s$ of length $l$ and an $l$-mer $r$, we generalize $r$ w.r.t. $s$ by replacing $r_i$ with a $*$ for all $i$ such that $s_i = 0$. 

$^*$Contributed equally

$^\dagger$Corresponding Author (karroje@miamiOH.edu)
Example: We generalize AACAGA w.r.t. 110101 to $AA * A * A$.

**Definition 5**  Let $s$ be a spaced seed of length $l$.

- We say $s$ and an $l$-mer $r$ are consistent if, when $r_i = *$, $s_i = 0$.
- We say seed $s$ hits sequence descriptor $r$ at position $i$, $i \leq |r| - |s|$, if $s$ is consistent with the substring $r_{i+|s|}$.
- We say seed $s$ covers sequence descriptor $r$ if, for every $j$, $0 \leq j < |r|$, there is an $i$ such that $s$ hits $r$ at $i$ and overlaps position $j$ in doing so. (That is, $i \leq j < i + |s|$.)

Example: For $s = 11011$ and $r = AAAAA * TTTTT$, $s$ covers $r$, hitting at $\{0, 3, 6\}$. $s$ does not cover $AAAAA * TTTTT$, as there is no position at which $s$ hits and extends over $j = 5$.

**Definition 6**  Let $r$ be a sequence descriptor and $s$ be a spaced seed that covers $r$.

- The coordinate decomposition of $r$ w.r.t $s$ is the set of all $i$ such that $s$ hits $r$ at position $i$.
- The sequence decomposition of $r$ w.r.t. $s$, denoted as $\Delta_s(r)$, is the set of all generalized (w.r.t. $s$) sequence descriptors $r_{i:i+|s|}$ for each $i$ in the coordinate decomposition.

Example: For $s = 11011$ and $r = TAAAAA * CCCCCC$, we see that $s$ hits at coordinates 0, 1, 4, 7, and 8 – and every character in $r$ is within $|s| = 5$ bases of one of those spots. Thus the sequence decomposition is $\Delta_s(r) = \{TA * AA, AA * AA, AA * CC, CC * CC\}$. (Note that two substrings will generalize to $CC * CC$, but as this is a set we include only unique instances of each string.)

**Elementary Repeats**

**Definition 7**  Given a fixed seed $s$, genome $G$, and frequency $f$, a sequence descriptor $r$ is an elementary repeat descriptor if the following holds true:

1. Length requirement: $s$ covers $r$.
2. Frequency requirement: $\nu_G(r) \geq f$.
3. Minimality requirement: $\nu_G(w) = \nu_G(r)$ for all $w \in \Delta_s(r)$.
4. Maximality requirement: There does not exist an repeat element descriptor $r'$ such that $\Delta_s(r) \subset \Delta_s(r')$.

**Definition 8**  An elementary repeat family is a set $S$ of genome coordinates such that there exists a sequence descriptor $r$ where:

- $r$ is consistent with the string $G_{i:i+|r|}$ for every $i \in S$.
- $r$ is not consistent with any substring $G_{j:j+|r|}$, $j \notin S$.

We will also refer to a repeat element family as the multi-set of sequences at the corresponding coordinates, as opposed to the set of coordinates. Such use will be clear from context.
Appendix 2  Equivalence proof: Z&L v. Figueroa

In the Figueroa thesis [2] it was proved that our definition of elementary repeats was equivalent to the Z&L definition [3]. Here we prove that our definition is a generalization of the Figueroa definition. Specifically, that when a seed contains no 0 symbols, the two definitions are equivalent.

Recall that for sequences $x$ and $y$, $x \circ y$ is defined only when $|x| = |y|$ and the $|x| - 1$ length prefix of $x$ is a suffix of $y$, and is defined by merging the two by that shared string. (e.g. $ACCC \circ CCCG = ACCCG$.) Given that, we can review the Figueroa definition:

**Definition 9 (Figueroa Elementary Repeats)** Given a genomic sequence $G$ and integers $f$ and $l$, a sequence $r$ is an elementary repeat if we can write $r = x_1 \circ x_2 \circ \cdots \circ x_k$ where $|x_i| = l$, $k = |r| - |l|$, and:

1. $k \geq 1$. (There is at least one $l$-mer substring).
2. $r$ appears at least $f$ times in $G$.
3. $\forall i : \nu_G(x_i) = \nu_G(r)$
4. There does not exist any $y$, $|y| = l$, such that either $y \circ r$ or $r \circ y$ meets conditions 1-3.

**Theorem 1** Consider a genome $G$, an integer $f$, and a seed $s$ with no zero characters. $r$ is an elementary repeat descriptor for $s$ (by Definition 7) if and only if $r$ is an elementary repeat for $l = |s|$ (by Definition 9).

**Proof:**

$(\Rightarrow)$ First we observe that if $s$ has no zeros, and covers $r$, then $r$ has no * characters. From this it follows that $\Delta_s(r) = \{r_0, r_1, \cdots, r_k\}$ where $|r_i| = l$ and $k = |r| - l$. That is, the sequence decomposition contains every $l$-mer of $r$. From this it follows that $r = r_0 \circ r_1 \circ \cdots \circ r_k$.

Now we prove each of the four conditions of Definition 9:

1. Since $r$ is covered by $s$, $|r| \geq |s|$, so $k = |r| - |s| + 1 \geq 1$.
2. Since $r$ contains no * symbols, the only string it is consistent with is itself. Meaning it has the same frequency by either definition. Hence (2) of the Figueroa definition follows directly from (2) of our definition.
3. Follows immediately from the fact that every $r_i$ is a member of $\Delta_s(r)$, and hence must have the right frequency.
4. Argue by contradiction: suppose there were such a $y$ (assume w.l.o.g. it follows $r$), and let $r' = r \circ y$. Since the Figueroa definition tells us $r'$ matches a string around every occurrence of a match to $r$, this means $r'$ is (trivially) consistent to a substring around every occurrence of a match to $r'$. Thus $\Delta_s(r) \subset \Delta_s(r')$, thus $r$ could not have been a repeat element descriptor.

$(\Leftarrow)$ The above argument is completely reversible. □
Appendix 3  phRAIDER algorithm

Supporting Lemmas and Observations

To prove the algorithm correct, we need the following. For each, assume (without loss of generality) a genome $G$, any fixed seed $s$, and any fixed frequency threshold $f > 1$.

**Lemma 1** Let $r$ be an elementary repeat descriptor covered by $s$. For any $d \in \Delta_s(r)$ there exists exactly one substring of $r$ that is consistent with $d$.

**Proof**: By the definition of sequence decomposition, there is at least one such substring $r'$. (Recall that since $r$ can be decomposed by $s$, $|r| \geq |s| = |d|$.) If there were multiple $r'$, then necessarily $\nu_G(r) < \nu_G(d)$, contradicting the minimality requirement of repeat elements descriptors. (This last part is true because $d$ is consistent with every instance of $r'$ in $G$.) □

**Lemma 2** Let $r$ be an elementary repeat descriptor that occurs $m$ times in $G$, let $d_1, d_2, \ldots, d_k$ be the elements of $\Delta_s(r)$ ordered by unique location in $r$, let $d^j_i$ be the $j$th instance of a substring of $G$ consistent with $d_i$, and let $c(d^j_i)$ be the index of the base number at which $d^j_i$ starts in $G$. Then $c(d^j_i) - c(d^j_{i-1}) = c(d^j_1) - c(d^j_1)$ for all $1 \leq i \leq k, 1 \leq j \leq m$.

**Proof**: $d_1$ must be consistent with the length $|s|$ prefix of $r$, else the first base of $r$ would be left uncovered in the decomposition. For $i > 1$, $d_i$ is consistent with a unique substring of $r$ (by Lemma 1), say at the $m$-th base of $r$. Or put otherwise, it is offset from the beginning of $r$ by $b$ bases. Now take any substring of $G$ consistent with $r$ in the genome, and $d_1$ must be consistent with the prefix of that substring, and $d_i$ must be offset from that substring by $b$ bases. The result follows directly. □

**Lemma 3** Using the notation from before, for $j > 1$, $c(d^j_i) - c(d^j_{i-1}) \leq |s|$.

**Proof**: Otherwise, there would be a base position in $r$ not covered by some $d \in \Delta_s(r)$, meaning that $\Delta_s(r)$ is not a decomposition.

Pseudocode

In (appendix) Table 1 we give a pseudocode description of the phRAIDER algorithm. Here we list a few of the core principles on which the algorithm operates. Assume for this discussion a fixed seed $s$ of length $l$, genome $G$, and frequency threshold $f$, have been supplied as parameters to the algorithm.

1. If $r$ is a repeat element descriptor w.r.t. to $f$, then it is also one w.r.t. $f = 2$. So we set $f$ to 2 for the duration of the algorithm, then filter out those missing the frequency requirement (line 36).

2. We are scanning the genome base-by-base (line 6). At coordinate $i$ we look at the length $|s|$ substring starting at $i$, generalize it (line 9), and update the hash table to create $v$. We now need to decide if $v$ is a member of some $\Delta_s(r)$, and add it to the appropriate family if so.
3. A family $F$ is a collection of generalized $l$-mers that is believed at any point in time to be a single decomposition set $\Delta_s(r)$ for some $r$ (until later information refutes this). As we scan the genome, we will build up different families (line 18), then course-correct in line 24 if we discover a violation of the maximality requirement.

4. The `BelongsToFam` function identifies whether a given $v$ is a member of a family $F$ by exploiting lemma 2. If we are looking at a generalized $l$-mer $v$ and want to know if it is a member of $F$, where $d_1$ is the first $l$-mer in $F$, then: $v$ must have occurred the same number of times as $d_1$, and it must be a fixed offset from $d_1$.

5. $Q$ is a sorted list of the families we have encountered in the last $l$ bases. When we move to a new base we pop off the first family in $Q$ if it is now out of range (line 8), and whenever we encounter an $l$-mer already associated with a family we push that family to the back of the list to maintain the sort. (Note, for run time bounds, that $Q$ never contains more than $l$ elements.) It follows from lemma 1 that if we encounter an $l$-mer $v$ such that $v$ should be added to $F$, but is not as the first member, then $F$ must be in $Q$ at the time (justifying line 12).

6. Let $r$ be an $l$-mer, and consider the second occurrence of $r$ in $G$.
   
   (a) It is easy to show that $r$ is either a repeat descriptor or belongs to a larger repeat descriptor. (Recalling that $f = 2$.) Hence the conditional at line 11, where we either add it to an existing family or make it its own family.
   
   (b) Looking at it the other way: when we encounter the second instance of repeat descriptor $r'$, we will add all members of $\Delta_s(r')$ to some family $F$, and never add to $F$ again.

7. Let $v$ be an $l$-mer has been encountered $k > 2$ times. Then either:
   
   (a) $v$ is the first element of some family $F$. We need to move $F$ to the back of $Q$ (as its now the most recently seen family). (Line 21)
   
   (b) $v$ is “out of place”. It was assigned to some family, but has now been encountered somewhere inconsistent with that assignment. Meaning that it violated maximality. It needs to be spliced out, and either start its own family or extend a family that was recently the recipient of a splice. (Line 23)
   
   (c) $v$ is a member of a family and exactly where it should be, in which case we just need to update maintenance information. (Line 32)

Correctness

**Theorem 2** Upon termination, there is a one-to-one mapping between the repeat element families in $G$ (w.r.t. $s$ and $f$) and the families created by `phRAIDER` such that for each repeat element descriptor $r$, the $l$-mers in the corresponding family $F$ are the $l$-mers in $\Delta_s(r)$.

In other words: `phRAIDER` as described exactly finds elementary repeats. The proof of this is somewhat tedious, but here we sketch the main ideas.

**Sketch of proof:**

First: assume $r$ is a repeat element descriptor, and let $\Delta_s(r) = \{d_1, d_2, \ldots, d_k\}$, ordered by occurrence in $r$. We want to show there is a unique $F$ that consists of exactly those $k$ $l$-mers.
• There is some $F$ that contains all the $d_i$. Because they exist sequentially in the genome at every instance of $r$, they will be grouped together at their second occurrence in the invocations of the block at line 11. Then minimality ensures they will stay together – if one is spliced in line 24, the rest will be spliced into the same family.

• By the end, that $F$ does not contain any other $d$. If it contained a $d \not\in \Delta_s(r)$, then $d$ must not be a member of $F$ because of the minimality and maximality requirements – it does not occur the same number of times as the other $d_i$. If it were ever added to $F$, then either it must be spliced out in line 24, or all of the $d_i$ must be spliced out form their own family.

• $F$ must be the only family that holds any of the $d_i$, as it holds all of them, and there is no mechanism in the algorithm for adding an $l$-mer to two families.

Second: consider the final contents of some family $F$. We need to show that the $l$-mers assigned to $F$ for the set $\Delta_s(r)$ for some repeat descriptor $r$. We also need to show that this is the only $F$ for which that is true for that $r$. Its easy to create an $r$ from $F$’s $l$-mer set such that the decomposition of $r$ is that set. Now we show the following:

• $r$ meets the length requirement: $\Delta_s(r)$ exists, so by definition $s$ covers $r$.

• $r$ meets the frequency requirement: If it did not, then none of the $d_i$ would meet the frequency requirement and all of them would have been filtered in line 36.

• $r$ meets the minimality requirement: If not, that means one of the $d_i$ occurred independently. But then $d_i$ would either not have been added into $F$ (line 11) or it later would have been spliced out (line 24).

• $r$ meets the maximality requirement: If not, some $d \not\in F$ would have been added to $F$, as it occurs the right number of times and has the right offset (thus violating maximality).

• $r$ does not describe any other family $F'$. If it did, then $d_1$ would have to be in both families, which is not possible.

Asymptotic Runtime Bound

For a given seed $s$ and genome $G$, let $|s|$ be the character length of $s$ and $n$ be the base length of $G$. Looking at the pseudocode, we see the loop at line (6) will execute $O(n)$ times. Assuming $H$ is implemented as a hash table and $Q$ as a linked list:

• Line 7-8: requires $O(1)$ time per iteration.

• Line 9: requires $O(|s|)$ time per iteration.

• Line 10: requires $O(1)$ time per iteration.

• Lines 12-18: Line (12) requires $O(|s|)$ per iteration (as we $Q$ can never get longer than $|s|$). The remainder are all constant time operations.

• Lines 21-31: Again, line (25) requires searching $Q$, and hence is $O(|s|)$ per iteration. Lines (24) and (27) can be done in constant time given properly constructed data structures, and all other lines are obviously $O(1)$. 

Table 1: Seeds used in Table 1 of the paper. Seed index is arbitrary. Weight is the number of 1 characters; length the total number of characters; density is the weight-to-length ratio. The notation $1^n$ indicates a string of $n$ consecutive 1 characters.

| Index   | Seed                                      |
|---------|-------------------------------------------|
| 70      | $1^70017^1501^401^401^4$                   |
| 154     | $1^401^401^401^411001100101^3001^41101^4$ |
| 262     | $1^3011001110101^3110108^0101101^31^3001^310110^3$ |
| 480     | $1^4501^30^31^30^30101^310^31^30^3010^30^3001^301^3001^3$ |
| 489     | $11^410^31^40^30^31010011010110101^31010^31^3001^3101100101^31101$ |
| 508     | $1011011010^41100110011001^3010^401^3110^410^31^30^30^30^30^3001101$ |
| 512     | $1^512$ (weight = 32, length = 76, density = 0.42) |
| 516     | $1^401^3010^31^41010110011001^3010^310^310^31^30^30^30^3101101$ |
| 536     | $1^536$ (weight = 32, length = 78, density = 0.41) |
| 537     | $1101^31100110^30^30^30110^30^30^30^30^30^30^30^30^30^30^30^3101100101^31101$ |
| 545     | $1^545$ (weight = 32, length = 78, density = 0.41) |

- Lines 33-35: Clearly all constant time per iteration.
- Lines 36: The worst-case requires scanning all bases of all families – a value considerably less then $n$. In practice it will do much less, but this still leaves us with a runtime of $O(n)$.

This results in an $O(n \cdot |s|)$ algorithm. As $s$ is generally fixed and fairly small, the algorithm is essentially linear in the size of the genome.

### Appendix 4 Supplementary Results

In Table 1 of the document we list a number of results by seed index. The seed indices were assigned arbitrarily. (While it would have made sense to use the seeds binary representation, the seed lengths being used would have resulted in very large indices.)
Algorithm phRAIDER

1: function BelongsToFam(F, v)  
2:    return |H[v]| == |H[F.first]| and  
3:    H[v].first - H[F.first].first == H[v].last - H[F.first].last  
4: function phRAIDER(Genome G, Seed s, MinFrequency f)  
5:    Queue<Family> Q;  
6:    for i ← 1 → |G| - |s| do  
7:        if i - front(Q).last_seen > |s| then  
8:            F = Q.dequeue(); F.last_seen = -∞  
9:            v ← generalize(G[i], x)  
10:           H[v].push(i)  
11:        if |H[v]| == 2 then  
12:            F = arg_find_{F ∈ Q}{BelongsToFam(F, v)}  
13:                if F found then  
14:                    F.addLmer(v)  
15:                    F.last_seen = i  
16:                       Q.enqueue(F) # Bump F to end of Q  
17:                else  
18:                    Q.enqueue(new Family(lmer = v, last_seen = i))  
19:        else if |H[v]| > 2 then  
20:            F ← v.family()  
21:                if v == F.first then  
22:                    F.enqueue(v)  
23:            else if F.last_seen == -∞ or not BelongsToFam(F, v) then  
24:                splice(F, v) # Remove v from l-mer list  
25:                    F' = arg_find_{F ∈ Q}{BelongsToFam(F, v)}  
26:                if F' found then  
27:                    F'.addLmer(v)  
28:                    F'.last_seen = i  
29:                       Q.enqueue(F')  
30:            else  
31:                Q.enqueue(new Family(lmer = v, last_seen = i))  
32:        else  
33:                Q.enqueue(F)  
34:            F.setLastLmer(v)  
35:            F.last_seen = i  
36:        cleanup() # Remove families that occur less than f times.
References

[1] Li, M., Ma, B., Kisman, D., Tromp, J.: PatternHunter II: highly sensitive and fast homology search. Genome informatics. International Conference on Genome Informatics 14, 164–175 (2003)

[2] Figueroa, N., Liu, X., Wang, J., Karro, J.: RAIDER: Rapid Ab Initio Detection of Elementary Repeats. In: Advances in Bioinformatics and Computational Biology, pp. 170–180. Springer, Cham (2013)

[3] Zheng, J., Lonardi, S.: Discovery of Repetitive Patterns in DNA with Accurate Boundaries. Fifth IEEE Symposium on Bioinformatics and Bioengineering (BIBE’05), 105–112 (2005)