Periodicity and Self-Overlap

Casey S. Schroeder

May 11, 2013

email: cs@csschroeder.com tel: 312-546-3048

Abstract

It is easily shown that self-overlap can be defined in terms of periodicity, and visa versa, should one allow for semi-periodicity. We show that the *the minimal period* is also inter-definable with the concept of self-overlap (and the general form of periodicity). By *the minimal period* we mean the $|p|$ such that $S = p^q$ for minimal length prefix p and $q \in Q^+$. It is clear that we can define the minimal period in terms of (maximal) self-overlap or (least) periodicity. However, it is less clear that self-overlap and semi-periodicity can be defined, in general, in terms of minimal periodicity alone. We give two recursive definitions of self-overlap in terms of minimal periodicity and basic string operations. We show that these definitions provide two simple linear time algorithms for finding self-overlap and semi-periodicity, one of which is optimal given a step within the preprocessing step of the Knuth-Morris-Pratt algorithm, running in time linear to the length of the string.

1 Introduction

The definition of *self-overlap* may be formally given as:

Definitions 1. *A string $S =< a_1, ..., a_k >$ is* ***self-overlapping at an index*** *i or* ***self-overlapping with length i****, if the sub-strings $< a_1, ..., a_i >$, $< a_{k-i+1}, ..., a_k >$ are identical, i.e. the prefix of length i is equal to the suffix of length i.*

We will understand a string raised to a power, perhaps fractional, as the number of times it is concatenated with itself, to the right. A sufficient definition of *a period* and *periodicity* is then given as,

Definitions 2. *For any $S =< a_1, ..., a_k >$ and $0 < i < k$, if $S = g^q$ for $g =< a_1...a_{k-i} >$ and $q = k/(k - i)$, then $|g| = k - i$ is* ***a period*** *of S. The set of all periods of S is the* ***periodicity*** *of S.*

Note now that the following equivalences also hold, and could also be used to define self-overlap.

Equivalence 1. *$S =< a_1, ..., a_k >$ overlaps with itself at index i if and only if $S = g^q$ for $g =< a_1...a_{k-i} >$ and $q = k/(k - i)$.*

This being for the fact that we allow fractional periodicity ($1 \leq q < 2$) in our definition of periodicity. So it is clear that self-overlap and periodicity are inter-definable concepts, namely.

Equivalence 2. *$S =< a_1, ..., a_k >$ overlaps with itself at index i if and only if $k - i$ is* ***a period*** *of S.*

This broad notion of 'periodicity', including as it does the full notion of semi-periodicity, is the direct complement of self-overlap. Semi-periodicity is not always what we consider the proper notion of a period. For example, the following string has a minimal period of length 5

```
aabba|aabba|aabba|aabba|aa
```

multiples of this length are also periods, i.e. 10, 15, and 20, but we cannot cover the full set of self-overlapping indices with multiples of this minimal period alone. Rather we need to include a semi-period of *21*

```
aabbaaabbaaabbaaabbaa|a
```

Similarly, the following has a minimal period of length 9,

```
ababaabba|ababaabba|ababaabba|ababa
```

and likewise, periods of 18 and 27; but also must include semi-periods of *29* and *31*.

2 The Self-Overlap Theorem 1

We will here show that a definition of self-overlap (and semi-periodicity) can be given in terms of the minimal period alone.

Definitions 3. *For any S of length k and maximal index of self overlap i, if $S = p^q$ for $p =< a_1 ... a_{k-i} >$ and $q = k/(k-i)$, then $|p| = k - i$ is* ***the minimal period*** *of S.*

To understand the adequacy of this definition, consider the following relationship:

$$\begin{array}{l}
a_1 \ldots\ldots\ldots\ldots a_{k-i+1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots a_k \\
\qquad\qquad a_1 \ldots\ldots\ldots\ldots a_{k-i+1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots a_i \ldots\ldots\ldots\ldots a_k \\
\qquad\qquad\qquad\qquad a_1 \ldots\ldots\ldots\ldots a_{k-i+1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots a_i \ldots\ldots\ldots\ldots a_k \\
\qquad\qquad\qquad\qquad\qquad\qquad \ldots \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \ldots \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \ldots
\end{array}$$

And convince yourself that if there is a minimal period of $k - i$, then there will be this overlap relationship, and if there is this overlap relationship, then $k - i$ is the minimal period.

Self-Overlap Theorem 1. *The indices of self-overlap for S are given by the set $SOver(S)$, defined in terms of the minimal period $|p|$ of S, where $q = |S|/|p|$, as*

$$SOver(S) = \{i \neq 0 : i = |S| - |p| \vee i \in SOver(p^{q-1})\} \tag{1}$$

Where p^{q-1} for $q - 1 = 0$ is understood to be the empty string and $SOver('') = \{\}$.

Note that we assume that p and q may change in our recursive calls to SOver. This makes the relationship very easy to prove. There cannot be self overlap at indices from k to $i + 1$ or p would not be minimal (i maximal), so all self overlap must occur from i to 1. If we add i to the set, all additional self overlap is equivalent to self overlap for a shorter string, precisely p^{q-1} with length $i = |p| * (q - 1) = |S| - |p|$, so we can evaluate this string with *SOver* to get the remaining indexes of self overlap. Using the diagram it is easy to see that after adding i to the set, all additional self overlap must occur on $a_{k-i+1} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots a_k = p^{q-1}$, which is the string occurring at both the front and the back of our initial string.

3 The Self-Overlap Theorem 2

The basic definition above assumes that we must make a recursive call for each shortening by the minimal period. If we know the length of the string in advance, we can do better. The considerations of the first section indicate that the oddities to defining self-overlap in terms of the minimal period occur at the end of the string. This suggests the following piecewise definition of the set of indexes of self-overlap in terms of the periodicity of S.

Self-Overlap Theorem 2. *For any S with length k and minimal period $|p|$, the indices of self-overlap for S are given by the set function $SOver(S)$, where $q = |S|/|p|$ and n and f are the integer and fractional part of q respectively:*

$$q = 1 : SOver(S) = \{\} \tag{2}$$
$$1 < q < 2 : SOver(S) = \{k - |p|\} \cup SOver(p^f) \tag{3}$$
$$2 \le q : SOver(S) = \{k - i|p| : 0 < i < n\} \cup SOver(p^{(1+f)}) \tag{4}$$

Formally, p may be calculated from S and $|p|$ by basic string operations, as the prefix of S of length $|p|$.

This essentially provides an analytic treatment of what would otherwise be many recursive calls. We must show that SOver returns all and only the indices of self overlap for any S.

Proof of Clause 1. *For the first case where $q = 1$, the definition is the same as the previous definition so there is nothing further to show.*

Proof of Clause 2. *For the second case, where q is in the range $1 < q < 2$, the definition is the same as the previous definition so there is nothing further to show.*

Note: 1. *We note, however, that the recursive call to SOver in the second clause is necessary and does not represent a special case to which some trick may be applied. To show this, we can note that p^f could be anything (at least without adding constraints to our underlying alphabet). Consider only the following pattern:*

$$p = \{A - Y\}^n + Z \tag{5}$$

which is intended to say, set p equal to any string of some length n constructed from letters A to Y, and concatenate it with the letter Z. Now consider,

$$S = p^{(1+n/(n+1))} \tag{6}$$

This string could have any string as a fractional suffix, failing one containing Z, so (with no constraints on our alphabet) there is nothing else that can be done but to ask about the self-overlap of this suffix as well.

Proof of Clause 3. *For the sub-case where q is in the range $2 \le q < 3$, this definition is also the same as our previous definition. It remains to show that for $3 \le q$ all further indexes of overlap are restricted to be j, such that $j < |p|(1 + f)$.*

*First it is clear that if p is minimal in a string $S = p^{(n+f)}$, then (left) concatenating an additional p to obtain **the minimal extension** of S, will yield an $S+ = p^{(n+1+f)}$ with the same minimal period $|p|$. This is enough to show, that if $S+$ is a minimal extension of some $S = p^{(n+f)}$, then there will be no j, such that $S+$ self overlaps at $j > |p|(n+f)$, otherwise p would not be minimal in $S+$. So it is clear that we can recursively strip* *the same p until it could no longer be the minimal period of the resulting string, and all indices of overlap must occur on the minimal period intervals according to clause 3.*

What we will show is that for any $S = p^{(n+f)}$ for $n \geq 3$, S is a minimal extension of $S' = p^{(n-1+f)}$. Therefore, all indices of overlap for any $S = p^{(n+f)}$ with $n \geq 2$ has indices of self-overlap only at it's minimal period intervals except possibly at the indices $j < |p|(1+f)$, so one can strip the same minimal period from any S until $S* = p^{(1+f)}$ and not change the minimality of p. So clause 3 will apply to any such S.

Lemma on Periods 1. *For integer $n \geq 3$, proper fraction f, and any p: if $|p|$ is the minimal period of $S = p^{(n+f)}$, then $|p|$ is the minimal period of $S' = p^{(n-1+f)}$.*

Suppose that a string $S = p^q$ exists with $q \geq 3$, which is not a minimal extension. Then there exists a string $S' = p^{q-1}$ which has a different minimal period $|r|$. Then $|r| < |p|$ because $|p|$ is clearly still a period of S. If $|r|$ evenly divides $|p|$, then $|r|$ must be a period of $|S|$ as well, and hence $|p|$ is not minimal in $|S|$, so $|r|$ must not evenly divide $|p|$. S' still contains $q-1 \geq 2$ copies of p, so the following relationship exists in S':

$$\begin{array}{l}
a_1 \cdots\cdots\cdots\cdots a_{|p|+1} \cdots\cdots \\
\hspace{4.5em} a_1 \cdots\cdots\cdots\cdots a_{|p|+1} \cdots\cdots
\end{array}$$

Where $a_{|p|+1}$ is left of the center of S'. S' has a minimal period smaller than $|p|$, so we have the following relationship too:

$$\begin{array}{l}
a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{|p|+1} \cdots\cdots \\
\hspace{3em} a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{|p|+1} \cdots\cdots
\end{array}$$

Combining these, we have the following relationship:

$$\begin{array}{l}
a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{|p|+1} \cdots\cdots \\
\hspace{8.5em} a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{|p|+1} \cdots\cdots \\
\hspace{3em} a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{|p|+1} \cdots\cdots
\end{array}$$

Now we have not claimed any restrictions on $|p|/|r|$, but that $|p|\%|r| > 0$, so imagine we overlap S' until the difference between $n|r|$ and $|p|$ is $|p|\%|r|$ as follows:

$$\begin{array}{l}
a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{n|r|+1} \cdot\cdot a_{|p|+1} \cdots\cdots \\
\hspace{10.5em} a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots\cdots\cdots a_{|p|+1} \cdots\cdots \\
\hspace{3em} a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots\cdots\cdots a_{|p|+1} \cdots\cdots \\
\hspace{3.7em} \ldots \\
\hspace{4.5em} \ldots \\
\hspace{8em} a_1 \cdots\cdots a_{|r|+1} \cdots\cdots\cdots a_{n|r|+1} \cdots a_{|p|+1} \cdots\cdots
\end{array}$$

You can see that our second string copy overlaps our final string copy at an index of overlap of $k - (|p|\%|r|)$, which must be greater than $k - |r|$, and therefore $|r|$ cannot *be the minimal period of S' contrary our assumption.*

Note: 2. *Note that including the full period (1) as well as the fractional part (f) in the recursive call to SOver in the third clause is necessary. Consider only the string $S' = aaaabbaaaaabbaaaa$. If we divide by the minimal period aaaabba, we have aaaabba|aaaabba|aaa. So S' is an extension of the string aaaabbaaaa by the period aaaabba, but aaaabba is not the minimal period of aaaabbaaaa (which is aaaabb). So $S' = |p|^{2+(3/7)}$ has minimal period of aaaabba, but is not the minimal extension of any string. You can think of this as: until a period repeats in full it's "real" minimal period is under-determined. For this reason, we must make our recursive calls to reevaluate the period for $p^{(1+f)}$. Though perhaps this full recursive call is non-optimal and the string $p^{(1+f)}$ can be further reduced.*

This completes the proof. We only remind the reader that the periods of S are simply $k - i$ for each i in $SOver$.

4 Relationship to Knuth-Morris-Pratt

The general problem of finding the indices of self-overlap of a string can be solved by brute force in $O(n^2)$ time. One might clearly expect that string matching methods like that found in Knuth-Morris-Pratt (KMP) [4] can improve on this. KMP has two steps to solve the general problem of finding all occurrences of one string in another. The first is the preprocessing step, which amounts to the construction of a *Next* table of values which tells the algorithm where to begin again if a mismatch is found. As applied to the problem of finding the indices of self-overlap, a step to creating this *Next* table is the creation of a table f, which effectively finds the maximal self-overlap of all the prefixes of the string. This table f we will call the *MSO* table.

The basic algorithm is as follows:

```
def table(word):
    pos=2
    cnd=0
    MSO=list()
    for c in word: table.append(0)
    table[0]=-1
    table[1]=0
    count=0
    while (pos<len(word)):
        if word[pos-1]==word[cnd]:
            cnd=cnd+1
            MSO[pos]=cnd
            pos=pos+1
        elif cnd>0:
            cnd=MSO[cnd]
        else:
            MSO[pos]=0
            pos=pos+1
        count=pos
    return table
```

This will produce, for example, the following array for string N="aabaaabaabaaabaabaaa"

MSO=[-1, 0, 1, 0, 1, 2, 2, 3, 4, 5, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]

For the purposes of their algorithm, this is used to indicate that if there is a failure while matching the string N against another longer string, M, at a given index i of N, the matching can start again at the current position in M, and at index MSO[i] of N (MSO[0]=-1 indicating to advance N against M and begin matching from i=0). The rest of their pre-processing is for the sake of making greater shifts in N against M.

To make use of this algorithm for our purposes, we feed the algorithm a modified string, e.g. "aabaaabaabaaabaabaaa*" which contains the special character '*' that is not otherwise in our alphabet. Feeding this string to the algorithm, we also get the maximal self-overlap of our string of interest, "aabaaabaabaaabaabaaa" itself, in the final position. We can then recover the indices of self-overlap directly using a variant of our definition as follows:

```
def SOver(MSO,length):
    max=MSO[length]
    if max<=0 return []
    indices=list()
    indices.append(max)
    indices.extend(SOver(MSO[0:max],max))
```

Alternatively, we can convert the number to the minimal period and apply our definition directly. Moreover, our second definition will give an optimal lookup of the indices of self-overlap from the MSO table as follows without recursion:

```
def SOver(MSO,length):
    i=length
    indices=list()
    while i>0:
        max=MSO[i]
        per=i-max
        rem=i%per
        num=i/per /*floor division*/
        if num>2
             for j in range(2,num)
```

```
                indices.append(rem+per*j)
            i=per+rem
        else:
            indices.append(per+rem)
            i=rem
```

In sum, there are some interesting and important algorithmic implications that result from this relationship between self-overlap and periodicity. The set of indices of self-overlap of a string can be found in linear time - from the linearity of the KMP [4] preprocessing algorithm and that the additional processing will run in time less than N. Our proof of the second theorem on self-overlap indicates that the second definition of self-overlap makes optimal use of the table of maximal overlap/minimal periods given by the KMP to extract the indices of self-overlap.

It has not been shown that this algorithm itself is optimal for the task of finding self-overlap. Only that it is optimal given the table provided by a step in KMP preprocessing. It remains to be shown whether this relationship can amount to any optimization of the preprocessing step of KMP over the optimizations given there or whether it may be an explanation why optimizations there are optimal.

5 Other Applications and Acknowledgment

Self-overlap is surprisingly important to probability [3]. A given string's overlap properties help determine how long one must wait for such a string to occur in a process generating characters, whether that process is uniformly random or otherwise distributed. This has clear applications in some forms of financial instruments [5], and possible applications in checking if pseudo-random processes are sufficiently random. This work was inspired by [3], [2], [1], and later [4].

References

[1] Amir, A. and Benson, G. (1998). Two-Dimensional Periodicity in Rectangular Arrays, SIAM J. Computing 27, 1, 90-106.

[2] Apostolico, A. and Breslauer, D. (1997). Of Periods, Quasiperiods, Repetitions, and Covers. Structures in Logic and Computer Science Lecture Notes in Computer Science, 1997, Volume 1261/1997, 236-248.

[3] Blom, G. and Thorburn, D. (1982). How many random digits are required until given sequences are obtained? J. Applied Prob. 19, 518-531.

[4] Knuth, Donald, Morris, James H., jr, Pratt, Vaughan (1977). Fast pattern matching in strings. SIAM Journal on Computing 6 (2), 323–350.

[5] Schroeder, C.S. and Di Pierro, M. (2011) 'Pattern Derivatives', Int. J. Financial Markets and Derivatives, Vol. 2, No. 4, pp.249–257.