The Use of Pattern Matching Algorithms to Determine the Products of Organic Reaction

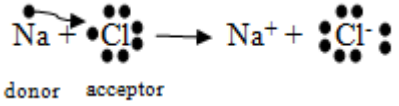
Vincentius Martin 13510017
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
vincentiusmartin@students.itb.ac.id

***Abstract*—Determining products from the reaction in organic chemistry is not a simple task. So many things about the compounds must be known, like : functional groups, reaction type, environmental conditions, etc. Due to this complexity, sometimes one must understand so many things about the reaction well and even memorize it. In this paper, the technique to analyze the reaction from its compounds is presented. The compounds' pattern will be treated as strings using the pattern matching algorithm, so the pattern that mark the particular reaction can be found. After the pattern are found, the reaction products can be determined using particular substring from the compounds and concate them in specified order. The compounds and characteristics from the specified reaction are used as the base to analyze the reaction and determine the products. Many conditions are used to test the technique so the accuracy and the efficiency of this technique can be determined.**

***Index Terms*—organic chemistry, pattern matching, chemical reaction, string.**

I. Introduction

Carbon has special characteristics that distinguish it from other elements in periodic table. Carbon create compounds with another elements using different way with what happened in the electron transfer at formation of inorganic compounds. Let's take example using Natrium (Na) in the formation of NaCl. It gives its outermost electron to Chlor (Cl) so the two of them can fullfill octet rule. There are electron donor (Na) and acceptor (Cl) in the formation. Carbon is different, with its four outer electrons, there is no need for it to give and take electron. Moreover, it because valence electrons from Carbon are really close to the center so the force is high and make it hard to be separated. This unique properties from Carbon makes the study of Carbon must be separated and it is called as organic chemistry.

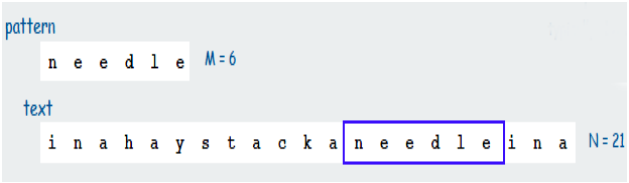


Picture 1.1 Electron transfer between Na and Cl.

Reaction in organic chemistry consist of compounds that are not classified as inorganic. Organic compounds can exist because of the unique parameters of carbon atoms that always have four bonds in organic compounds. At the same time, carbon atoms can bonding strongly to other non-metal atoms such as H,N, and O.

The reaction in organic chemistry consist of some kind of reactions. Organic compounds are involved in every organic reaction. Every reaction can be said modify the arrangement of elements in the compound to produce new compounds as production from the reaction. One type of compound if it reacted with other type of compound will produce different kind of product based of its reactants.

Compound analyzing can be done using pattern matching to identify the pattern in the reaction especially in one compound from the reaction. Pattern matching or sometimes called string matching is algorithm that is used to check wether a pattern is exist in the specified text or not. A text with n characters is tested to search for the existence of a pattern with m characters (m<n). There are some algorithms that are used for pattern matching, each of them has their own way to search for the specified pattern that make one algorithm is different with other algorithms.



Picture 1.2 Example of string matching.

Like any other algorithms, brute force can be used in pattern matching. With brute force, all position in the text will be checked. If the specified pattern match the beggining of the text then all other parts of the pattern will be matched with other parts from the text start from the beginning index that has been found before. The matching process continue until unmatched pattern (single character) is found. Brute force is good to be used when the alphabet of the text is large in diversity. In this paper, brute force is going to be used as comparison to other algorithm to determine the efficiency of the used algorithm. With effective algorithms, the result from the reaction can be found faster.

II. SOME THEORIES

2.1. Carbon Compounds Classification Based on Its Functional Groups

Every carbon compound has different characteristics that can be seen from its functional group. Every compound from the same group has the same functional group and similar chemical properties.

2.1.1. Haloalkane

Haloalkane (halogenoalkane or alkyl halide) is a group of chemical compounds which is a derivation from alkane that consist of one or more halogens such as Fluoride or Chlorin. There is a large distinction between structural and physical properties of haloalkane with alkane. Structural difference is caused because of replacement of one or more hydrogen with halogen atom. The difference in physical properties can be caused because of electronegativity, bond length, bond strength, and molecular size.

The physical properties of haloalkane is the structure of -X bond with X is halogen atoms that are bound with carbon chain. Different halogen atoms bound will give different polarity. Haloalkane can be formed from alkanes, alkenes, or alkynes with particular reaction.

2.1.2. Alcohol

Alcohol (alkanol) is a derivative compound from alkene with one or more of its hydrogen atoms are replaced with hidroxyl (-OH) functional group. General formula for alcohol is $C_nH_{2n+1}OH$. Based on the number of –OH groups that bounded, alcohol can be divided into two types : monovalent alcohol that bound only one –OH group and polyvalent alcohol that bound more than one –OH groups. Alcohol is polar but as the longer the size of its chain, the polarity is reduced.

2.1.3. Ether

Ether (alkoxy alkane) can be derivated by replace two hydrogen groups from water with alkyl groups. Ether can be made from alcohol or another compounds. The general formula for ether is $C_nH_{2n+2}O$. Compared to alcohol, ether has stronger bonds so it is harder to break.

2.1.4. Aldehyde

Aldehyde (alkanal) is carbonyl compound with –COH functional groups that can be called as formyl groups. It is named aldehyde because it can be made from alcohol dehydrogenation. General function for aldehyde is $C_nH_{2n}O$. Aldehyde is polar because of greater electronegativity from oxygen in carbonyl groups.

2.1.5. Ketone

Ketone (alkanon) is derivative from alkane that has carbonyl functional groups. The difference from aldehide is ketone has two alkyl functional groups that are bounded with carbon atom. The general function for ketone is equal to aldehyde : $C_nH_{2n}O$. Because ketone is carbonyl groups like aldehyde, ketone is also polar.

2.1.6. Carboxylic Acid

Carboxylic acid (alkanoic acid) is weak acid that is hard to unravel. Carboxylic acid can be obtained from the oxidation of aldehyde. The functional groups for carboxylic acid is –COOH with the general function : $C_nH_{2n}O_2$. In water, the solubility of carboxylic acid is decreasing with the increasing of C atoms amount.

2.1.7. Esther

Esther (alkyl alkanoate) is carbon compounds with –COOR functional groups. Esther is a derivation from carboxylic acid and can be produced by esterification of acid catalyst. The general function for esther is equal to carboxylic acid : $C_nH_{2n}O_2$. Ether is a good solvent for organic compounds that insoluble in water.

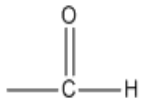
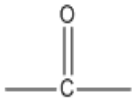
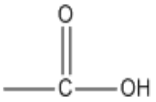
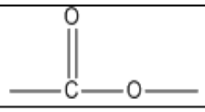
No	Name of compound	Functional Groups	General Function
1	Haloalkane	—X	$C_nH_{2n+1}X$
2	Alcohol	—OH	$C_nH_{2n+2}O$
3	Ether	—O—	$C_nH_{2n+2}O$
4	Aldehyde		$C_nH_{2n}O$
5	Ketone		$C_nH_{2n}O$
6	Carboxylic Acid		$C_nH_{2n}O_2$
7	Esther		$C_nH_{2n}O_2$

Table 2.1. Functional Groups and General Function of Carbon Compounds

2.2. Carbon Compounds Reaction

Carbon atoms can form many compounds compared to other atoms. It is because carbon can form single bond, double bond, triple bond, or make bonds with other carbons to make chain structure. Functional groups in carbon compounds have a role in determining the reaction from an organic compounds. Carbon compounds can react to other compounds with some mechanisms there are substitution, addition, elimination, reduction, and oxidation.

2.2.1. Substitution Reaction

Chemical reactions where an atom, ion, or group of atoms or ions in a compound is replaced by another atom, ion, or group. The example of this reaction is when haloalkane is reacted with base (hydroxide ion) to form methanol :

$$CH_3Cl + OH^- \rightarrow CH_3OH + Cl^-$$

In the reaction above, we can see that Cl atom is replaced by hydroxide ion. The Cl atom is substituted so now hydroxide ion is making a bond with carbon chain.

2.2.2. Addition Reaction

Addition reaction is the breaking of unsaturated bond to form saturated bond. Unsaturated bonds can be found in alkene, alkyne, aldehyde, keton, or other organic compounds that have two or triple bonds in it. In addition reaction, unsaturated bond is given atom or other functional group so the single bond is formed. This is the

example of addition reaction :

$$CH_3CH = CH_2 + HCl \rightarrow CH_3CHClCH_3.$$

In reaction above, the double bond is broken and ion H^+ and Cl^- from HCl is inserted. Sometimes addition reactions is called as saturatation reactions because the reaction makes carbon atoms saturated by inserting maximum number of attached groups.

2.2.3. Elimination Reaction

Opposite of addition reaction, elimination reaction is change of saturated bond to unsaturated bond. Saturated bonds can be found in organic compounds that have single bonds like alkene. Usually at least one hydrogen atom is separated from compounds to form double bonds. The example of elimination reaction is :

$$CH_3CH_2CH_2 - OH \xrightarrow{H2SO4} H_2O + CH_3CH = CH2$$

In reaction above, single bond on the last carbon atom is lost and OH join with H atom from second atom C to form H_2O.

2.2.4. Reduction and Oxidation

Reduction and oxidation rection (redox) is the reaction that consist of releasing (reduction) and accepting (oxidation) oxygen. Oxidation is about releasing electron and reduction is about gaining electron. There are oxidising agent which oxidises particular compounds and reducing agent which reduce particular compounds. The example of reduction reaction is alcohol formation from keton and the example of oxidation reaction is carboxylic acid formation from aldehide. Reduction reaction is like addition because it form single bonds from double bonds and oxidation reaction is like elimination because it form double bonds from single bonds.

2.3. String Matching

There are some algorithms that will be used in this paper to solve the problem. The used algorithms are :

2.3.1. Knuth-Morris-Pratt (KMP)

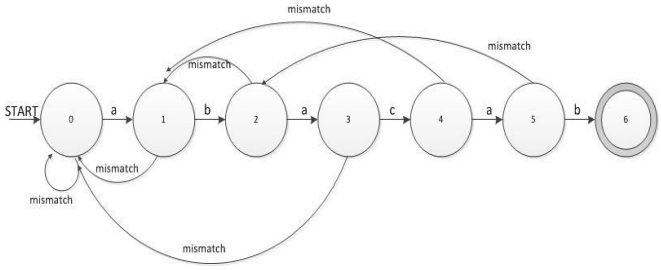
KMP algorithm check the pattern from left to right just like brute force but using smarter way in shifting the pattern. If mismatch is found in pattern P at P[j], the pattern is shift from the largest prefix of P[1..j-1] that is a suffix of P[1..j-1]. The example is like when the largest prefix that is a suffix from string "abaab" is about to be found. String "ab" is the answer, because the largest prefix can be found here : "abaab" and the suffix that is also largest prefix can be found here : "abaab". From the example, the string will be start checked from the third character if mistmatch occurs because "ab" length is two and that two characters is skipped, then the pattern is start checked from the third character. Else, if the character match then next character is checked.

KMP notes all the number of how far it must go from the specified character in the pattern if mismatch occurs. All of movement number is noted in a table that called Border Function. Let's take example of pattern "abacab", from the pattern can be made border function like this :

k	1	2	3	4	5	6
b(k)	0	0	1	0	1	2

Table 2.2. Border Function for Pattern "abacab"

Where k denotes the character number in specified pattern and b(k) denotes how far the pattern must be shifted if mismatch occurs in character of index k. The k=6 is not used because KMP algorithm only used until j-1 where j is string length.



Picture 2.1 DFA for KMP with String Pattern "abacab"

Algorithm for KMP is shown below, assuming that the border function has already made :

```
function kmpMatch (input text : array[0..n-1] of char,
                   input pattern : array[0..m-1] of
                   char) →integer
{function to compute the index of specified pattern in the
text using Knuth Morris Pratt algorithm, -1 if the pattern
is not found in the text}
DECLARATION
{assume that n is known as text length and m is known as
pattern length}
  index,i,j : integer
  border : array [0..m-1] of integer
ALGORITHM
  index ← -1
  i ← 0
  j ← 0
  border ← borderFunction(pattern)
  while (i<n) do
    if pattern[j]=text[i]
      if j=m-1 {has reached pattern end}
        index ← i-m+1
        break
      i ← i+1
      j ← j+1
    else if j>0
      j←border[j-1]
    else
      i ← i + 1
    endif
  endwhile
  →index
```

2.3.2 Boyer-Moore (BM)

Different with KMP algorithm, BM algorithm check the pattern from righ to left. BM uses two techniques, first is looking glass to search pattern in text by moving backwards through the pattern starting from the end of the pattern. Second is character jump to move the pattern when mismatch occurs. The movement is done based on

specific condition.

BM has Last Occurence function that map all letters in alphabet (usually use ASCII characters) to integer. The integer is index that sign the last occurence of specified character in pattern. Let's take example of pattern "abacab". Character 'b' has last occurence value 6 because the last 'b' occupies the 6th index. For character that is not exist in pattern will have -1 as index. Complete last occurence table for "abacab" is shown below :

x	a	b	c	d
L(x)	5	6	4	-1

Table 2.3. Last Occurence for Pattern "abacab"

Where x is the character that is mapped and L(x) is index for that character from last occurence function. Below is the BM algorithm, assuming that the last occurence function has already made :

```
function bmMatch (input text : array[0..n-1] of char, input
                  pattern : array[0..m-1] of char)
                  →integer
{function to compute the index of specified pattern in the
text using Boyer Moore algorithm, -1 if the pattern is not
found in the text}
DECLARATION
{assume that n is known as text length and m is known as
pattern length}
  i,j,lo : integer
  last : array[0..127] of integer
ALGORITHM
  last ←lastOccurence(pattern)
  i ← m-1
   if i > n-1then
    →i
   else
    j ← m-1
    repeat
      if pattern[j]=text[i] then
        if j=0 then
          →i
        else
          i←i-1
          j←j-1
        endif
      else
        lo←last[text[i]]
        i←i+m-min(j,1+lo)
        j←m-1
      endif
    until (i > n-1)
    →-1
  endif
```

2.3.3 Rabin-Karp (RK)

RK is an unique algorithm, because it uses hash value from string to check the similarity between elements in text and pattern. Hash value for every M subsequence from the text is calculated and then it compared with the hash value from the pattern. The similarity of hash values doesn't guarantee that the string match so if the hash values are equal then brute force comparison is used to check the similarity between pattern and M-character sequence. By doing this, there is only one comparison for each character subsequence and brute force only needs to be used once.

Hash function is needed to calculate the hash value from the specified substring. The hash value calculation is done by taking the specified string and mod it with value of q, where q is a prime number. There are two version of RK algorithm : Las Vegas (guaranteed to be correct) and Monte Carlo (guaranteed to be fast). The Las Vegas algorithm is the algorithm that is used in this paper. Below is the algorithm for Rabin-Karp, with hashFunction() and check() are already made.

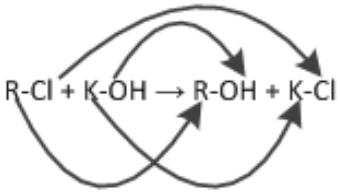
```
function rkMatch (input text : array[0..n-1] of char, input
                  pattern : array[0..m-1] of char)
                  →integer
{Rabin Karb - Las Vegas version. Like other string
matching algorithms that have been specified before,
return the index of specified pattern in the text}
DECLARATION
  {assume that n is known as text length and m is known
  as pattern length}
  RM,q,i,patHash,txtHash,offset : integer
ALGORITHM
    q ← randomPrime() {assume it is a function to
                       generate random prime number}
    RM ← 1
    for i←1 to m-1 do
      RM ← (256 * RM) mod q {256 is radix}
    endfor
    patHash ← hashFunction(pattern,m) {hash function
              that map pattern with hash for m
              sequences}
    {check is used for brute force comparison if the hash
    values are equal}
    if n<m then
      → -1 {text is shorter than pattern}
    else
      if patHash = txtHash and check(text,0)=true then
        → 0
      else
        for i←m to n-1 do
          txtHash ← (txtHash + q - RM*text[i-M] mod q)
                    mod q
          txtHash ← (txtHash*256 + text[i]) mod q
          offset ← i -m + 1;
          if patHash = txtHash and check(txt, offset)=true
          then
            → offset;
          endif
        endfor
        → -1 {no match}
      endif
    endif
```

III. IMPLEMENTATION

Some organic reactions have specific characteristics in their structures. Pattern matching is used to find some of the characteristics in compound structure that is used for the reaction. The reaction can be the manufacture of alkohol, manufacture of alkanon, alcohol decomposition, ester decomposition, etc. With the complete list of reaction between compounds, the accurate result can be more likely to be gotten. Example of compounds that are used to make alcohol is listed below. Where R is denoted the carbon chain in the compound. Of course there are many other reactions type that can be made but as an example, alcohol creation reactions is listed.

Reactant	**Product**
R-Cl + KOH	KCl + R-OH
R-MgX + H_2O_2 (Alkil Mg Halogenide)	Mg(OH)X + R-OH
R-COO-R + H_2O + HCl (Alkyl Alkanoic)	HCOOH + R-OH
R-NH_2 + HNO_2 (Alkyl Amine)	N_2 + H_2O + R-OH
R-I + H_2O (Alkyl Iodide)	HI + R-OH
R-X + AgOH (Alkil Halogenide)	AgX + R-OH

Table 3.1. Some Reactions to Make Alcohol (R-OH)



Picture 3.1 Substitution Reaction Between RCl and KOH

Let's take one example of a reaction that is used to make alcohol. :

RCl + KOH → ROH + KCl

Reaction above is a substitution reaction. We can see that Cl is substituted with OH to form the reaction products. With pattern matching we can determine whether the specified compounds are the compounds that is used in specific reaction. First, we use the function in the algorithm like this :

function makeReaction(input : array [0..] of String: reactants) →array [0..*] of String*

The procedure above take dynamic array of string as an input that denotes the reactants that are involved in the reaction. Then, substring that marks specified compound can be searched. In R-Cl, there is substring "Cl" that can be used to mark that the compound is R-Cl. But it must be ensured that Cl is located in the last place in the string. The pattern matching algorithms are used to determine this condition, if Cl is located in the index of string length - 2 ("Cl" lengths is two) then it can be known that the compound is in the form of R-Cl. After that, check the remaining reactant, if it is KOH then the reaction is the type that can make KCl and R-OH. Process that is used is same for other kind of reactions. Output from this function is dynamic array of string that denotes the products from the reaction. Pseudo code for the used algorithm approximately can be seen below :

```
function makeReaction(input array [0..*] of string : reactants) →array [0..*] of string
{function that is used to determine the products from the reaction, output is string "0" if the reaction is not on the list}
DECLARATION
  products : array [0..*] of string
ALGORITHM
 products ← 0
 foreach EVERY REACTION IN LIST do
    if SOLUTION(reactants) = true then
        products ← PRODUCT(reactants)
        break
    endif
 endfor
  → products
```

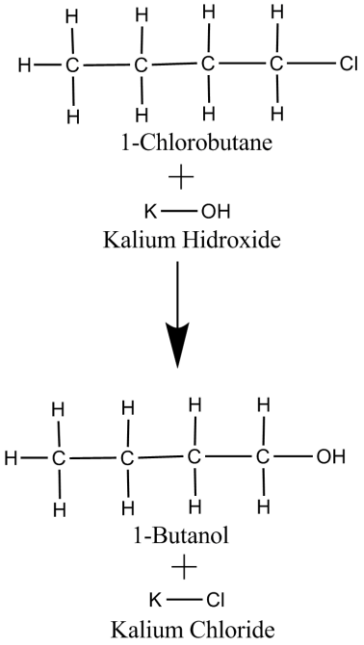
First, the algorithm iterates every reaction in the reaction list that has been specified before. SOLUTION(reactants) is the part that checks whether the reactants meet the conditions that are needed to form products. The SOLUTION part uses pattern matching algorithm to determine the compatibility between reactant strings and reaction in list that is being checked. If the conditions can be fullfilled then products array is filled with the products from the reaction that are made with certain string processing. PRODUCT(reactants) is process that is used to fill the products array with product results from reaction. To be able to determine the product results, the index of the pattern in the compound string must be known, so the string process can be done at the specified index. This index can be gotten with pattern matching algorithm that return the index of the pattern in the text. Example of process that is used is like this : the process can be used to move the specified substring from one compound to designated compound in substitution reaction. String that is moved start from the index that has been found with pattern matching algorithm. Of course if the pattern is not at the text, the index that is returned is -1. If this condition happen then no string process is made and the function continue to iterate the list to check if next registered reaction has match with the pattern. If no matched reaction found, then the function returns array with one size that contains 0 as a result.

IV. EXPERIMENTS

In order to validate the correctness of the methodology, some experiments were made to be run with some algorithms. The experimets are consist of some reactions with various compounds. Goal of these experiments are to determine the most effective pattern matching (string matching) algorithm to determine the products from organic reaction.

4.1. Substitution Reaction

Let's take a subtitution reaction as a text case. The reaction involves chlorobutane (C_4H_9Cl) with kalium hidroxide (KOH). C_4H_9Cl is a derivataion from butane with one Chlor atom. Products from the reaction will be determined using four algorithms, they are : brute force, KMP, BM, and RK. Patterns that are used as inputs in each algorithm are these two strings : "CH3CH2CH2CH2Cl" and "KOH".



Picture 4.1 Substitution Reaction C_4H_9Cl + KOH

The result can be known after running the program. To check the most optimal algorithm, the program is run several times. After several times, some different results are choosen. For this case, five results were choosen and the results can be seen below :

No	Brute	KMP	BM	RK
1	0 ms	0 ms	0 ms	1 ms
2	0 ms	0 ms	1 ms	2 ms
3	0 ms	0 ms	1 ms	0 ms
4	0 ms	0 ms	1 ms	1 ms
5	0 ms	0 ms	0 ms	0 ms

Table 4.1. Five Different Time Results from Program for reaction C_4H_9Cl + KOH

Brute	KMP	BM	RK
18 times	18 times	10 times	12 times

Table 4.2. Total Character Matcing for Each Algorithm

Four algorithms above can give correct result that is : CH3CH2CH2CH2OH + KCl. So, for the result there is nothing to be discussed about the accuracy of each algorithm because all of them give the correct result. The thing left is the efficiency of each algorithm, it can be seen that BM algorithm has least character matching process. This result is gotten from searching "Cl" in "CH3CH2CH2CH2Cl". Reason why BM algorithm can has lesser number of character matching compared to other algorithms is because BM starts matching from the last character in pattern string. Last character in pattern that is "l" match nothing until the last character in text is compared with "l". BM algorithm skips many useless process by moving the pattern to index of last pattern character in text + 1. From the process time of each algorithm, RK has the longest time, it is because of the hash function that needs some time to compute the hash value. While using brute force and KMP can give faster process, it is because the processes in brute force and KMP algorithm are more simple than processes in BM and RK.

Second test case using substitution reaction is the reaction that involves Grignard reagent. Grignard reagent has the general formula : RMgX. Where X is a halogen and R is carbon chain. This is the reaction formula to create aldehide using Grignard reagent :

$$HCOOR' + RMgX \rightarrow Mg(OR')X + R\text{-}COH$$

Patterns that are used for this test are : "HCOOCH2CH2CH3" and "CH3CH2MgCl", with four algorithms like before.

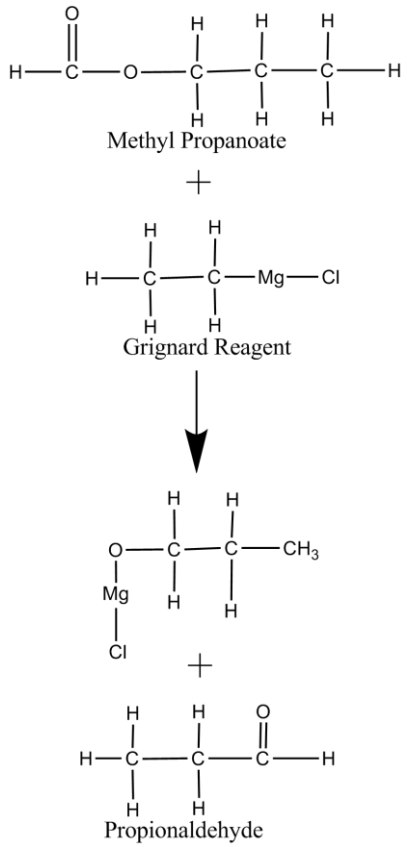
Again, to check the most optimal algorithm, each algorithm is compared. After running program several times, these are five different results that are choosen :

No	Brute	KMP	BM	RK
1	0 ms	0 ms	0 ms	1 ms
2	0 ms	0 ms	1 ms	1 ms
3	0 ms	0 ms	0 ms	0 ms
4	1 ms	0 ms	0 ms	1 ms
5	0 ms	1 ms	0 ms	2 ms

Table 4.3. Five Different Time Results from Program for reaction HCOOCH2CH2CH3 and CH3CH2MgCl

Brute	KMP	BM	RK
8 times	8 times	5 times	6 times

Table 4.4. Total Character Matching when Searching Mg in CH3CH2MgCl



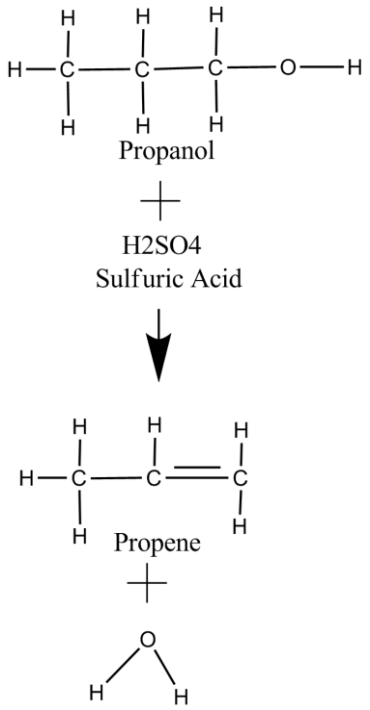
Picture 4.2 Substitution Reaction "HCOOCH2CH2CH3" and " CH3CH2MgCl"

The result from all algorithms that are used is Mg(OCH2CH2CH3)Cl + CH3CH2COH which means the result is right. The results are really similar with the reaction before. Same as before, BM algorithm gives the least character matching process number. So now, it can be more assured that BM do character matching processes lesser than any other pattern matching algorithms that are used in this paper when determining the product of organic reaction. The used time for BM sometimes slower than other algorithms but the difference is not large.

4.2. Elimination Reaction

Now, let's try another type of reaction, the elimination reaction. In alcohol compound, the elimination reaction produce H_2O and it can be produced with concentrated H_2SO_4 at 180°C. Of course not only alcohol that can do elimination reaction, but in this discussion the alcohol compound is going to be used.

The elimination reaction that is going to be discussed takes propanol (C3H8O) that reacted with concentrated H_2SO_4. Product from the reaction should be an unsaturated bond, in this reaction is propene. Again, four algorithms are going to be used. The input string for the reaction is "$CH_3CH_2CH_2OH$" and "H_2SO_4".



Picture 4.3 Elimination Reaction from Propanol

The results from running program several times are shown below with two different results are choosen.

No	Brute	KMP	BM	RK
1	0 ms	0 ms	0 ms	1 ms
2	0 ms	0 ms	1 ms	1 ms

Table 4.5. Two Different Time Results from Input "CH3CH2CH2OH" and "H2SO4"

Brute	KMP	BM	RK
11 times	11 times	11 times	9 times

Table 4.6. Total Character Matching when Searching "OH" in "CH3CH2CH2OH"

Product of the reaction that can be gotten are "CH3CHCH2" and "H2O". Only two different time results that can be gotten from the experiment. Program compilation is very stable for this reaction. Most of the result is the first result in table 4.5. It can be seen from table 4.6 that the least charcter matching process is in the RK algorithm, it is because the other algorithms must be compred with all character in the text. It is because the first 'O' in "OH" is no match with anything until the 'O' character at the end (before 'H') in the text is found. In this case, RK function that using hash function can do lesser comparison, but still the time is longer because it has to compute the hash value from the string. Elimination reaction still use the same methods with substitution reaction, it also apply to substitution and redox reaction.

V. Algorithm Analysis

For the four algorithms that are used in this paper, there are some analysis to determine the effective algorithms based on their complexity, algorithms method, etc. Brute force algorithm runs in time O(mn) for the worst case, it means that all characters in pattern must be compared to every single character in the text. But, most search for normal text usually runs in time O(m+n) which is very quick. Brute force algorithm is effective to be used to analyze organic reaction because most organic compounds usually don't have specific substring that almost match with the pattern. Brute force algorithm is slow if there are many substrings in text that really similar but different with the pattern. Other things is, the alphabet size usually large in organic reaction so it is good for the brute force algorithm.

KMP algorithm usually runs in time O(m+n) which is very fast. Although KMP algorithm doesn't need to run backward, but unlike brute force, KMP is not good to be used when the size of the alphabet increases. More chances of KMP algortihm to get mismatches when the size of alphabet in the text is large. For organic reaction analysis, KMP reaction is not so good to be used because the characters in the text may vary.

BM algorithm runs in worst time of O(mn+A). When the size of alphabet A is large, BM algorithm can perform the search faster. If the pattern is not found in the text, the complexity for BM algorithm is O(m+n). But for the best match, the complexity is O(n/m). BM algorithm is good to be used in organic reaction analysis because the characters in compound may vary so the alphabet size is large. Only when preparing the last occurence table that takes some time in BM algoritm.

RK algorithm is like brute force in the complexity, it has O(mn) for the worst case and O(m+n) expected running time. What makes the algorithm is heavy it is because the computation of hash value from the string. To compute the hash value, hash table must be filled and this takes some time. But still, RK algorithm is effective to be used for string matching because matching hash value from the string is easier. Usually, RK takes longer time than the other algorithm, it is because of the computation of hash value. When used to analyze the compounds that are used in the reaction, RK algorithm takes longer time but gives small number of matching for some reaction.

VI. Conclusion

Pattern matching algorithm is really suitable to determine the products from organic reaction. By using pattern matching algorithm, the reactants can be analyzed to determine whether they have some specific substrings that denote a structure in the reactant strings. After the structure is found, the products can be determined. But, every reaction must be specified in the database that is contained, if no matched reaction in the database, of course pattern matching algorithm can't be used to determine the products.

From the experiments, maybe good pattern matching algorithm that can be used to analyze reactant strings is Boyer Moore. It is because usually BM algorithm do smallest matching processes between pattern and the text. Also, BM algorithm needs small time to analyze the input text in organic reaction case. Other algorithms such as brute force, KMP, and RK also can be used and they are still good. But some of the algorithms do complex process that makes the pattern matching process becomes more severe.

References

[1] Brady, James E; Frederick A.Senese, & Neil D.Jespersen.2009. *CHEMISTRY 5th ed*. Asia : John Wiley & Sons.

[2] Britannica. *Addition Reaction*. December 21, 2012 (2.00 PM) < http://www.britannica.com/EBchecked/topic/5488/addition-reaction>

[3] Britannica. *Substitution Reaction*. December 21, 2012 (2.00 PM) < http://www.britannica.com/EBchecked/topic/571075/substitution-reaction>

[4] Cahyana, Ucu; Dede Sukandar; Rahmat. 2007. *KIMIA 3rd ed*. Jakarta : Piranti Darma Kalokatama.

[5] Chem-is-try. *Pengantar Halogenalkana (haloalkana atau alkil halida)*. December 21, 2012 (1.00 PM) < http://www.chem-is-try.org/materi_kimia/sifat_senyawa_organik/halogenalkana__haloalkana_atau_alkil_halida_/pengantar_halogenalkana_haloalkana_atau_alkil_halida/>

[6] Chem-is-try.*Alkanon*. December 21, 2012 (1.00 PM) < http://www.chem-is-try.org/materi_kimia/kimia-kesehatan/senyawa-hidrokarbon/alkanon/>

[7] CHEMWiki. *Haloalkanes*. December 21, 2012 (1.30 PM) < http://chemwiki.ucdavis.edu/Organic_Chemistry/Hydrocarbons/Haloalkanes>

[8] Munir,Rinaldi. 2009. *Diktat Kuliah IF3051 Strategi Algoritma*. Program Studi Teknik Informatika STEI ITB

[9] Fieser, Louis F; Mary Fieser. 1963. *Pengantar Kimia Organik*. Bandung : DHIWANTARA

[10] IGM. *Boyer-Moore Algorithm*. December 21, 2012 (2.50 PM) < http://www-igm.univ-mlv.fr/~lecroq/string/node14.html>

[11] IGM. *Karp-Rabin Algorithm*. December 21, 2012 (2.50 PM) < http://www-igm.univ-mlv.fr/~lecroq/string/node5.html>

[12] Princeton. *RabinKarp.java*. December 21, 2012 (2.47 PM) < http://algs4.cs.princeton.edu/53substring/RabinKarp.java.html>

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 21 Desember 2012

Vincentius Martin 13510017