# Chapter 3 Development of customized Biotool for gene prediction

## 3.1 Introduction

Computational-biology research is a vast area of science having complex queries and essential requirements for reliability, availability, and scalability of sequence analysis. Gene tool deals with DNA sequence assembly; gene finding and analysis, Protein expression and regulation. It predicts examine some of the application of computational biology. It allows researchers to precisely adopt genes and gene products to suite their specific requirement. It also includes various operations as Reverse sequence, Double Stranded sequence, Complement Of sequence, Features Information that helps the researcher to find out the function of a particular gene sequence.

## 3.2 Scope

Bioinformatics includes research in the application of computer science in biology, including the implementation of algorithms, strategies of mathematics, graph theory in life science [**1**]. Gene tool deals with DNA sequence assembly; gene finding and analysis, Protein expression and regulation. It predicts examine some of the application of computational biology. It allows researchers to precisely adopt genes and gene products to suite their specific requirement. It also includes various operations as Reverse sequence, Double Stranded sequence, Complement of sequence, Features Information that helps the researcher to find out the function of a particular gene sequence [**2, 3**]. One important branch of bioinformatics concerns the creation and maintenance of databases of biological information whereby researchers can both access existing information and submit new entries. The most pressing tasks in bioinformatics involve the sequence Analysis information.

### 3.3 Analysis

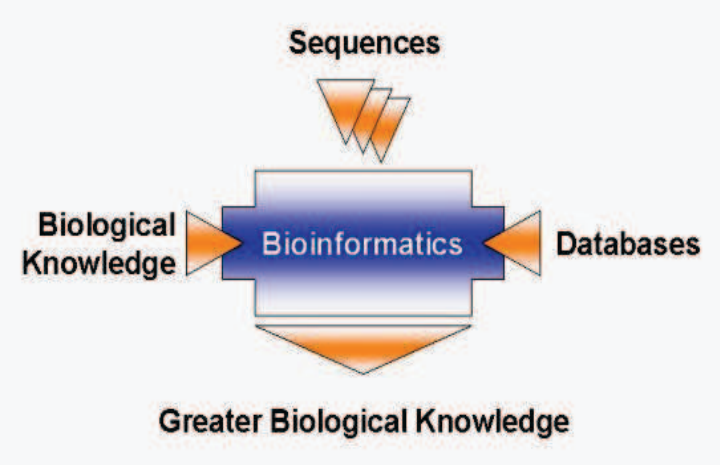


Fig. 3.1- Bioinformatics Knowledge

Computational Biology is the name given to this process, and it involves the following function:

1. Finding the genes in the DNA sequences of various organisms
2. Developing methods to predict the structure and/or function of newly discovered proteins and structural RNA sequences.
3. Clustering protein sequences into families of related sequences and the development of protein model.

### 3.4 Bioinformatics Databases

Bioinformatics databases are electronic reservoirs of biological data.

**GenBank:** GenBank (Genetic Sequence Data Bank) genetic sequence database, an annotated collection of all publicly available DNA sequences. It is a rapidly growing international repository of known genetic sequences from a variety of organisms. Its use is central to modern biology and to bioinformatics. It is a part of the International Nucleotide Sequence Database Collaboration, which includes:

a. DNA Databank of Japan (DDBJ),

b. European Molecular Biology Laboratory (EMBL),
c. GenBank at the National Center for Biological Information (NCBI).

There are more than 13 millions sequences in the GenBank. We can compare our putative gene with the millions of other sequences in the GenBank Databases at the National Center for Biological Information (NCBI) and search results give the functional role of our newfound gene. Gene Sequence is stored in the GenBank in FASTA format [**4-6**].

### 3.5 The Need of Gene Tool

It encompasses fields such as comparative genomics, structural genomics, transcriptiomics, proteomics, cellunomics and metabolic pathway engineering. Developments in these fields have direct implications to healthcare, medicine, discovery of next generation drugs, development of agricultural products, renewable energy, environmental protection etc. Bioinformatics plays a key role in the cutting edge research & development areas such as protein engineering, pharmacogenomics, discovery of new drugs and vaccines, molecular diagnostic kits, agro-biotechnology etc. This has attracted attention of several companies and entrepreneurs. Sequence analysis is one of the most peer research fields today. Scientists perform high quality research in this field [**7-9**]; Comparison of sequences is done in order to find similar sequences [**10-13**]; gene-structures, reading frames, and for predicting regulatory elements prediction of protein structures is done. We present a simple tool called Gene Tool that can perform all the predefined tasks easily, accurately and quickly [**14-19**].

Information Technology, particularly the Internet, is utilized to collect, distribute and access ever-increasing data which are later analyzed with mathematics and statistics-based tools so, the Gene Tool needs for following applications-

a. Whole genome analysis and sequences,
b. Experimental Analyses involving thousands of Genes simultaneously,
c. Medical applications: Genetic Disease,
d. Pharmaceutical and Biotech Industry,
e. Forensic application,
f. Agricultural applications.

### 3.6 Overview

This section is divided into several sub-sections. The first sub-section, gives an outline the various components of this tool. The next sub-section deals with its hardware and software requirements specifications, usage criteria and deployment procedures. Algorithm design and functional-level analysis is put forward in a separate sub-section. The last sub-section deals with implementation criteria, testing and snapshots.

#### 3.6.1 Components of Gene Tool

Gene tool consist of menus as File, Edit, View, Analyze on the given sequence. The sequences used in the analysis are available in the example folder so we can perform each operation. Files that are generated by each sequence analysis step are also available in this folder. It also provides Help menu provides to navigate to information relating to specific applications and commands.

Following menus are provided for Gene Tool.

##### 3.6.1.1 File

It has many options, that are summarized as,

a. Open –Load the sequence file.
b. Save- Save the contents of the sequence editor and associated sequence analysis results to sequence file.
c. Save as- Save contains under new file name.
d. Print-Print contains of the sequence editor.
e. Page set up-Adjust the page orientation and margins for printing.
f. Close-Exit from the sequence editor.
g. Quit-Quit from Gene tool.

### 3.6.1.2 Edit

The options present in this are,

a. Select all-Select the entire DNA sequence.
b. Cut-Cut the selected DNA sequence from the sequence editor.
c. Copy-Copy the selected DNA sequence from the sequence editor.
d. Paste-Paste the contents of foreign sequence file into sequence editor.
e. Double stranded-Entire DNA sequence converted into Double stranded form.
f. E.g. A T C G Single Strand

   T A G C Double Strand

g. Reverse-Reverse the order of bases in the selected DNA sequence.
h. E.g. G G A A T T T T Original Sequence

   T T T T A A G G Reverse of original

*Note:* Result is not equivalent representation of the same double Strand DNA molecule.

i. Reverse Complement- Reverse and complement the selected DNA sequence.
j. In complement A is replace by T and G is replace by C and vice –versa.
k. For e.g. A T G C Original DNA sequence

   C G T A Reverse of original sequence

   G C A T Complement

*Note:* Result is equivalent representation of the same double strand DNA molecule.

### 3.6.1.3 View

The options present in this are,

a. Layout Editor-It contains option as double stranded, Uppercase, Bold, and Underline, and Italic.
b. Grouping -1, 2, 3…10 allow you to choose group size the DNA sequence.

c. Translation-Allows you to add the protein translation of the DNA sequence to your layout.
d. Feature -It shows the feature information that contains feature name, key, and location.
e. Set Group Size- set the group size.

#### 3.6.1.4 Analyze

The options present in this are,

a. Translate- Used to generate DNA sequence into Protein sequence.
b. Compute statistics - Display the statistical information of selected DNA sequence.
c. Find ORF- Used for finding open reading frames in sequence.

#### 3.6.1.5 Help

One can access Gene Tool Help's viewer at any time by clicking on Help button.

### 3.6.2 Specifications

This section deals with various hardware, software specifications, usage criteria and deployment strategies.

#### 3.6.2.1 Hardware and Software Specification

This tool is a windows-based application. It requires at least a Pentium IV processor having a minimum of 256 MB and 1GB of free space. Visual Basic .NET should be installed in the machine running on Windows 2000, XP or higher versions of operating systems.

#### 3.6.2.2 Usage Criteria

This can be used by a person having very basic knowledge of molecular biology, sequence analysis and more specifically bioinformatics. Users having preliminary knowledge of computers may also find it very useful to handle.

### 3.6.3 Deployment Strategies

Visual Basic .NET delivers new productivity features for building more robust applications easily and quickly. With an improved integrated development environment and a significantly reduced startup time, it offers fast, automatic formatting of code as one type in. With this, one can build applications more rapidly and deploy and maintain them with greater efficiency. One can also create reusable, enterprise-class code using full object-oriented constructs. Language features include full implementation inheritance, encapsulation, and polymorphism. Structured exception handling provides a global error handler and eliminates spaghetti code [**20-26**].

## 3.7 Requirement Analysis

### 3.7.1 Hardware and Software Requirements

**Hardware Requirements:**

It requires at least a Pentium IV processor having a minimum of 256 MB and 1GB of free space.

***Software Requirements:***

Windows Application required Visual Basic.net

Operating System: Windows 2000, XP.

### 3.7.2 Platform Requirements

Microsoft Visual Basic.net used as it has following features:

#### 3.7.2.1 Powerful Windows-based Applications in Less Time

Visual Basic .NET 2003 delivers new productivity features for building more robust applications easily and quickly. With an improved integrated development environment and a significantly reduced startup time, Visual Basic .NET 2003 offers fast, automatic formatting of code as you type

#### 3.7.2.2 Simplified Deployment

With Visual Basic .NET 2003, you can build applications more rapidly and deploy and maintain them with greater efficiency.

#### 3.7.2.3 Full object-oriented construct

Create reusable, enterprise-class code using full object-oriented constructs. Language features include full implementation inheritance, encapsulation, and polymorphism. Structured exception handling provides a global error handler and eliminates spaghetti code.

#### 3.7.2.4 Improved Coding

Code faster and more effectively. A background compiler for real-time notification of syntax errors transforms you into a rapid application development-coding machine

### 3.7 System Design

This section gives detailed information about the algorithm followed by this tool as well as its pseudocode. This section includes Data Flow diagrams, UML diagrams, functional analysis and flow-chart.

### 3.7.1 Data Flow Diagrams

This sub-section represents the flow of information in the Gene Tool.

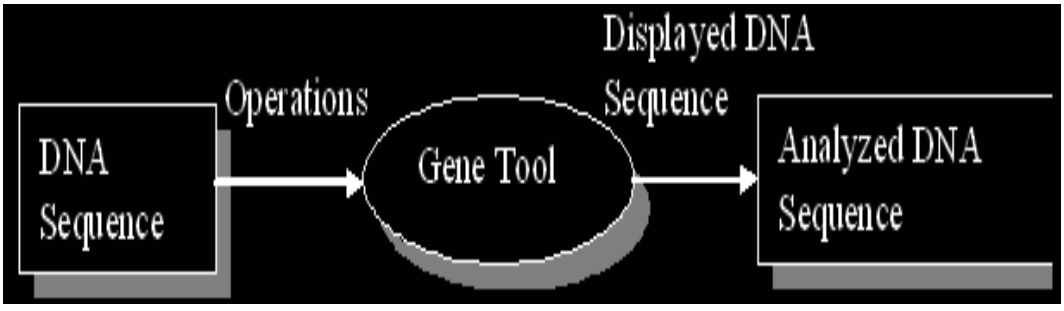


**Fig. 3.2- Level 0 of development**

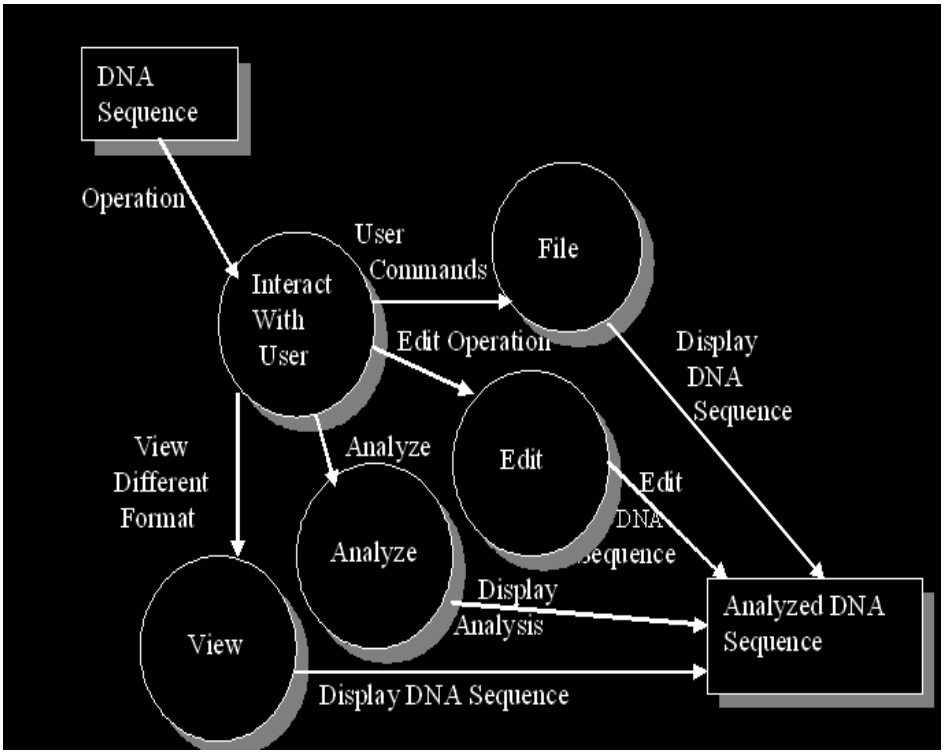


**Fig. 3.3- Level 1 of development**

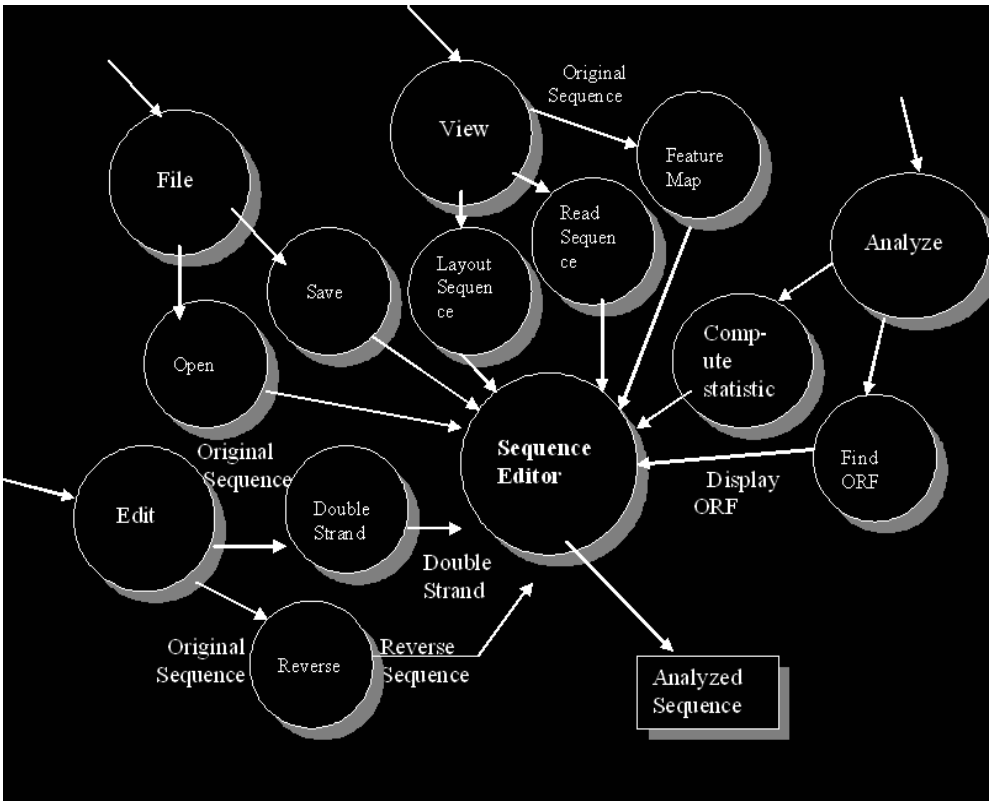


**Fig. 3.4- Level 2 of development**

### 3.7.2 UML Diagrams

This sub-section represents the sub-categorization of the various options of the tool that have been already discussed above.

#### 3.7.2.1 Use case diagrams

1. **Menu**

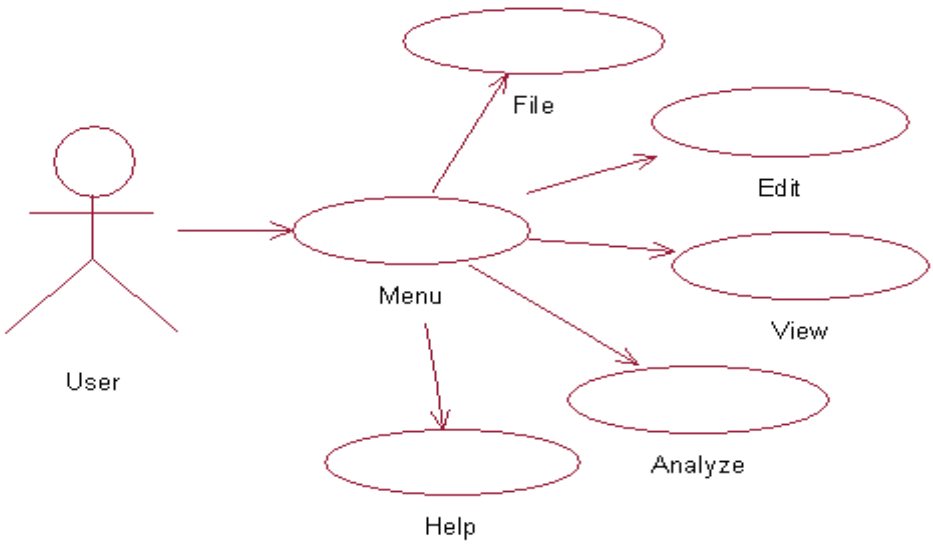


Fig. 3.5-- Menus uml diagram of gene tool

**2. File**

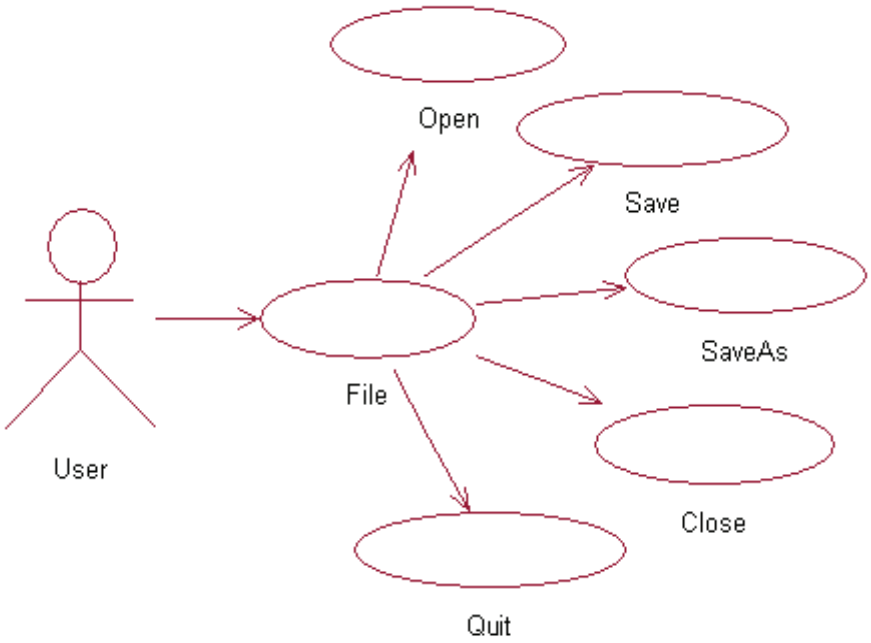


Fig. 3.5 – File uml diagram of gene tool

**3. Edit**

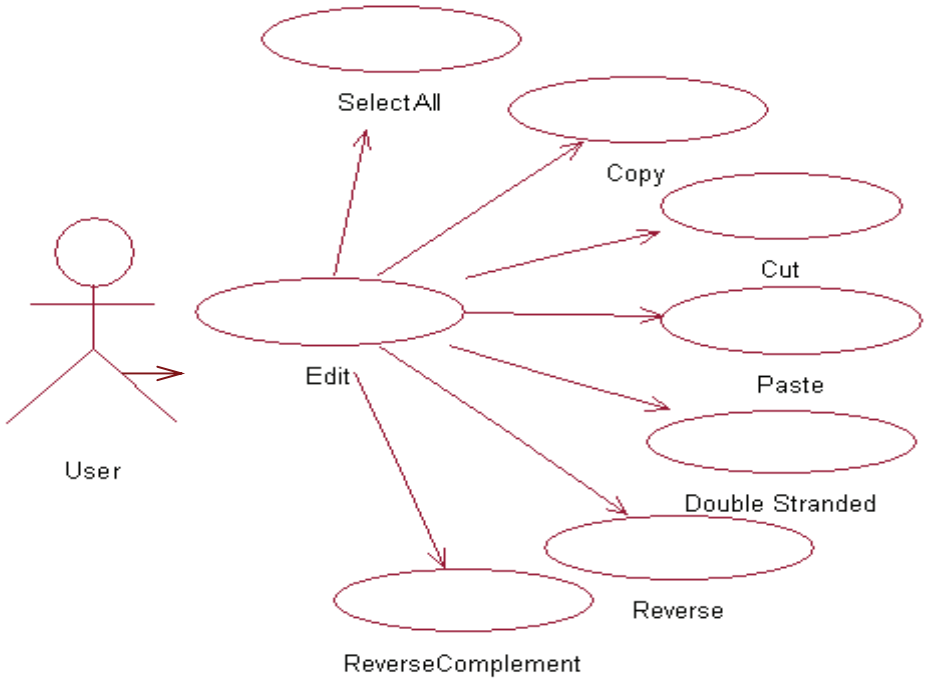


Fig. 3.6 - Edit uml diagram of gene tool

**4.View**

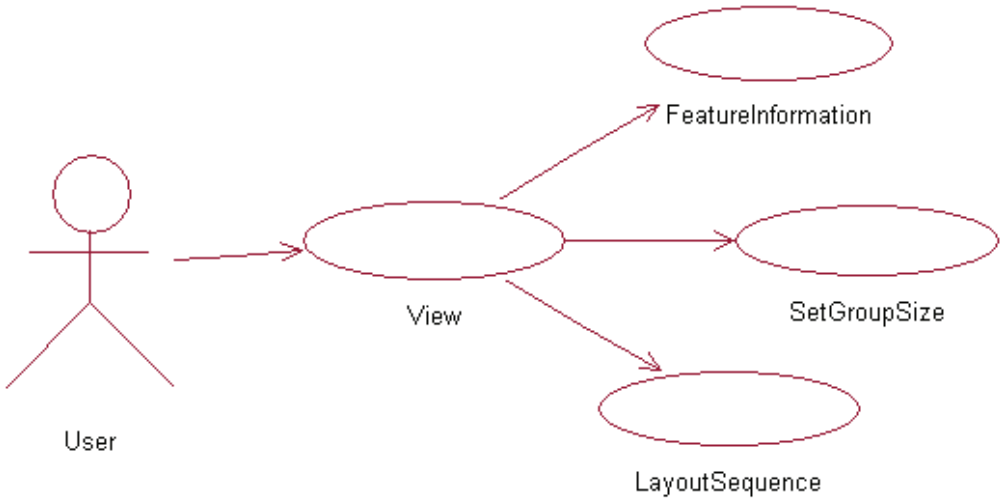


Fig. 3.7 - View uml diagram of gene tool

**5. Analyze**

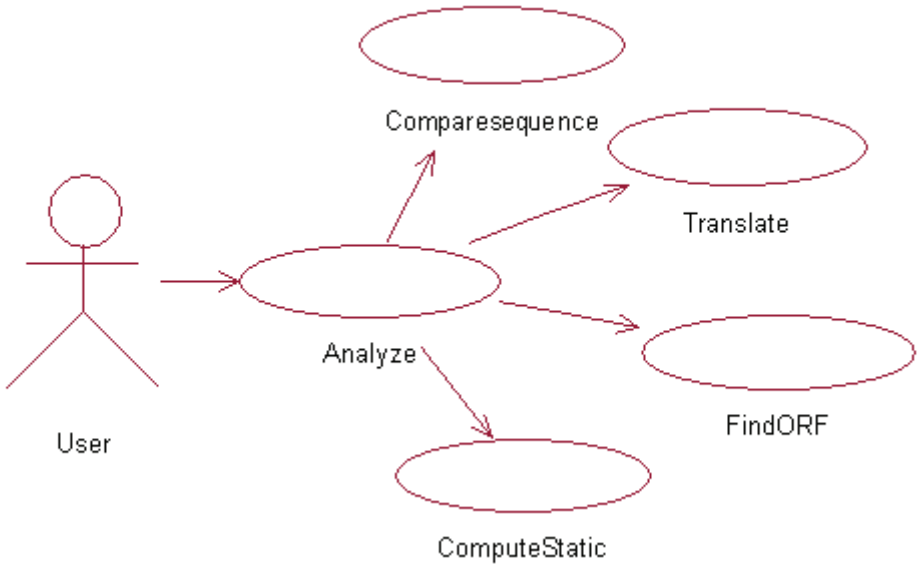


Fig. 3.8- Analyze uml diagram of gene tool

### 3.7.3 Functional Analysis

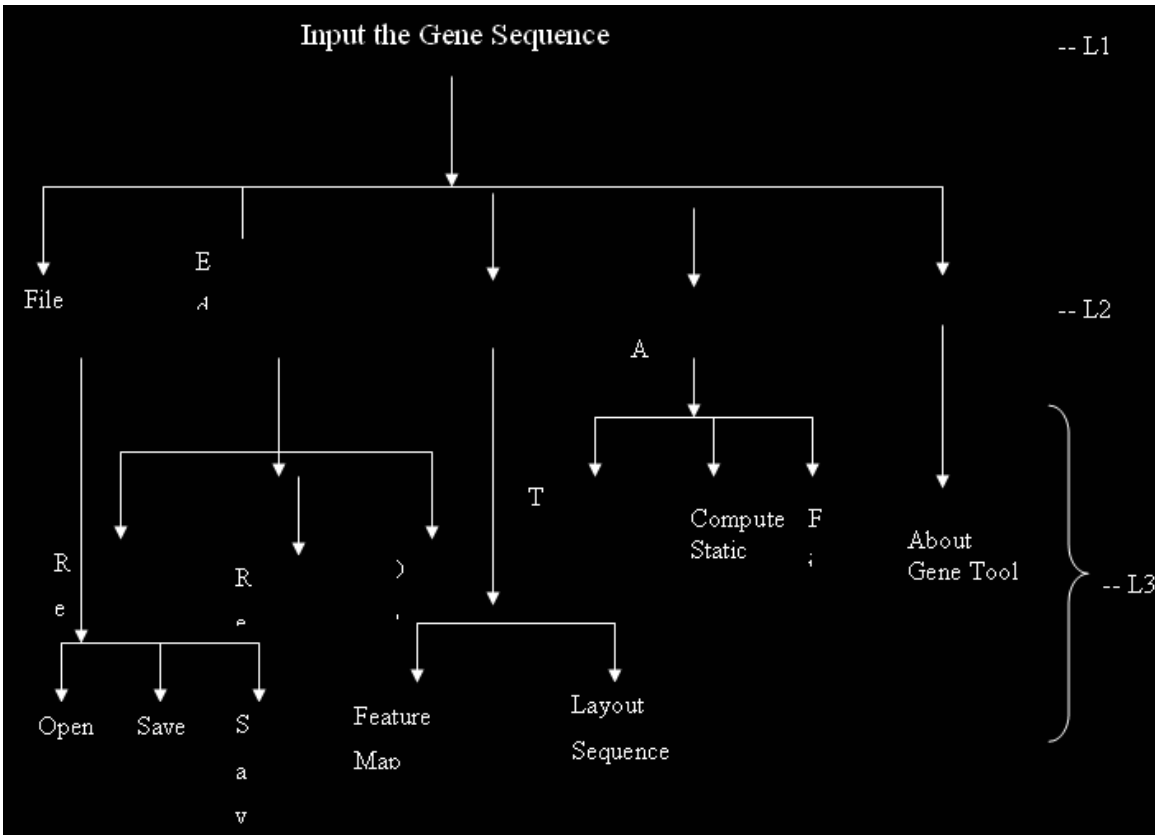


Fig. 3.9- Level diagram

**L1 (Level 1):**

**Input the Gene Sequence**

**Input:** A, T, G, C Gene Sequence (Stored in File in Particular format).

**Output:** Analyzed Sequence

**Description:** This process analyzes the sequence taken from the input file by implementing the level 2 and level 3 functions.

**L2 (Level 2):**

**1. File:**

**Input:** A, T, G, C Gene Sequence (Stored in File in Particular format).

**Output:** Shows the sequence in Sequence Editor with scaling.

**Description:** This function shows the A, T, G, C gene Sequence in Sequence Editor Using scale.

**2. Edit:**

**Input:** A, T, G, and C Gene Sequence in Sequence Editor.

**Output:** shows the edited sequence in Sequence Editor.

**Description:** This function shows the A, T, G, C gene Sequence in Sequence Editor With different formats as Reverse sequence, Double Stranded Sequence.

**3. View:**

**Input:** A, T, G, and C Gene Sequence in Sequence Editor

**Output:** Views the Sequence in Layout Sequence Editor to do Different Operations on it.

**Description:** This function shows the gene Sequence in Layout Sequence Editor For doing different operations on it as Grouping, Uppercase, Bold etc.

**4. Analyze:**

**Input:** A, T, G, and C Gene Sequence in Sequence Editor

**Output:** Translate the sequence in other type of sequence by analyzing Gene Sequence.

**Description:** This function analyzes the Gene Sequence and translates it into protein Sequence also it shows the compute static of the sequence.

**5. Help:**

**Description:** This function shows the Help for Gene Tool .It Guides the user how to use the Gene Tool.

**L3 (Level 3):**

**1. Open:**

**Input:** Open the File where the Gene sequence is stored.

**Output:** Shows the Gene Sequence in Editor with formatted and scale.

**Description:** This function opens the input Gene Sequence in Sequence Editor.

**2. Save and Save As:**

**Input:** The gene Sequence in the Sequence Editor.

**Output:** Saves the analyzed sequence in required location.

**Description:** This function saves the Sequence at the required location in file.

**3. Reverse:**

**Input:** Sequence in Sequence Editor.

**Output:** Shows the Reverse of input Sequence.

**Description:** This function shows the Reverse Sequence.

E.g. Input: ATTGG ATGCC

Output: CCGTA GGTTA

**4. Reverse Complement:**

**Input:** Sequence in Sequence Editor.

**Output:** Shows the Reverse Complement of input Sequence.

**Description:** This function shows the Reverse Complement Sequence.

E.g. Input: ATTGG ATGCC

Output: GGCAT CCAAT

**5. Double Stranded:**

**Input:** Sequence in Sequence Editor.

**Output:** Shows the Double Stranded of input Sequence.

**Description:** This function shows the Double Stranded of Sequence.

E.g. Input: ATTGG ATGCC

Output: TAACC TACGG

**6. Layout Sequence:**

**Input:** Sequence in Sequence Editor.

**Output:** Shows the input Sequence in different format.

**Description:** This function shows the Sequence in different format for convince of user.

**7. Translate:**

**Input:** Sequence in Sequence Editor.

**Output:** Shows the Translation of input Sequence in protein Sequence.

**Description:** This function shows the Protein sequence of input Sequence.

E.g. Input: TTCTCATGTT TGACAGCTTA

Output: FSCLTA

**Level 1**

In level 1, after inputting the gene sequence the process analyzes the sequence taken from the input file by implementing the level 2 and level 3 functions, viz., File, Edit etc.

**Level 2**

In level 2, the *File* function shows the A, T, G, C gene Sequence in Sequence Editor using scale. *Edit* function shows the A, T, G, C gene Sequence in Sequence Editor with different formats as Reverse sequence, Double Stranded Sequence. *View* function shows the gene Sequence in Layout Sequence Editor. For doing different operations on it as Grouping, Uppercase, Bold etc. *Analyze* function analyzes the Gene Sequence and translates it into protein sequence and also shows the compute static of the sequence. *Help* function shows the Help for Gene Tool. It Guides the user how to use the Gene Tool.

**Level 3**

In level 3, *Reverse* function shows the reverse sequence. *Reverse Complement* function shows the reverse complement of the sequence. *Double Stranded* function shows the double stranded of sequence. *Translate* function shows the protein sequence of input sequence.

### 3.7.4 Flow Chart

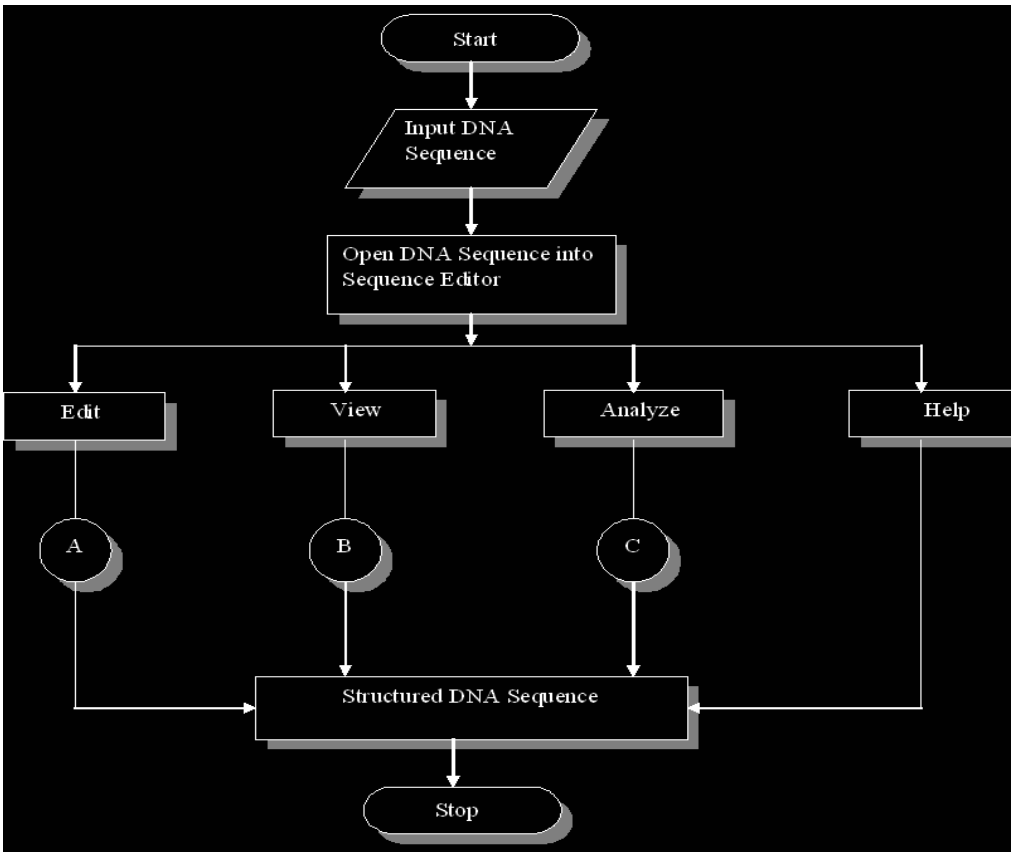


**Fig. 3.10- Flow chart of Gene Tool**

The above flow chart represents the basic procedure that Gene Tool follows. In the above case there are 4 operations that it does, including edit, view, analyze and help. A, B, C and D represent the further sub-operations performed by edit, view, analyze and help.

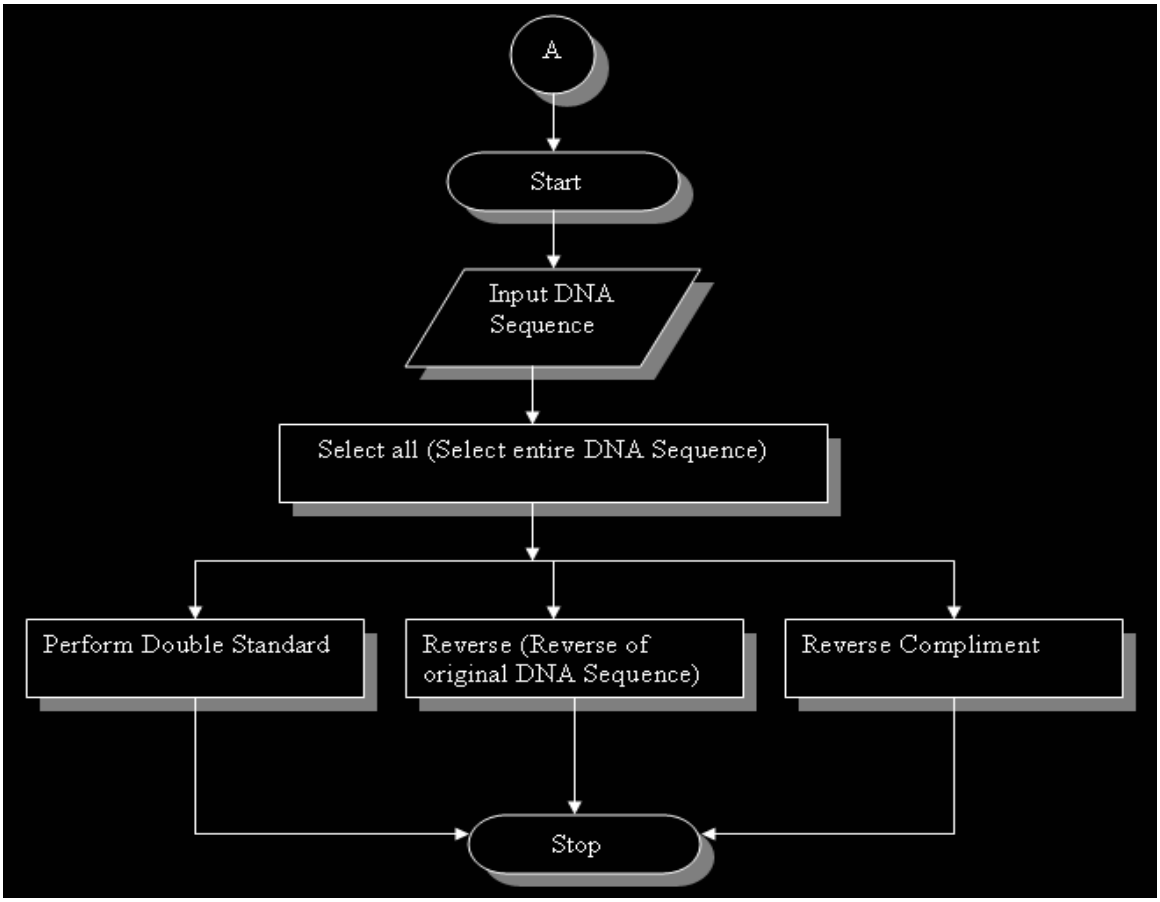

Fig. 3.11 - Flowchart for Edit Menu

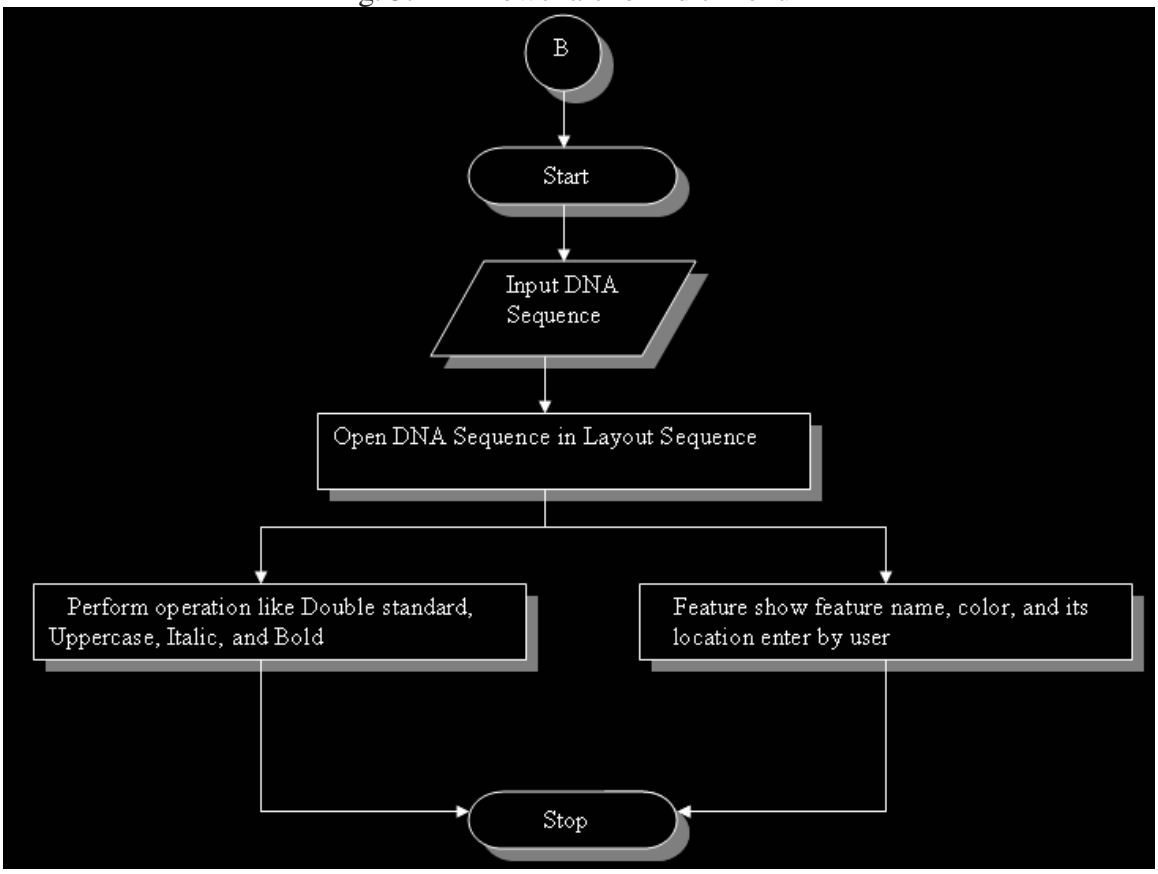

Fig. 3.12 - Flowchart for View Menu

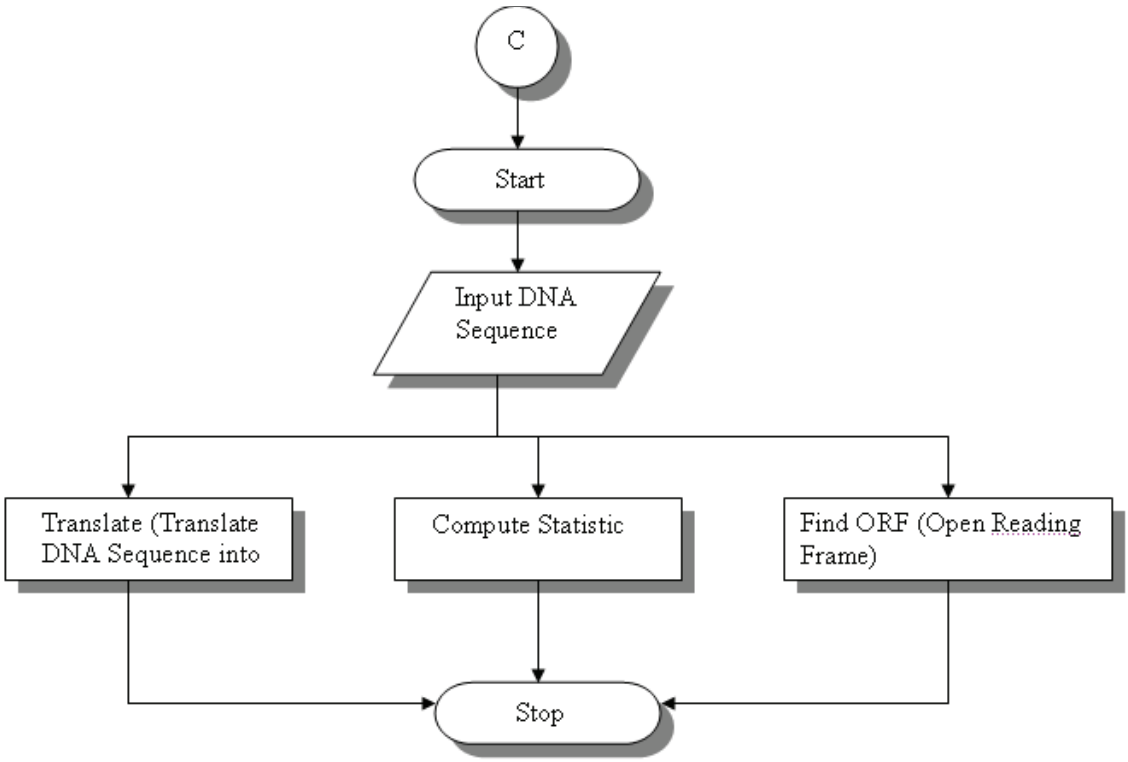

Fig. 3.13 - Flowchart for Analyze Menu

### 3.8 Algorithm Design and Pseudocode

Three approaches to gene finding
One can distinguish between three types of approaches:

A. Statistical or *ab initio* methods. These methods attempt to predict genes based on statistical properties of the given DNA sequence. Programs are e.g. Genscan, GeneID, GENIE and FGENEH.
B. Homology methods. The given DNA sequence is compared with known protein structures. Programs are e.g. TBLASTN or TBLASTX, Procrustes and GeneWise.
C. Comparative methods. The given DNA string is compared with a similar DNA string from a different species at the appropriate evolutionary distance and genes are predicted in both sequences based on the assumption that exons will be well conserved, whereas introns will not. Programs are e.g. CEM (conserved exon method) and Twinscan.

This section gives detailed information about the algorithm followed by this tool as well as its pseudocode. This section includes Data Flow diagrams, UML diagrams, functional analysis and flow-chart. The user has to select one of the DNA sequence in Sequence Editor. User can perform the operation using Edit menu like reverse, reverse complement, double stranded. The following snapshot specifies the sequence editor feature of Gene Tool. User can directly open the sequence in Sequence Analysis Window to view the sequence in different format such as Double Stranded, Uppercase, Bold, Underline, Italic and he can also see the sequence in different group size. Also analyze menu having facility to generate the ORF of any nucleotide sequence, i.e. useful for the gene prediction.

### 3.8.1 Code for Translate DNA sequence to protein sequence.

Declaration of 64 codon using 20 amino acids

| **The table shows the 64 codons and the amino acid for each. The direction of the mRNA is 5' to 3'.** | | | | | |
|---|---|---|---|---|---|
| | | **2nd base** | | | |
| | | **U** | **C** | **A** | **G** |
| **1st base** | **U** | UUU (Phe/F)<br>UUC (Phe/F)<br>UUA (Leu/L)<br>UUG (Leu/L) | UCU (Ser/S)<br>UCC (Ser/S)<br>UCA (Ser/S)<br>UCG (Ser/S) | UAU (Tyr/Y)<br>UAC (Tyr/Y)<br>UAA Ochre (*Stop*)<br>UAG Amber (*Stop*) | UGU (Cys/C)<br>UGC (Cys/C)<br>UGA Opal (*Stop*)<br>UGG (Trp/W) |
| | **C** | CUU (Leu/L)<br>CUC (Leu/L)<br>CUA (Leu/L)<br>CUG (Leu/L) | CCU (Pro/P)<br>CCC (Pro/P)<br>CCA (Pro/P)<br>CCG (Pro/P) | CAU (His/H)<br>CAC (His/H)<br>CAA (Gln/Q)<br>CAG (Gln/Q) | CGU (Arg/R)<br>CGC (Arg/R)<br>CGA (Arg/R)<br>CGG (Arg/R) |
| | **A** | AUU (Ile/I)<br>AUC (Ile/I)<br>AUA (Ile/I)<br>AUG (Met/M) Start | ACU (Thr/T)<br>ACC (Thr/T)<br>ACA (Thr/T)<br>ACG (Thr/T) | AAU (Asn/N)<br>AAC (Asn/N)<br>AAA (Lys/K)<br>AAG (Lys/K) | AGU (Ser/S)<br>AGC (Ser/S)<br>AGA (Arg/R)<br>AGG (Arg/R) |
| | **G** | GUU (Val/V)<br>GUC (Val/V)<br>GUA (Val/V)<br>GUG (Val/V) | GCU (Ala/A)<br>GCC (Ala/A)<br>GCA (Ala/A)<br>GCG (Ala/A) | GAU (Asp/D)<br>GAC (Asp/D)<br>GAA (Glu/E)<br>GAG (Glu/E) | GGU (Gly/G)<br>GGC (Gly/G)<br>GGA (Gly/G)<br>GGG (Gly/G) |

| Inverse table | | | |
|---|---|---|---|
| **Ala/A** | GCU, GCC, GCA, GCG | **Leu/L** | UUA, UUG, CUU, CUC, CUA, CUG |
| **Arg/R** | CGU, CGC, CGA, CGG, AGA, AGG | **Lys/K** | AAA, AAG |
| **Asn/N** | AAU, AAC | **Met/M** | AUG |
| **Asp/D** | GAU, GAC | **Phe/F** | UUU, UUC |
| **Cys/C** | UGU, UGC | **Pro/P** | CCU, CCC, CCA, CCG |
| **Gln/Q** | CAA, CAG | **Ser/S** | UCU, UCC, UCA, UCG, AGU, AGC |
| **Glu/E** | GAA, GAG | **Thr/T** | ACU, ACC, ACA, ACG |
| **Gly/G** | GGU, GGC, GGA, GGG | **Trp/W** | UGG |
| **His/H** | CAU, CAC | **Tyr/Y** | UAU, UAC |
| **Ile/I** | AUU, AUC, AUA | **Val/V** | GUU, GUC, GUA, GUG |
| **START** | AUG | **STOP** | UAG, UGA, UAA |

**Gene prediction in prokaryotes**

In prokaryotic cells most of the DNA sequence is coding for genes/proteins. In comparison, almost 70% of the genome of H. influenzae is coding, while only about 3-5% of the human genome codes for proteins. Also the gene structure is quite different, e.g. there are no introns in the coding regions in prokaryotic genes:

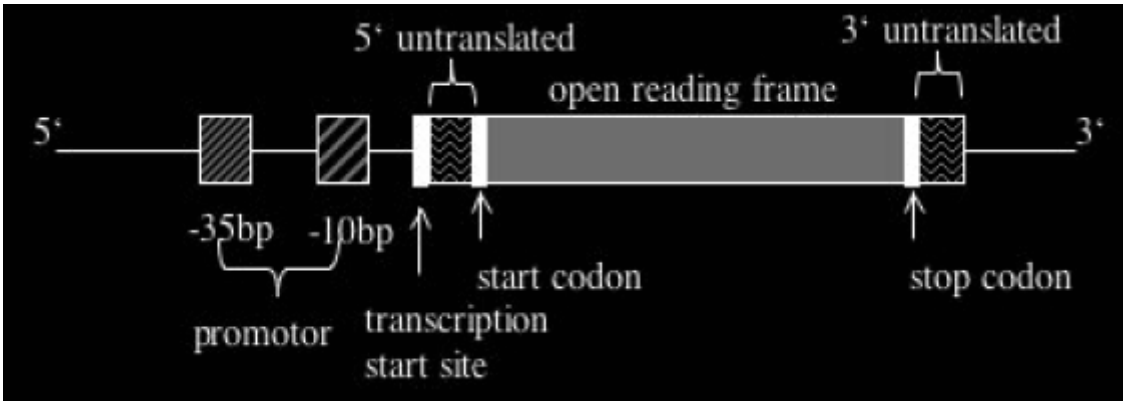

Fig. 3.13 - ORF position in nucleotide sequence

During the transcription process the RNA polymerase copies one DNA strand into the mRNA. The polymerase attaches at the transcription start site (TSS) to the DNA from which the transcription is started and stops the transcription when the signal for the transcription end is reached. The signal or pattern for the transcription start is (mostly) upstream of the start codon, and the one for the transcription end downstream of the stop codon. Thus there are regions at both ends of the coding region that become transcribed but not translated, thus they are called untranslated regions. Again upstream of the TSS is a regulatory region that contains the promoters.

**ORF prediction**

The simplest way to detect potential coding regions is to look at Open Reading Frames (ORFs). An ORF is a sequence of codons in DNA that starts with a Start codon (ATG), ends with a Stop codon (TAA, TAG or TGA) and has no other (in-frame) stop codons inside.

**Evaluate lengths of ORFs:**

The average distance between stop codons in “random” DNA is 64 / 3 ≈ 21, much smaller than the number of codons in an average protein (≈300).
An algorithm would then take a given DNA sequence and search within each of the possible reading frames stop codons and its corresponding start codon. For each such potential ORF determine the length and evaluate that.

**Evaluate codon usage:**

Here we use the fact that codon usage in coding regions differs substantially from that in non-coding regions. A number of these measures have been proposed, such as codon usage or hexamer counts. The codon usage of a string of DNA is given by a 64-component vector that counts how many times each codon is present in the string.

Example Codon Preference in E. Coli:

| **AA** | **Codon** | **1000** |
|---|---|---|
| Gly | GGG | 1.89 |
| Gly | GGA | 0.44 |
| Gly | GGU | 52.99 |
| Gly | GGC | 34.55 |
| Glu | GAG | 15.68 |

| | | |
|---|---|---|
| Glu | GAA | 57.20 |
| Asp | GAU | 21.63 |
| Asp | GAC | 43.26 |

The in-phase hexamer feature measures the frequency of occurrence of oligonucleotides of length six in a specific reading frame. In a study by Fickett and Tung (1992) it has been shown to be the most effective. Hexamer counts are mostly modeled as fifth-order Hidden Markov Models.

**Fifth-order:** $P(X_n = s| \cap_j<_n X_j) = P(X_n = s|Xn-_1X_n-_2X_n-_3X_n-_4X_n-5)$

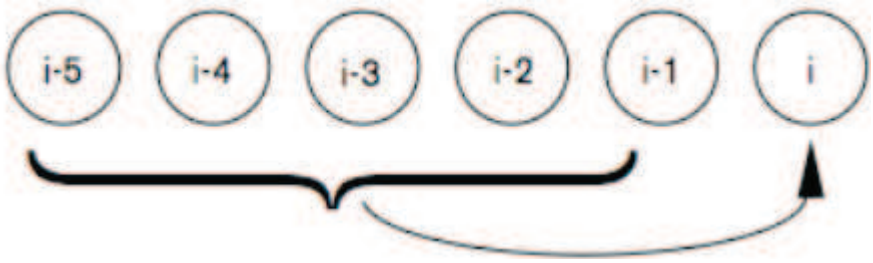


Fig. 3.14- Periodic fifth order Markov model. Circles represent consecutive DNA bases, numbers indicate codon position, and the arrows indicate that the next base is generated conditionally on the previous five and on the codon position.

**Codon preference:**

For each reading frame a codon preference statistics1 at each position is computed. The statistic is calculated over a window of length lw (lw is usually between 25 and 50), where the window is moved along the sequence in increments of three bases, maintaining the reading frame. The magnitude of the codon preference statistic is a measure of the likeness of a particular window of codons to a predetermined preferred usage.

The statistic is based on the concept of synonymous codons. Synonymous codons are those codons specifying the same amino acid.

For example, the bases Leucine, Alanine and Tryptophan are coded by 6, 4 and 1 different codons respectively. So in a uniformly random DNA sequence, the bases should occur in the ratio 6:4:1. But in a protein they occur in a different ratio – eg. 6.9:6.5:1. Therefore coding DNA is not random.

A codon parameter is calculated for each codon c in the reading frame based on the codon's frequency of occurrence (fc) and the total number of occurrences of its synonymous family (Fc) in the codon frequency table, and the calculated occurrences of the codon (rc) and its synonymous family (Rc) in a random sequence with the same base composition as the sequence being analyzed. The codon preference statistic for each codon c, pc, is then given by:

$$p_c = \frac{f_c/F_c}{r_c/R_c}$$

A pc value of 1.0 indicates that a codon is used equally in the random sequence and the codon frequency table. Values greater than 1.0 indicate the codon is present at higher than the random frequency in the codon frequency table, and values less than 1 indicate a codon is present at less than the random frequency in the codon frequency table. The probability of the sequence in the window w is then

$$P(w) = \prod_{i=1}^{l_w} p_{c_i}$$

Again a log-based score is used and the codon preference statistic for each window (Pw) is given by

$$P_w = e^{(\sum_{i=1}^{l_w} \log p_{c_i})/l_w}$$

Since the statistic is strongly affected by codons whose occurrence is zero in the codon frequency table, these codons are assigned an occurrence of 1. This is equivalent to saying that a zero value in the table doesn't mean that these codons are never seen, it only means that they haven't been seen in Fc observations, and that the upper bound on their occurrence as a fraction of their synonymous family is 1/Fc.

**ORF prediction using Markov models and HMMs:**

There are many more ORFs than real genes. For example, the E. coli genome contains about 6600 ORFs but only about 4400 real genes. Here we want to briefly mention how a Markov model and an HMM can be used to distinguish between non-coding ORFs and real genes. In principle, a model as described for distinguishing CpG-islands can be set up, which can distinguish between coding and non-coding ORFs. One possibility is to model the DNA sequences as 64-states (alphabet thus consists now of 64 letters) Markov chains of codons. The transition probabilities are then the probabilities that a certain codon is followed by another codon in a coding ORF. We can thus compute the log-odds scores. Non-coding ORFs have log-odds distribution centered on zero (as can be seen from the following figure), from which one concludes that codon usage in such regions is essentially random.

Public S()() As String = {New String() {"TTT", "F"}, New String() {"TCT", "S"}, New String() {"TAT", "Y"}, New String() {"TGT", "C"}, New String() {"TTC", "F"}, New String() {"TCC", "S"}, New String() {"TAC", "Y"}, New String() {"TGC", "C"}, New String() {"TTA", "L"}, New String() {"TCA", "S"}, New String() {"TAA", "*"}, New String() {"TGA", "*"}, New String() {"TTG", "L"}, New String() {"TCG", ""}, New String() {"TAG", "*"}, New String() {"TGG", "W"}, New String() {"CTT", "L"}, New String() {"CCT", "P"}, New String() {"CAT",

"H"}, New String() {"CGT", "R"}, New String() {"CTC", "L"}, New String() {"CCC", "P"}, New String() {"CAC", "H"}, New String() {"CGC", "R"}, New String() {"CTA", "L"}, New String() {"CCA", "P"}, New String() {"CAA", "Q"}, New String() {"CGA", "R"}, New String() {"CTG", "L"}, New String() {"CCG", "P"}, New String() {"CAG", "Q"}, New String() {"CGG", "R"}, New String() {"ATT", "I"}, New String() {"ACT", "T"}, New String() {"AAT", "N"}, New String() {"AGT", "S"}, New String() {"ATC", "I"}, New String() {"ACC", "T"}, New String() {"AAC", "N"}, New String() {"AGC", "S"}, New String() {"ATA", "I"}, New String() {"ACA", "T"}, New String() {"AAA", "K"}, New String() {"AGA", "R"}, New String() {"ATG", "M"}, New String() {"ACG", "T"}, New String() {"AAG", "K"}, New String() {"AGG", "R"}, New String() {"GTT", "V"}, New String() {"GCT", "A"}, New String() {"GAT", "D"}, New String() {"GGT", "G"}, New String() {"GTC", "V"}, New String() {"GCC", "A"}, New String() {"GAC", "D"}, New String() {"GGC", "G"}, New String() {"GTA", "V"}, New String() {"GCA", "A"}, New String() {"GAA", "E"}, New String() {"GGA", "G"}, New String() {"GTG", "V"}, New String() {"GCG", "A"}, New String() {"GAG", "E"}, New String() {"GGG", "G"}}

### 3.8.2 Sample of VB.Net Code for implementation

```
Public Function Translate(ByVal Tseq As String) As String
    Dim afile As System.IO.File
    Dim arfile As System.IO.StreamReader
    Dim i, k, j, l As Integer
    Dim x As String
    Dim T As String
    Dim fl As Boolean = True
    Dim rm, rs As Integer

    i = 0
    If fl Then

      For i = 0 To Tseq.Length - 1
        If i Mod 3 = 0 Then
```

```
            For k = i To i + 2
                x = String.Concat(x, Tseq.Chars(k))
            Next
        End If

        For j = 0 To 63
            If x.ToUpper = Declaration.S(j)(0) Then
                T = String.Concat(T, Declaration.S(j)(1))
                Exit For
            End If
        Next
        x = ""
        i = i + 2
    Next

  End If
  Return T
End Function
```

**Pseudocode for double stranded nucleotide sequence from linear**

```
function doubleStrand()
/* Algorithm for double strand conversion */

    if (doubleStr (Checked) equals 'true') then
        l1 := 8
        while (l1 := 0 to seqLength) do
            s1 := concatenation (s1, seqLength)
            l1 := l1 + 68
        end while
        while (i := length (s1) – 1) do
            x := chars (s1)
            if (x notequals " ") then
                s := concatenation (s, x)
            end if
        end while
        while (l := 0 to seqLength) do
            select seqLength (l, 58)
            color := blue
            l := l + 36
        end while
```

```
        end if
```

**Pseudocode for ORF prediction from nucleotide sequence**

```
function translate_load (sender , e)
/* Algorithm for translation */
        l1 := 8
        while (l1 := 0 to seqTextLength) do
                select length
                trans := concatenation (trans, seqTextLength)
        end while
        i := 0
        while (i := 0 to length (trans) – 1)
                x := trans.chars (i)
                if (x notequals “ ”) then
                        s := concatenation (s, x)
                end if
                increment i
        end while
        tr := translate (s)
        copy translate (s) to TextBox
```

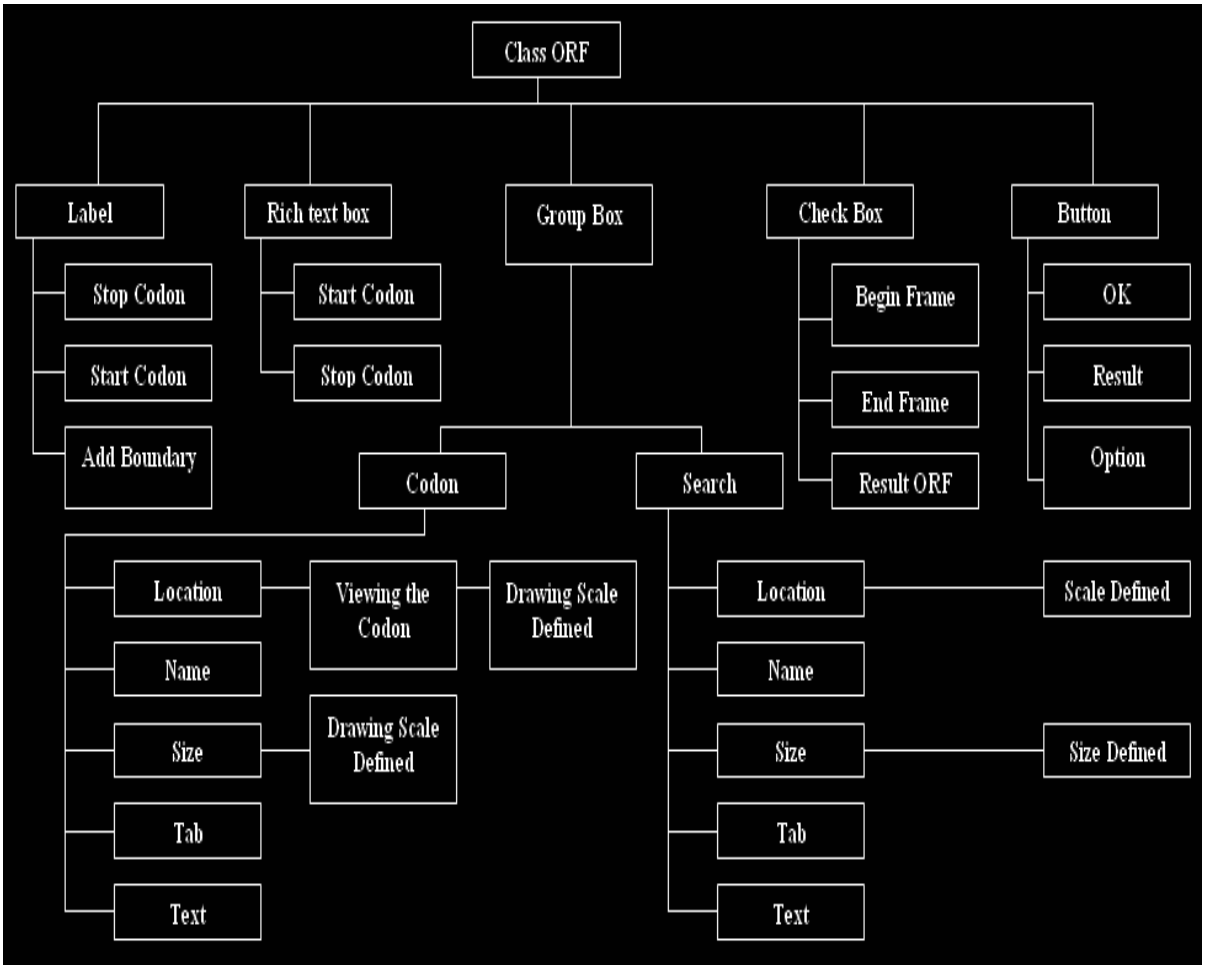


Fig. 3.15- Class diagram of ORF prediction

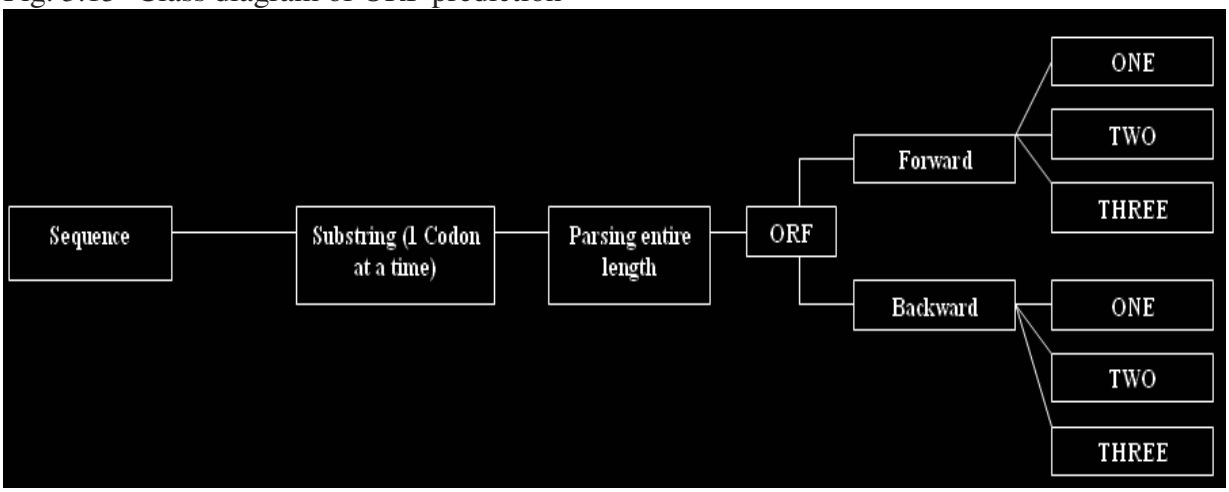


Fig. 3.16- Protocol for ORF prediction

### 3.9 Testing

#### 3.9.1 Manual Testing

Screen Name: Gene Tool Launcher

**Features to be tested:**

**1. User Interface:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Navigational interfaces using TAB key | Should be sequential | Yes |
| 2 | Tool tip | Properly defined | Yes |
| 3 | Font style | MS Serif font is used | Yes |

**2. Testing the functionality:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Open Button | Should open OpenDialougBox to select the DNA Sequence. | Yes |
| 2 | Help Button | Should open help about Gene Tool | Yes |

**Screen Name: Sequence Editor**

**Features to be tested:**

**1. User Interface:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Navigational interfaces using shortcuts key | Should be particular. | Yes |
| 2 | Tool tip | Properly defined | Yes |
| 3 | Font style | MS Serif font is used | Yes |

**2. Generic Test Conditions:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Sequence field | Should contain only DNA and protein sequence. | Yes |

**Screen Name: Sequence Analysis**
**Features to be tested:**

**1. User Interface:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Navigational interfaces using shortcuts key | Should be particular. | Yes |
| 2 | Tool tip | Properly defined | Yes |
| 3 | Font style | MS Serif font is used | Yes |

**2. Generic Test Conditions:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Sequence field | Should contain only DNA and protein sequence. | Yes |
| 2 | Check property | Should display particular result if check property is enabled. | Yes |
| 3 | Menu | Should open particular form or display particular result. | Yes |

**Screen Name: Translate**
**Features to be tested:**

**1. User Interface:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Navigational interfaces using shortcuts key | Should be particular. | Yes |
| 2 | Tool tip | Properly defined | Yes |
| 3 | Font style | MS Serif font is used | Yes |

**2. Generic Test Conditions:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Sequence field | Should contain only protein sequence. | Yes |
| 2 | Menu | Should open particular form or display particular result. | Yes |

**Screen Name: Comparison**
**Features to be tested:**

**1. User Interface:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Navigational interfaces using TAB key | Should be sequential. | Yes |
| 2 | Tool tip | Properly defined | Yes |
| 3 | Font style | MS Serif font is used | Yes |

**2. Generic Test Conditions:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Input file field | Should contain only file name. | Yes |

**3. Testing the functionality:**

| Test Case No | Condition to be tested | Expected Result | Actual Result |
|---|---|---|---|
| 1 | View Button | Should view particular sequence form. | Yes |
| 2 | Compare Button | Should display only comparison file with results. | Yes |

**3.10 Analysis**

**3.10.1 Functional Aspect**

**Level 1**

In level 1, after inputting the gene sequence the process analyzes the sequence taken from the input file by implementing the level 2 and level 3 functions, viz., File, Edit etc.

**Level 2**

In level 2, the *File* function shows the A, T, G, C gene Sequence in Sequence Editor using scale. *Edit* function shows the A, T, G, C gene Sequence in Sequence Editor with different formats as Reverse sequence, Double Stranded Sequence. *View* function shows the gene Sequence in Layout Sequence Editor. For doing different operations on it as Grouping, Uppercase, Bold etc. *Analyze* function analyzes the Gene Sequence and translates it into protein sequence and also shows the compute static of the sequence. *Help* function shows the Help for Gene Tool. It Guides the user how to use the Gene Tool.

**Level 3**

In level 3, *Reverse* function shows the reverse sequence. *Reverse Complement* function shows the reverse complement of the sequence. *Double Stranded* function

shows the double stranded of sequence. *Translate* function shows the protein sequence of input sequence.

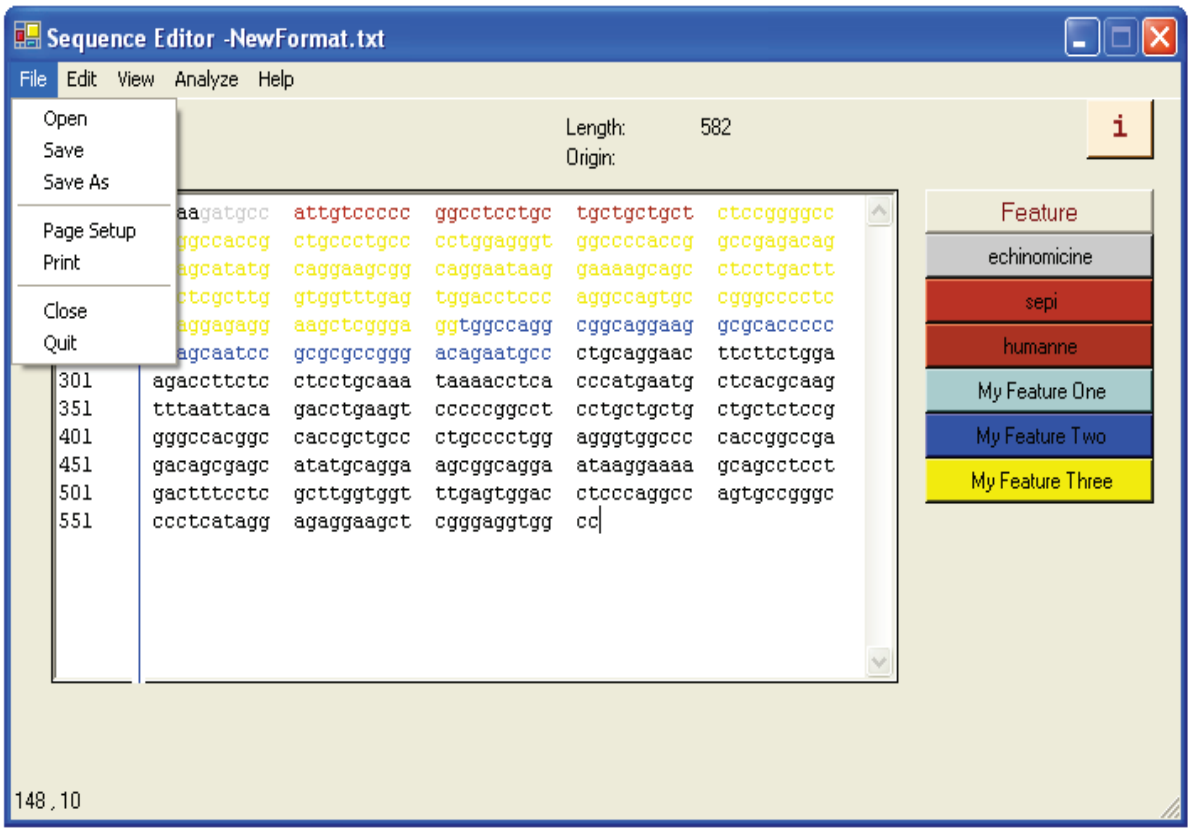


Fig. 3.17- Tool represents four operations that it does, including edit, view, analyze and help. The further sub-operations performed by edit, view, analyze and help.

### 3.10.2 Software analysis

**Following menus are provided for Gene Tool**

- Select all-Select the entire DNA sequence.
- Cut-Cut the selected DNA sequence from the sequence editor.
- Copy-Copy the selected DNA sequence from the sequence editor.
- Paste-Paste the contents of foreign sequence file into sequence editor.
- Double stranded-Entire DNA sequence converted into Double stranded form.
- E.g. A T C G Single Strand
- T A G C Double Strand
- Reverse-Reverse the order of bases in the selected DNA sequence.
- E.g. G G A A T T T T Original Sequence
- T T T T A A G G Reverse of original
- *Note:* Result is not equivalent representation of the same double
- Strand DNA molecule.
- Reverse Complement- Reverse and complement the selected DNA sequence.
- In complement A is replace by T and G is replace by C and vice –versa.
- For e.g. A T G C Original DNA sequence
- C G T A Reverse of original sequence
- G C A T Complement
- *Note:* Result is equivalent representation of the same double strand DNA molecule.

### 3.11 Snapshots

The user has to select one of the DNA sequence in Sequence Editor. User can perform the operation using Edit menu like reverse, reverse complement, double stranded. The following snapshot specifies the sequence editor feature of Gene Tool.

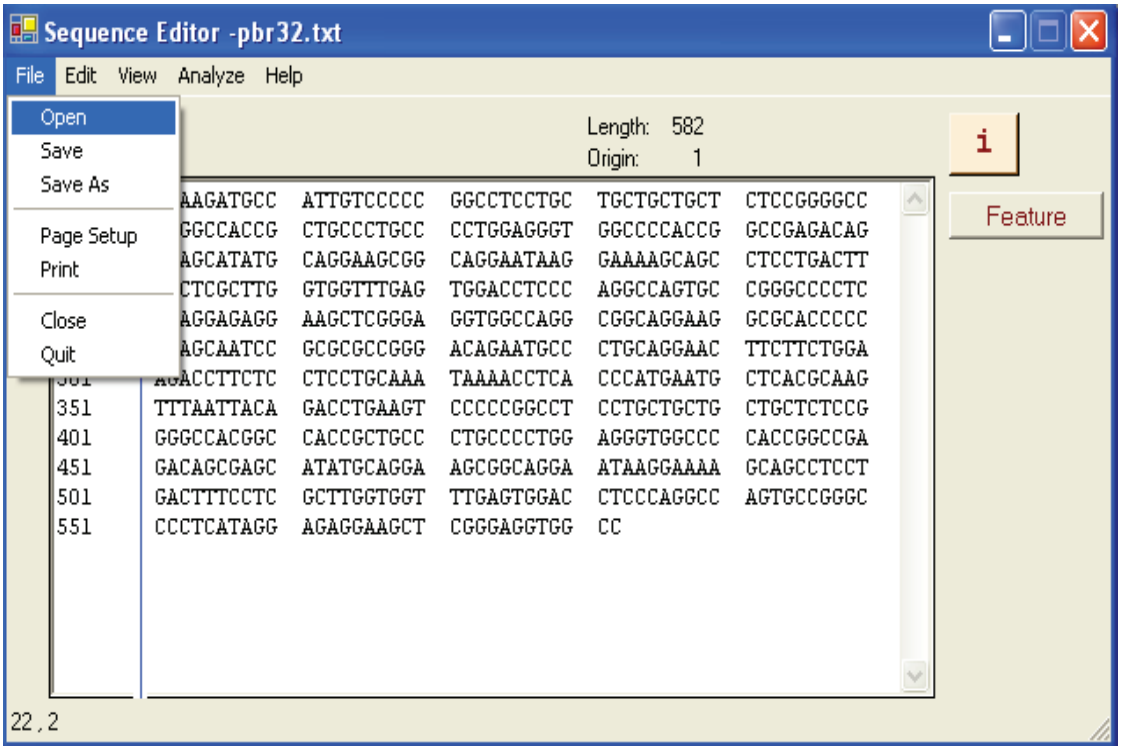


Fig. 3.18- Sequence Editor Front view of Gene Tool

User can directly open the sequence in Sequence Analysis Window to view the sequence in different format such as Double Stranded, Uppercase, Bold, Underline, Italic and he can also see the sequence in different group size.

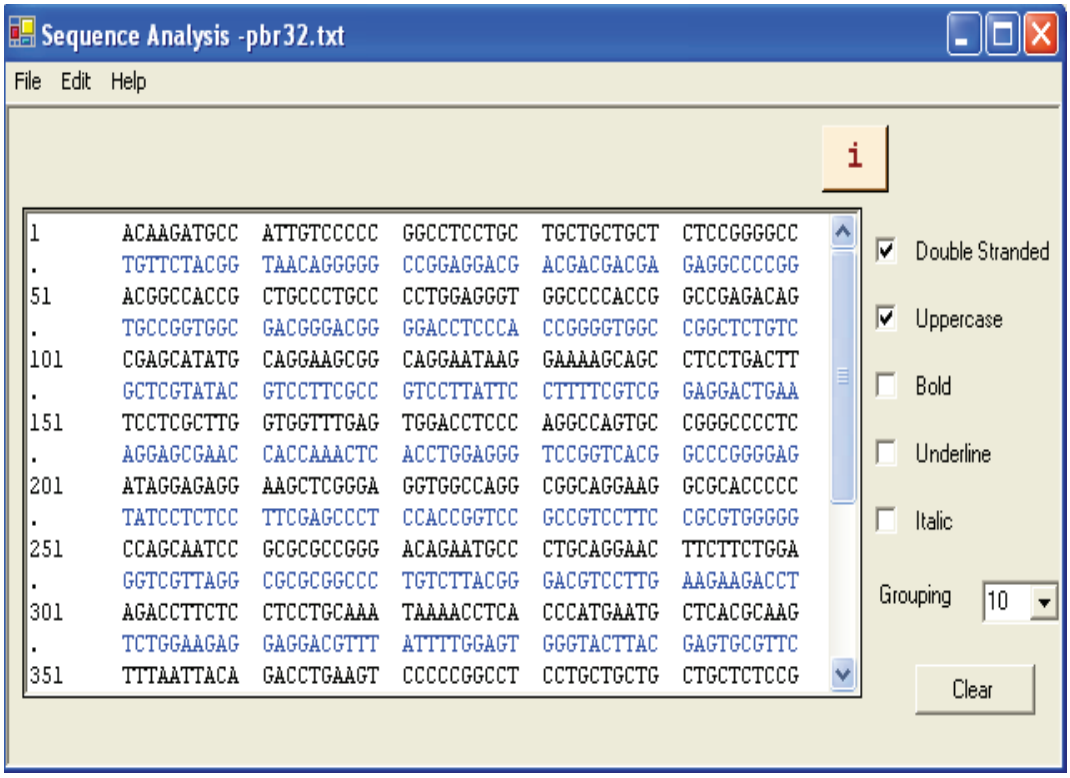


Fig. 3.19- Sequence Editor View for sequence analysis

**Prediction ORF**

Gene Tool is helpful for those scientists who want a quick result before research and it also helpful for ORF prediction in genetic engineering and molecular biology. The user can compare the original sequence and the input sequence. Also the DNA sequence can be analyzed for later reference.

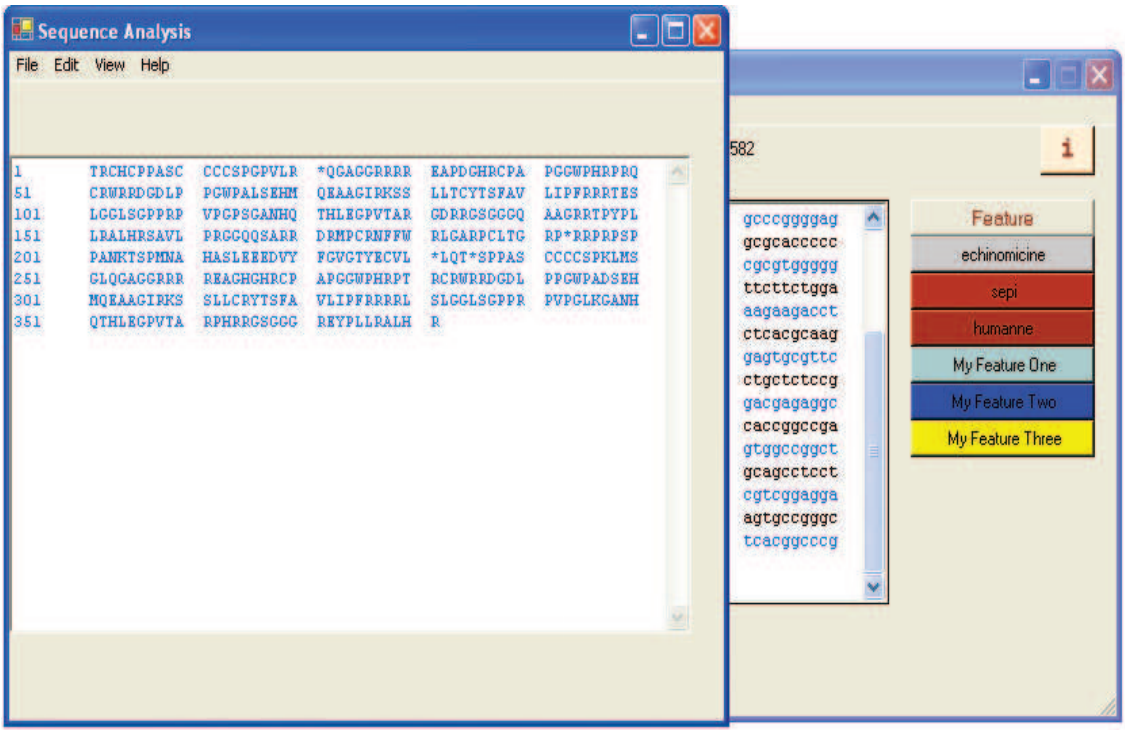


Fig. 3.20- Translate option in analyze menu convert DNA sequence into ORF

Here the user has given the flexibility to compare the sequence entered by the User and original sequence. By clicking on compare button he can directly see the input sequence with the error display in red color. In Result txt box he can also see the error, which are occurred in respective position.

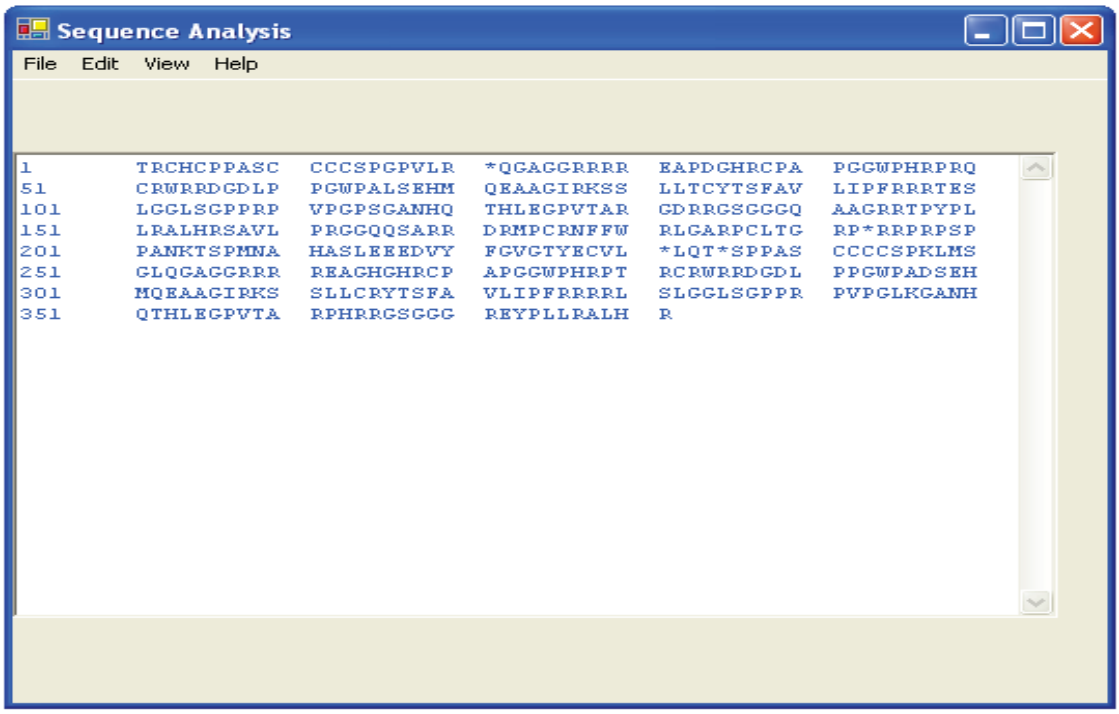

Fig. 3.21- Translate option in analyze menu convert DNA sequence into ORF-1

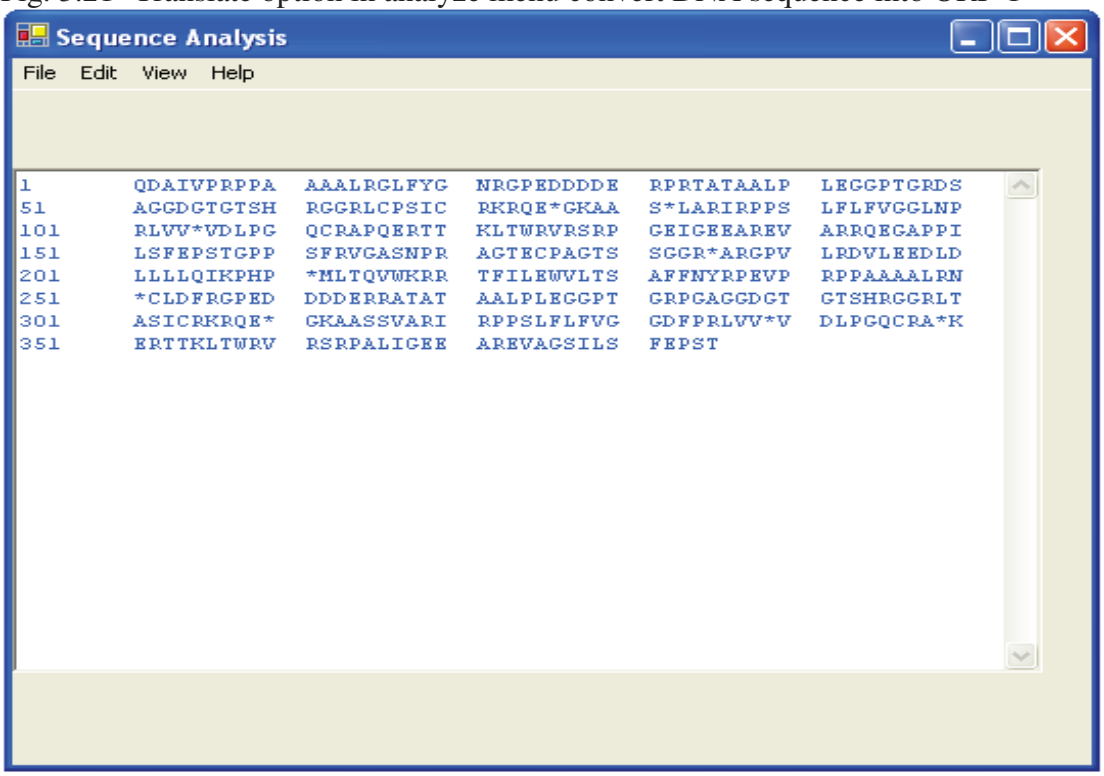

Fig. 3.22- Translate option in analyze menu convert DNA sequence into ORF-2

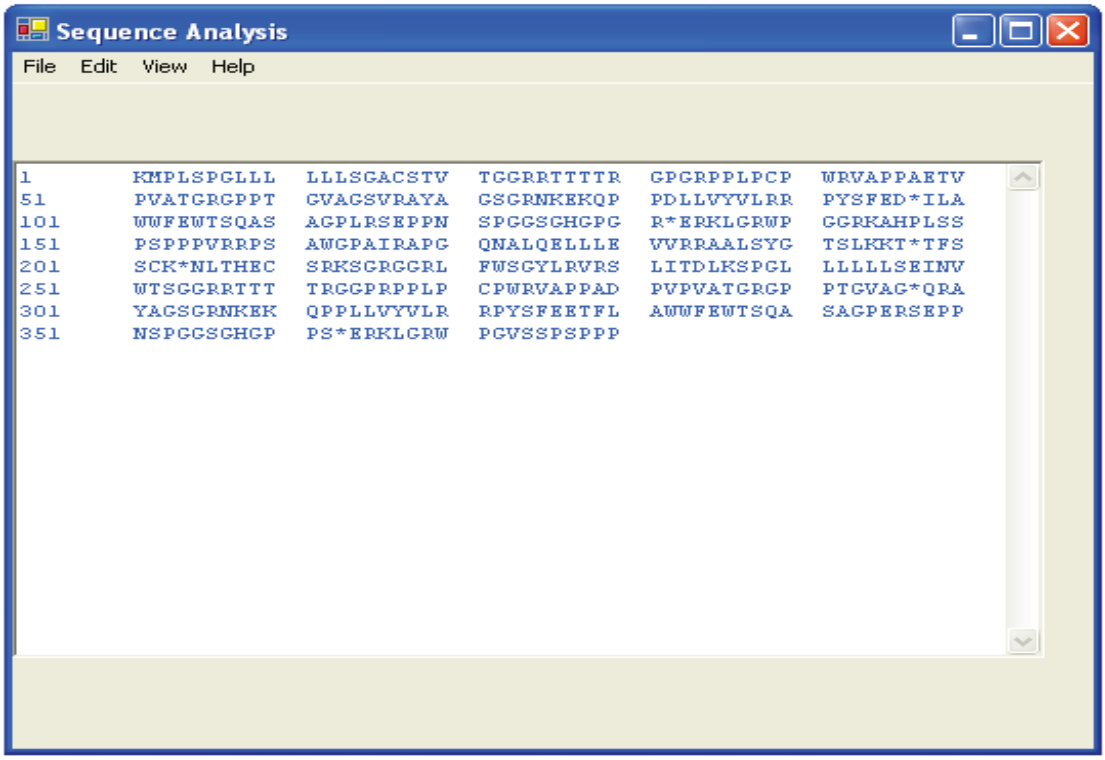


Fig. 3.23- Translate option in analyze menu convert DNA sequence into ORF-3

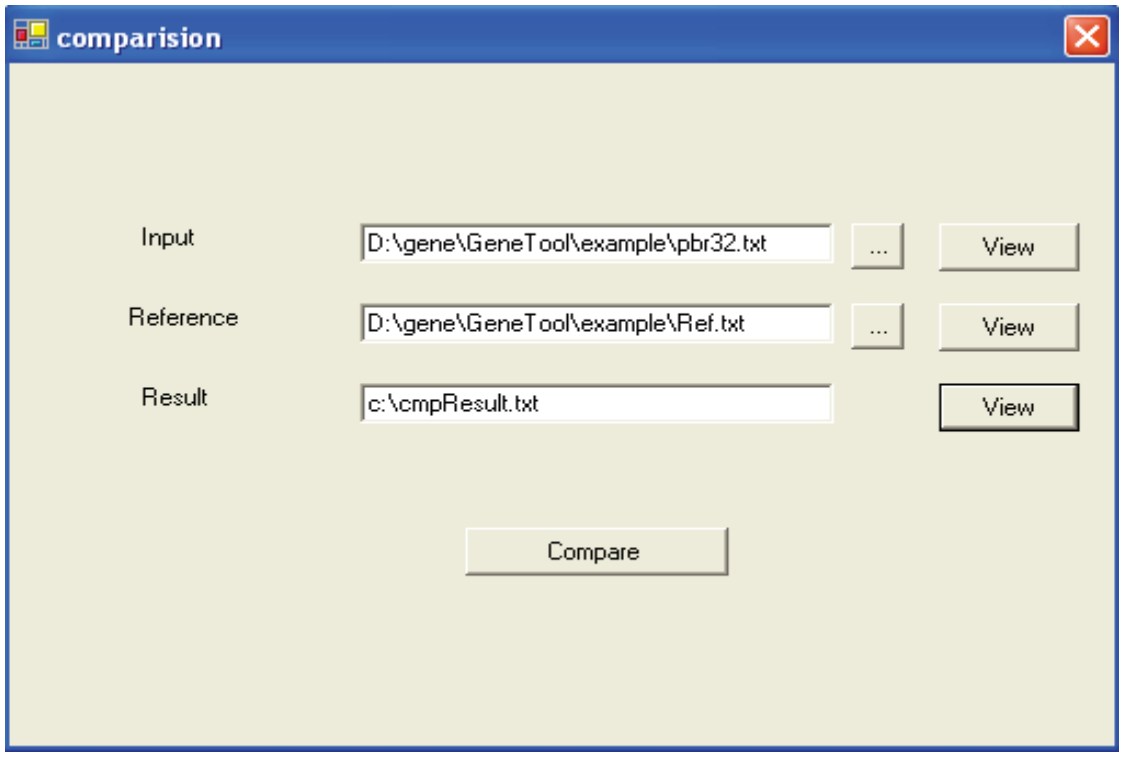


Fig. 3.23- Comparison of two sequences

### 3.12 Theoretical overview

Bioinformatics encompasses fields such as comparative genomic, structural genomic, transcriptiomics, Proteomics, cellunomics and metabolic pathway engineering. Developments in these fields have direct implications to healthcare, medicine, discovery of next generation drugs, development of agricultural products, renewable energy, environmental protection etc. It is also important in protein engineering, pharmacogenomics, discovery of new drugs and vaccines, molecular diagnostic kits, agro-biotechnology etc.

### 3.13 Implementation

Visual Basic .NET delivers new productivity features for building more robust applications easily and quickly. With an improved integrated development environment and a significantly reduced startup time, it offers fast, automatic formatting of code as one type in. With this one can build applications more rapidly and deploy and maintain them with greater efficiency. One can also create reusable, enterprise-class code using full object-oriented constructs. Language features include full implementation inheritance, encapsulation, and polymorphism. Structured exception handling provides a global error handler and eliminates spaghetti code. This section deals with the major implementation criteria including Code for Translate DNA sequence to protein sequence, Sample of VB.Net Code, testing and snapshots.

### 3.14 Software Usage

The user has to select one of the DNA sequence in Sequence Editor. User can perform the operation using Edit menu like reverse, reverse complement, double stranded of Nucleotide sequence. Here the user has given the flexibility to compare the sequence entered by the User and original sequence. By clicking on compare button he can directly see the input sequence with the error display in red color. In Result txt box he can also see the errors, which are occurred in respective position. User can directly open the sequence in Sequence Analysis Window to view the sequence in different format such as Double Stranded, Uppercase, Bold, Underline, Italic and he can also see the sequence in different group size. Translate option in analyze menu convert DNA sequence into protein sequence.

### 3.15 Summary

Gene Tool is helpful for those scientists who want a quick result before research and it also helpful for ORF prediction in genetic engineering and molecular biology. The user can compare the original sequence and the input sequence. Also the DNA sequence can be analyzed for later reference. Gene tool deals with DNA sequence assembly; gene finding and analysis, Protein expression and regulation. Gene predicts examine some of the application of computational biology. Gene predicts allows researchers to precisely adopt genes and gene products to suite their specific requirement. Gene Tool also includes the various operations as Reverse sequence, Double Stranded sequence, Complement of sequence, Features Information that helps the researcher to find out the function of particular gene sequence.

### 3.16 References


[1] P.Baldi and S. Brunak , "Bioinformatics: The Machine Learning Approach", MIT Press, ISBN 0-262-02506-X, 2001.

[2] D.W. Mount, Bioinformatics sequence and genome analysis (Cold Spring Harbor Laboratory Press: 2nd edition, ISBN: 0879696087, 2004).

[3] S.F. Smith, "Homology search with binary and trinary scoring matrices", *Int. J. Bioinform. Res. Appl.,* 2(2), 119-31, 2006.

[4] D.A. Benson, I. Karsch-Mizrachi, D.J. Lipman, J. Ostell, D.L. Wheeler, "GenBank", *Nucleic Acids Res*., 35(Database issue), D21-D25, 2007.

[5] H. Sugawara, O. Ogasawara, K. Okubo, T. Gojobori, Y. Tateno, "DDBJ with new system and face", *Nucleic Acids Res*., 36(Database issue), D22-D24, 2008.

[6] G. Cochrane, P. Aldebert, N. Althorpe, M. Andersson, W. Baker, A. Baldwin, K. Bates, S. Bhattacharyya, P. Browne, A. van den Broek, M. Castro, K. Duggan, R. Eberhardt, N. Faruque, J. Gamble, C. Kanz, T. Kulikova, C. Lee, R. Leinonen, Q. Lin, V. Lombard, R. Lopez, M. McHale, H. McWilliam, G. Mukherjee, F. Nardone, M.P. Pastor, S. Sobhany, P. Stoehr, K. Tzouvara, R. Vaughan, D. Wu, W. Zhu, R. Apweiler, "EMBL Nucleotide Sequence Database: developments in 2005", *Nucleic Acids Res*., 34(Database issue), D10-D15, 2006.

[7] W. Gish and D.J. States, "Identification of protein coding regions by database similarity search", *Nature Genet*., 3, 266-272, 1993.

[8] S. Karlin and S.F. Altschul, "Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes", *Proc. Natl. Acad. Sci. USA*, 87, 2264-2268, 1998.

[9] S.F. Altschul, W. Gish, W. Miller, E.W. Myers, D.J. Lipman, "Basic local alignment search tool", *J. Mol. Biol.*, 215, 403–410, 1990.

[10] W.R. Pearson and D.J. Lipman, "Improved Tools for Biological Sequence Comparison", *PNAS*, 85, 2444-2448, 1988.

[11] W.R. Pearson, "Rapid and Sensitive Sequence Comparison with FASTP and FASTA", *Methods in Enzymology*, 183, 63-98, 1990.

[12] W.R. Pearson, "Training for bioinformatics and computational biology", *Bioinformatics*, 17(9), 761-762, 2001.

[13] T.A. Tatusova, T.L. Madden, "Blast 2 sequences - a new tool for comparing protein and nucleotide sequences", *FEMS Microbiol Lett.*, 174, 247-250, 1999.

[14] J.D. Retief, K.R. Lynch, W.R. Pearson, "Panning for genes--A visual strategy for identifying novel gene orthologs and paralogs", *Genome Res*., 9(4), 373-382, 1999.

[15] J. Besemer, M. Borodovsky, "GeneMark: web software for gene finding in prokaryotes, eukaryotes and viruses", *Nucleic Acids Res*., 33(Web Server issue), W451-W454.

[16] R.K. Azad, M. Borodovsky, "Effects of choice of DNA sequence model structure on gene identification accuracy", *Bioinformatics*, 20(7), 993-1005, 2004.

[17] M. Borodovsky and J. McIninch, "GeneMark: parallel gene recognition for both DNA strands", *Computers and Chemistry*, 17(19), 123-133, 1993.

[18] V. Brendel, P. Bucher, I. Nourbakhsh, B.E. Blaisdell, S. Karlin, "Methods and algorithms for statistical analysis of protein sequences", *Proceedings of the National Academy of Sciences USA 89,* 2002-2006, 1992.

[19] Z. Zhang, S. Schwartz, L. Wagner, W. Miller, "A greedy algorithm for aligning DNA sequences", *J Comput Biol 2000*, 7(1-2), 203-14, 2000.

[20] Alex Horovitz, Jesse Liberty, Programming .NET 3.5 Rough Cuts Version (O'Reilly Media, Inc.. ISBN 13:9780596510398, 2007).

[21] Cristian Darie, Zak Ruvalcaba, Build Your Own ASP.NET 2.0 Web Site Using C# & VB, Second Edition (O'Reilly Media, Inc.. ISBN 13:9780975240281, 2006).

[22] Jesse Liberty, Alex Horovitz, Getting Started with .NET 3.0 Writing Your First .NET 3.0 Application (O'Reilly Media, Inc.. ISBN 13:9780596529215, 2006).

[23] Daniel R. Clark, An Introduction to Object-Oriented Programming with Visual Basic .NET (Apress 2002).

[24] Harvey M. Deitel, Paul J. Deitel, Cheryl H. Yaeger, Tem R. Nieto, Visual Basic .NET For Experienced Programmers (Prentice Hall PTR, 2002).

[25] Ron Patton, Software Testing (2nd Edition, Sams, ISBN-10: 0672327988, 2005).

[26] William E. Perry, Effective Methods for Software Testing (Wiley, ISBN-10: 0764598376, 2006).