Domain Specific Language for Protein Sequence Analysis in BioInformatics

Submitted By:

Mohit Gupta 2012A7PS021P
Ronil Pancholia 2012C6PS629P

Contents

Description of Application Domain

Proteins are made up of Amino Acids (such as Lysine, Valine, Glysine etc.). Amino Acids are made up small components called nucleotides. Nucleotides are of 5 types : adenine (A), thymine (T), guanine (G), cytosine (C) and uracil (U)

A,T,G and C are present in DNA and A,U,G and C are present in mRNA. To convert the nucleotide sequence from DNA to mRNA we just have to replace T by U. This process is called Transcription. From mRNA, it is easy to find which Amino Acids are present. This process is called Translation.

Amino acids are actually formed by different permutations of A,T,G and C. And Proteins are formed by different permutaions of Amino Acids, so if the Amino acids are known, we can know which protein is present.

Codons are a combination of 3 nucleotides (start codon usually AUG and stop codon UAA, UGA or UAG) which may vary upon the type of organism. The given nucleotide sequence is usually a circular sequence so we donot usually know where to start matching Amino Acids from. Start Codons help us finding the start point. Similarly, there are Stop Codon which denote the end point.

It is difficult to deal with such neucleotide sequences with existing programming languages because they require a lot of string manipulations and biologists may feel uncomfortable in doing it using the existing programming languages.

Some more concrete reasons on why it is difficult to use existing programming languages in this domain will be clear as we show the uses of this programming language.

List of problems in your domain that can be solved by using your language

1. Identification of the type of protein molecule present can be done. Amino acids are actually formed by different permutations of A,T,G and C. And Proteins are formed by different permutaions of Amino Acids, so if the Amino acids are known, we can know which protein is present.
Suppose we are given a sequence as :-

GGUAA|AUG|AUCGUUGCAAAU|UGA|GACU (|start| and |stop| codon added just for understanding, user will not input the sequence with start or stop codon marked.)

So, the part "AUCGUUGCAAAU" will be used to match amino acids from a hash table which will directly give us that the protein is made of Isoleucine,Valine,Alanine and Asparagine. So, from this we can find out which protein is present.

NOTE : This is a very useful thing so, this can be made as an "in-built" function in the language.

2. This language can also be used in forensics to find out if the DNA belongs to the same person. Suppose we are given 2 proteins P1 and P2. We can compare them by matching the substrings of P1 and P2 and calculate what percentage of similarity is present in the sequences. If the % similarity is above some threshold, we can say that the protein belongs to the same person.

3. Restriction enzymes "cut" DNA in some specific sites. It is very useful to know where these cuts will be made. This can be found out by mapping a particular amino acid to the protein sequence by finding at what all places, the amino acid is present in the sequence. For this, we retrieve the 3-nucleotide sequence corresponding to the given amino acid from a hash table and check at what all places that amino acid is present.

4. This language could also be useful in solving cloning related problems. Cloning process involves making a “cut” in a protein sequence using restriction enzymes. This will give us a protein sequence which will be like a jigsaw puzzle. Since, the protein sequence was circular, it will now form a linear sequence. Now, we do the same with some other protein sequence. Next, we insert the 2^{nd} sequence into the 1^{st} sequence using an enzyme called “ligase”.

This insertion of one protein sequence into another sequence might be used more often(in other applications also) so we assign **“-+-” operator** for this insertion process to simplify it.

We may have to use different small parts of a single protein for cloning purposes (like adding a small sequence into a large sequence to get some specific properties of the 2^{nd} type and major structure of the 1^{st} type), so we also introduce **“-/-” operator** which cuts a sequence a given number of times. Example : “P-/-3” will cut the protein P at the first 3 restriction sites.

List of programming features that the language is providing

Readablity :

- Single line Comments are supported by using the % operator.
- As, the language will be used by biologist, we use “is” for assignment so that the language looks like English.

Writablity :

- Entries can be added and removed from the Hash Table using some basic functions like getAminoAcid(Seq s).
- Conditional Statements like if<condition> <stmt> else <stmt> are supported.
- A print function is provided to print the output on stdout which takes as argument the value to be printed.

Reliablity :

- All variables will be dealt with static type binding so that programmer is not confused.
- The language will have static typing so that the user is given all the type errors (if any) at compile time itself.

Portablity :

- The language is parsed by python, so it will be highly portable as it will work in all places where python is installed.

Cost :

- A hash table to map Amino Acids to Nucleotides is provided for Searching in constant time once we get the 3-nucleotide sequence which is a frequent operation. All other operations will be small artihmetic operations. So, the *execution speed will be fast*.
- As for memory management, the language will use an *automatic garbage collector* working on some Mark and Sweep type algorithm.

List of all token types along with all their allowable instances in your language

As biologists have limited (in many cases, no) exposure to Mathematics or Programming, we wanted to keep our data types simple.

Some derived data types we will use are :

1. Nucleotide : A derived data type which stores one of the 5 nucleotides adenine (A), thymine (T), guanine (G), cytosine (C) and uracil (U). But, we donot hard code them for flexiblity purposes. These are basically subunits of Amino Acids.

2. Amino Acid : A derived data type which stores 3 different nucleotides (order sensitive) and a bit identifying if it is an essential amino acid or a non-essential amino acid. There are 22 types of Amino Acids corresponding to different sequences of nucleotides. But, we donot hard code them for flexiblity purposes.

3. Protein : A derived data type which is basically a wrapper of a circular queue which can be converted into a linear queue which stores a sequence of nucleotides. It has a length associated to it.

4. Seq : A derived data type containing a Nucleotide Sequence of 3 nucleotides

5. int : Standard C-style primitive Integer data type.

Other Keywords :

for, with, in, is, if, else, end

Some operators we will use are :

"-+-" operator : The insertion of one protein sequence into another sequence

"-/-" operator : To cut a sequence a given number of times.

"(" and ")" : Standard paranthesis **"," :** To seperate different parameters

"<","<=",">=","==","!=" : Comparison operators

"%" : Insert Single line comment **";" :** End of line indicator

"++","--" : Increment and decrement operators

"+,-,*,/" : arithmetic operators **"," :** for simultaneous use of variables

"[","]" : for array access **"{","}" :** to specify scope of variable

Finding which protein is present

```
Protein P is GGUAAAUGAUCGUUGCAAAUUGAGACU
Seq startcodon is AUG
Seq stopcodon is UGA
AminoAcids A[]
int j is 0
int i is 0
for ( i is 0 ; i < P.length ; i++ )
{
     Seq s is P[i],P[i+1],P[i+2]
     if ( s == startcodon )
     {
          for ( j is i + 3 ; j < P.length - 2 ; j is j + 3 )
          {
               Seq s1 = P[j],P[j+1],P[j+2]
               if ( s1 == stopcodon )
               {
                    end
               }
               print ( getAminoAcid (s1) )
          }
     }
}
end
```

Matching two protein sequences (Forensics)

Assume that we have a function getSequence(Protein) which can get us the amino acid sequence from the protein (function implemented in previous code).

```
Protein P1 is GGUAAAUGAUCGUUGCAAAUUGAGACU

Protein P2 is AGUAAAUGAGCGUUGCAAAUUGAGACG

Seq startcodon is AUG

Seq stopcodon is UGA

AminoAcid A1[]

AminoAcid A2[]

A1 is getSequence(P1)

A2 is getSequence(P2)

int i is 0

int match is 0

for (i is 0 ; i < A1.length ; i++)

{

     if ( A1[i] == A2[i] )

     {

          match is match + 1

     }

}

int percentageMatch is (match/A1.length)*100

if ( percentageMatch > 70 )

     { print ( “Match” )  }

else

     { print ( “Not Match” )  }

end
```

Mapping DNA to restriction sites

This program will give us the locations where restriction enzymes will cut the given protein. RestrictionSeq is determined by the particular restriction enzyme used.

```
Protein P is GGUAAAUGAUCGUUGCAGUUUGAGACU

Seq startcodon is AUG

Seq stopcodon is UGA

AminoAcid A[]

A is getSequence(P)

Seq restrictionSeq is GUU

restAA is getAminoAcid(restrictionSite)

for ( i is 0 ; i < A.length ; i++)

{

     if ( A[i] == restAA)

     {

          print ( i )

     }

}

end
```

Predicting Cloning Results

For this program, we assume that we have made a function getRestrictionSites(Protein P, Seq S) which returns us location of restriction sites (function implemented in previous code).

```
Protein P1 is GGUAAAUGAUCGUUGCAAAUUGAGACU

Protein P2 is AGUAAAUGAGCGUUGCAAAUUGAGACG

Seq startcodon is AUG

Seq stopcodon is UGA

AminoAcid A1[]

AminoAcid A2[]

A1 is getSequence(P1)

A2 is getSequence(P2)

Seq restrictionSeq is GUU

int site1 is getRestrictionSites(P1,restrictionSeq)

int site2 is getRestrictionSites(P2,restrictionSeq)

Protein P1_new is P1[0] to P1[site1]

Protein P2_new is P2[site2] to P2[P2.length]

Protein ClonedProtein

ClonedProtein is P1_new -+- P2_new

end
```