CAP 5510: Introduction to Bioinformatics

Giri Narasimhan

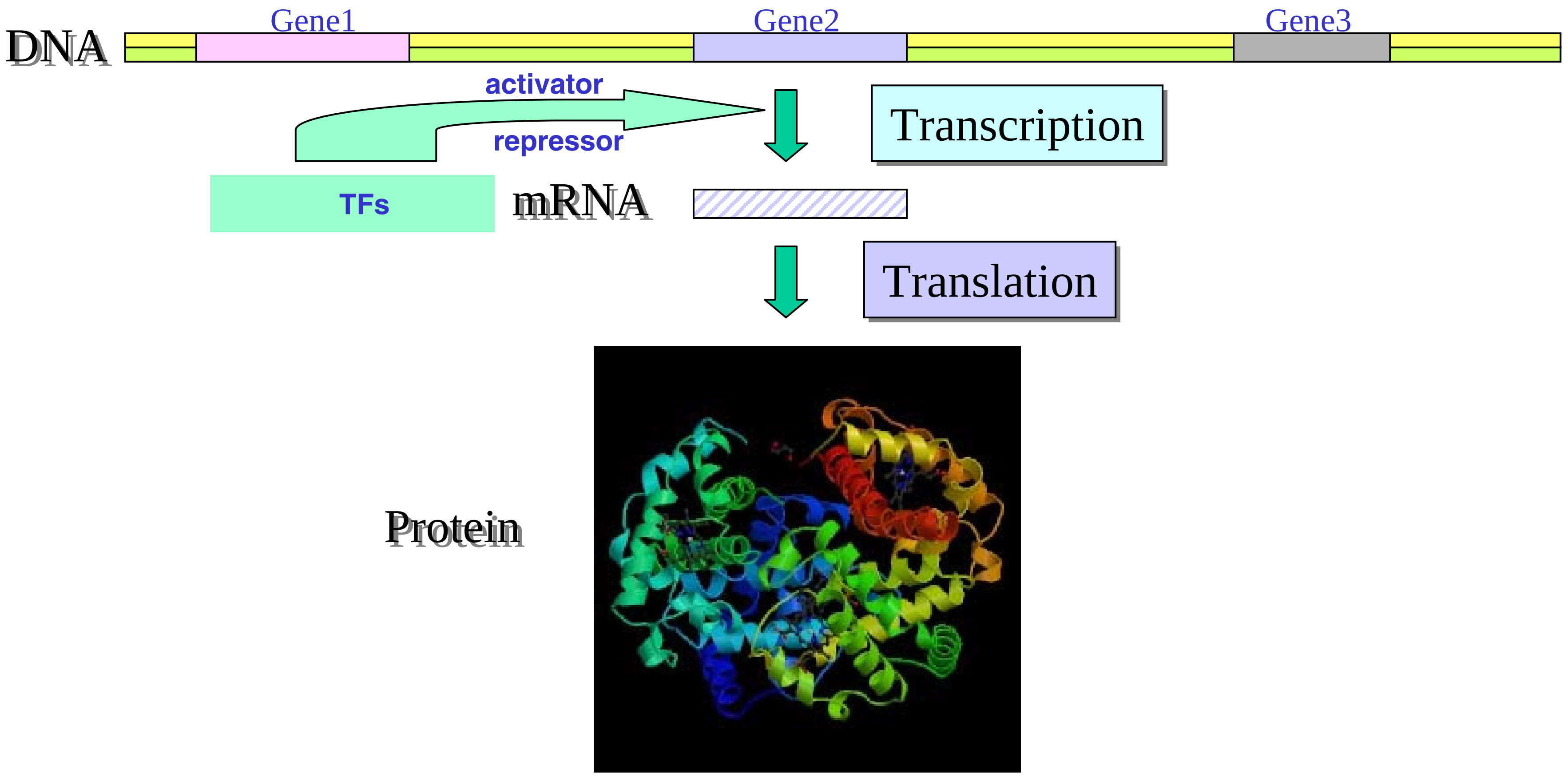
ECS 254; Phone: x3748

giri@cis.fiu.edu

www.cis.fiu.edu/~giri/teach/BioinfS07.html

Molecular Biology Background

Central Dogma



Enterobacteria phage lambda

```
ORIGIN      5' end of the ecori-g fragment, polarity of l strand.
        1 aattcttttg ctttttaccc tggaagaaat actcataagc cacctctgtt atttaccccc
       61 aatcttcaca agaaaaactg tatttgacaa acaagataca ttgtatgaaa atacaagaaa
      121 gtttgttgat ggaggcgata tgcaaactct ttctgaacgc ctcaagaaga ggcgaattgc
      181 gttaaaaatg acgcaaaccg aactggcaac caaagccggt gttaaacagc aatcaattca
      241 actgattgaa gctggagtaa ccaagcgacc gcgcttcttg tttgagattg ctatggcgct
      301 taactgtgat ccggtttggt tacagtacgg aactaaacgc ggtaaagccg cttaagacat
      361 tcccgctctt acacattcca gccctgaaaa agggcatcaa attaaaccac acctatggtg
      421 tatgcattta tttgcataca ttcaatcaat tgttatctaa ggaaatactt acatatggtt
      481 cgtgcaaaca aacgcaacga ggctctacga atcgagagtg cgttgcttaa caaaatcgca
      • • •
     1141 aatggtgcat ccctcaaaac gagggaaaat cccctaaaac gagggataaa acatccctca
     1201 aattggggga ttgctatccc tcaaaacagg gggacacaaa agacactatt acaaaagaaa
     1261 aaagaaaaga ttattcgtca gagaatt
//
```

DNA

Gene for protein Cro
140-355

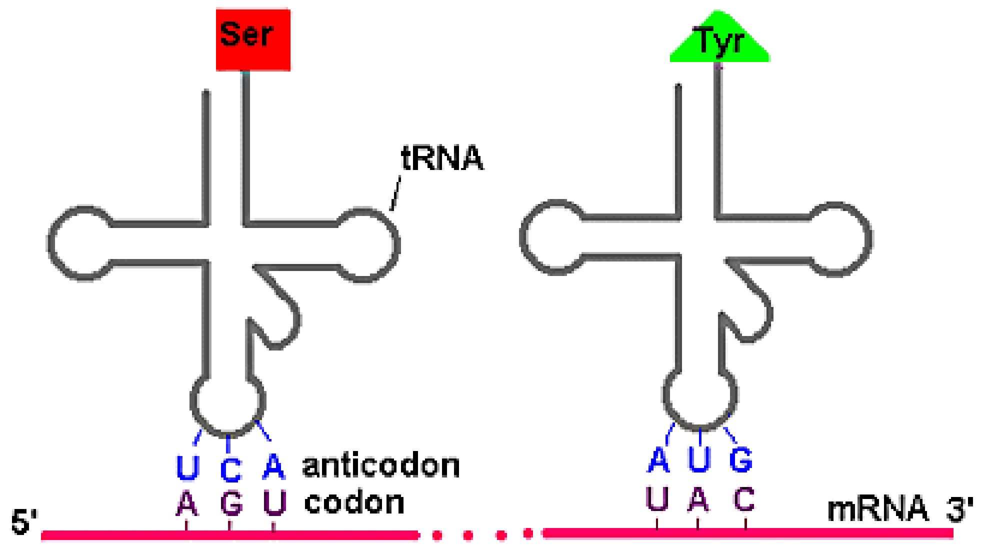
mRNA

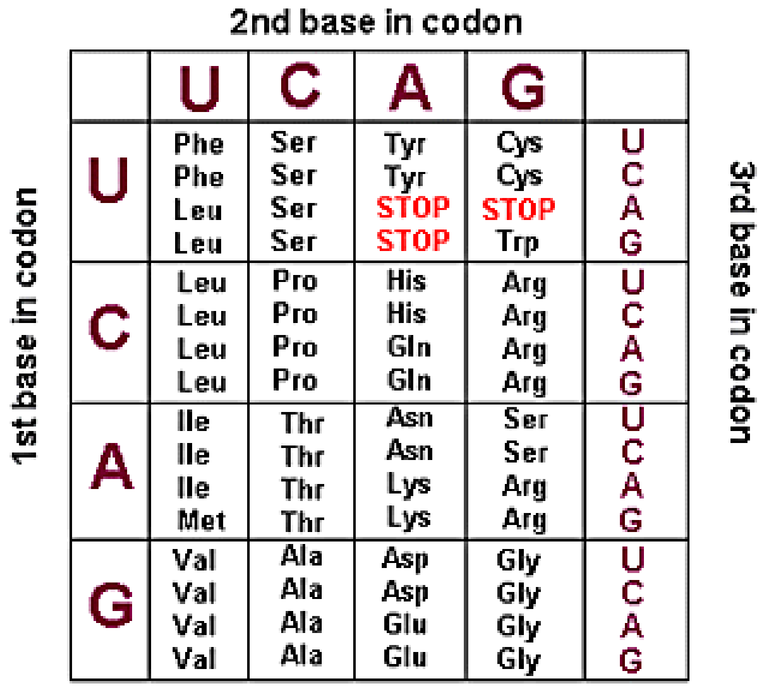
```
                         a ugcaaacucu uucugaacgc cucaagaaga ggcgaauugc
guuaaaaaug acgcaaaccg aacuggcaac caaagccggu guuaaacagc aaucaauuca
acugauugaa gcuggaguaa ccaagcgacc gcgcuucuug uuugagauug cuauggcgcu
uaacugugau ccgguuuggu uacaguacgg aacuaaacgc gguaaagccg cuuaa
```

```
ORIGIN
        1 mqtlserlkk rrialkmtqt elatkagvkq qsiqlieagv tkrprflfei amalncdpvw
       61 lqygtkrgka a
//
```

Protein

The Genetic Code





2nd base in codon

1st base in codon

3rd base in codon

	U	C	A	G	
U	Phe	Ser	Tyr	Cys	U
	Phe	Ser	Tyr	Cys	C
	Leu	Ser	STOP	STOP	A
	Leu	Ser	STOP	Trp	G
C	Leu	Pro	His	Arg	U
	Leu	Pro	His	Arg	C
	Leu	Pro	Gln	Arg	A
	Leu	Pro	Gln	Arg	G
A	Ile	Thr	Asn	Ser	U
	Ile	Thr	Asn	Ser	C
	Ile	Thr	Lys	Arg	A
	Met	Thr	Lys	Arg	G
G	Val	Ala	Asp	Gly	U
	Val	Ala	Asp	Gly	C
	Val	Ala	Glu	Gly	A
	Val	Ala	Glu	Gly	G

The Genetic Code

Enterobacteria phage lambda

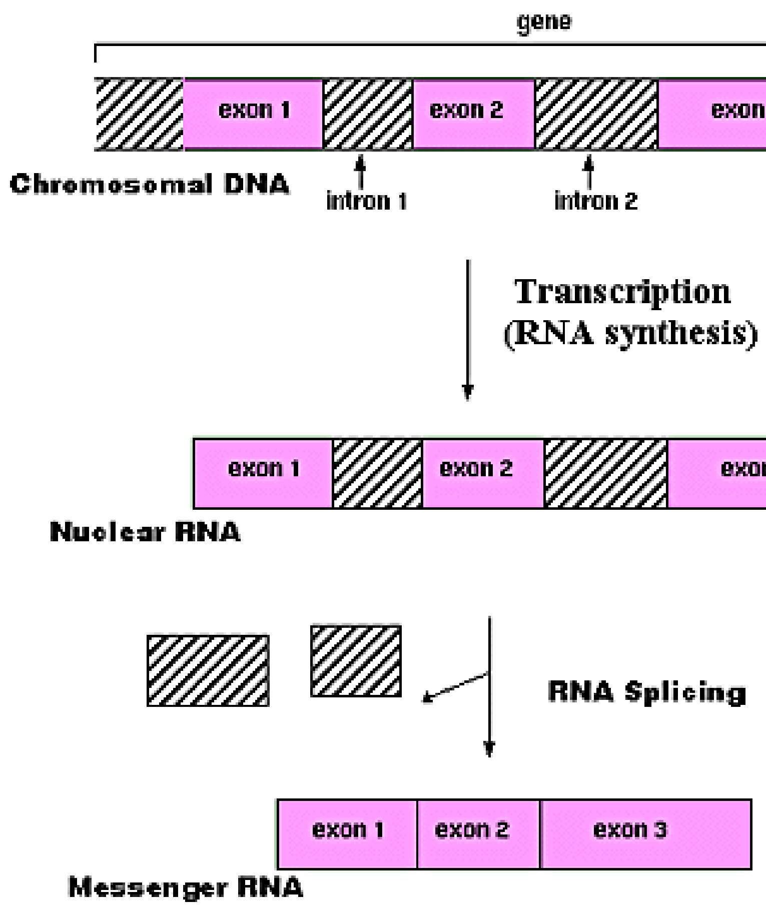
mRNA

```
                 a ugcaaacucu uucugaacgc cucaagaaga ggcgaauugc
guuaaaaaug acgcaaaccg aacuggcaac caaagccggu guuaaacagc aaucaauuca
acugauugaa gcuggaguaa ccaagcgacc gcgcuucuug uuugagauug cuauggcgcu
uaacugugau ccgguuuggu uacaguacgg aacuaaacgc gguaaagccg cuuaa
```

Protein

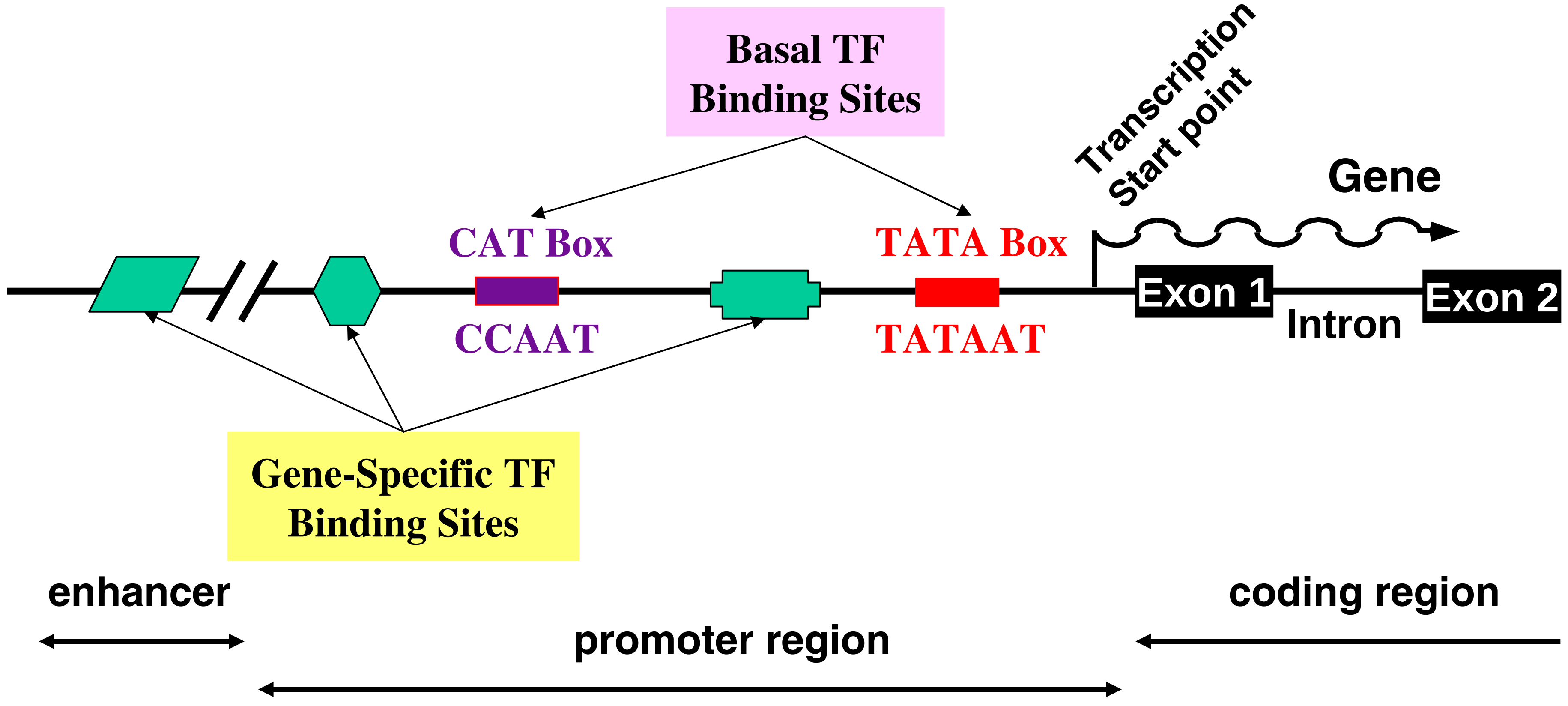
```
 1 mqtlserlkk rrialkmtqt elatkagvkq qsiqlieagv tkrprflfei amalncdpvw
61 lqygtkrgka a
```

DNA Transcription

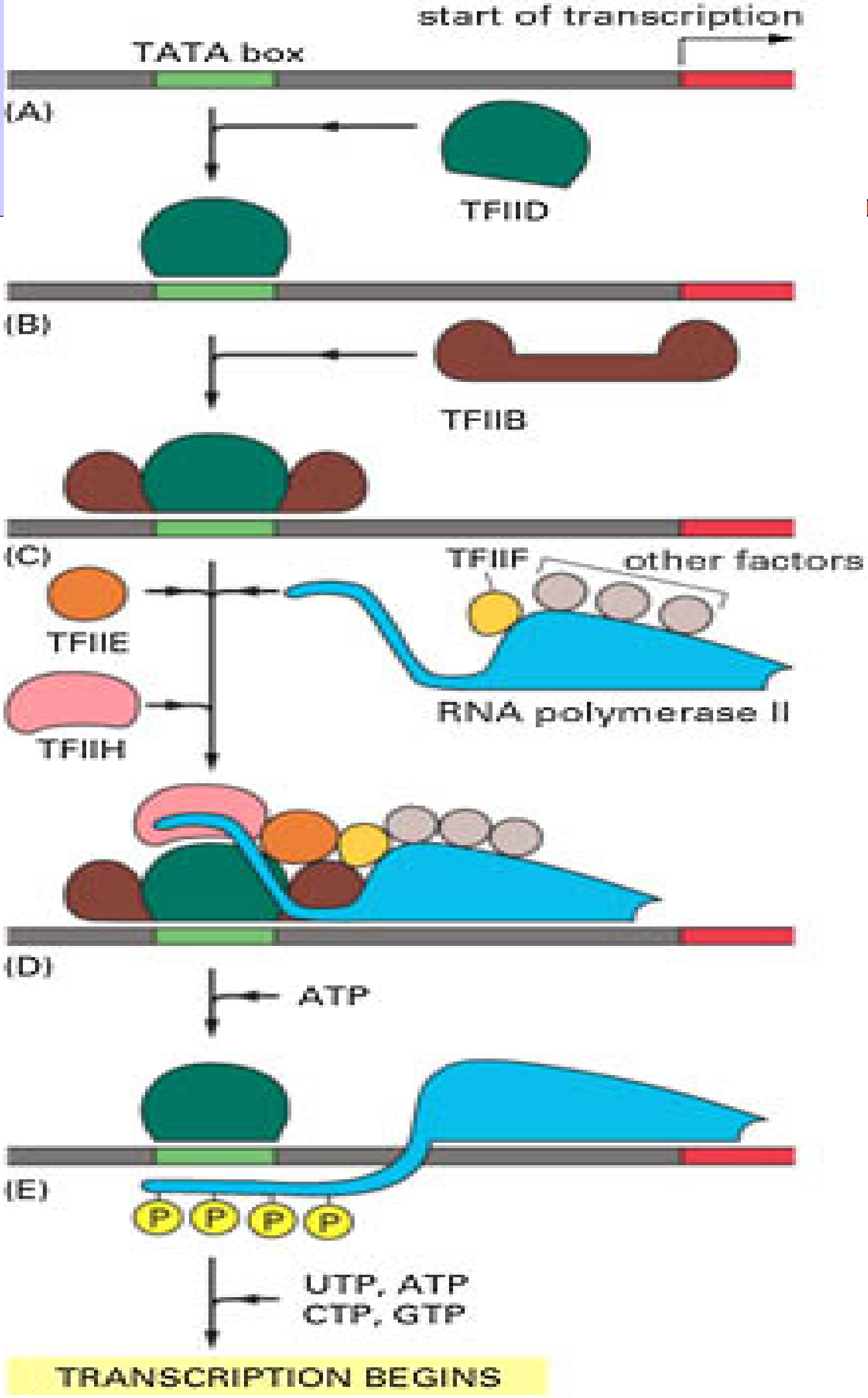


RNA synthesis and processing

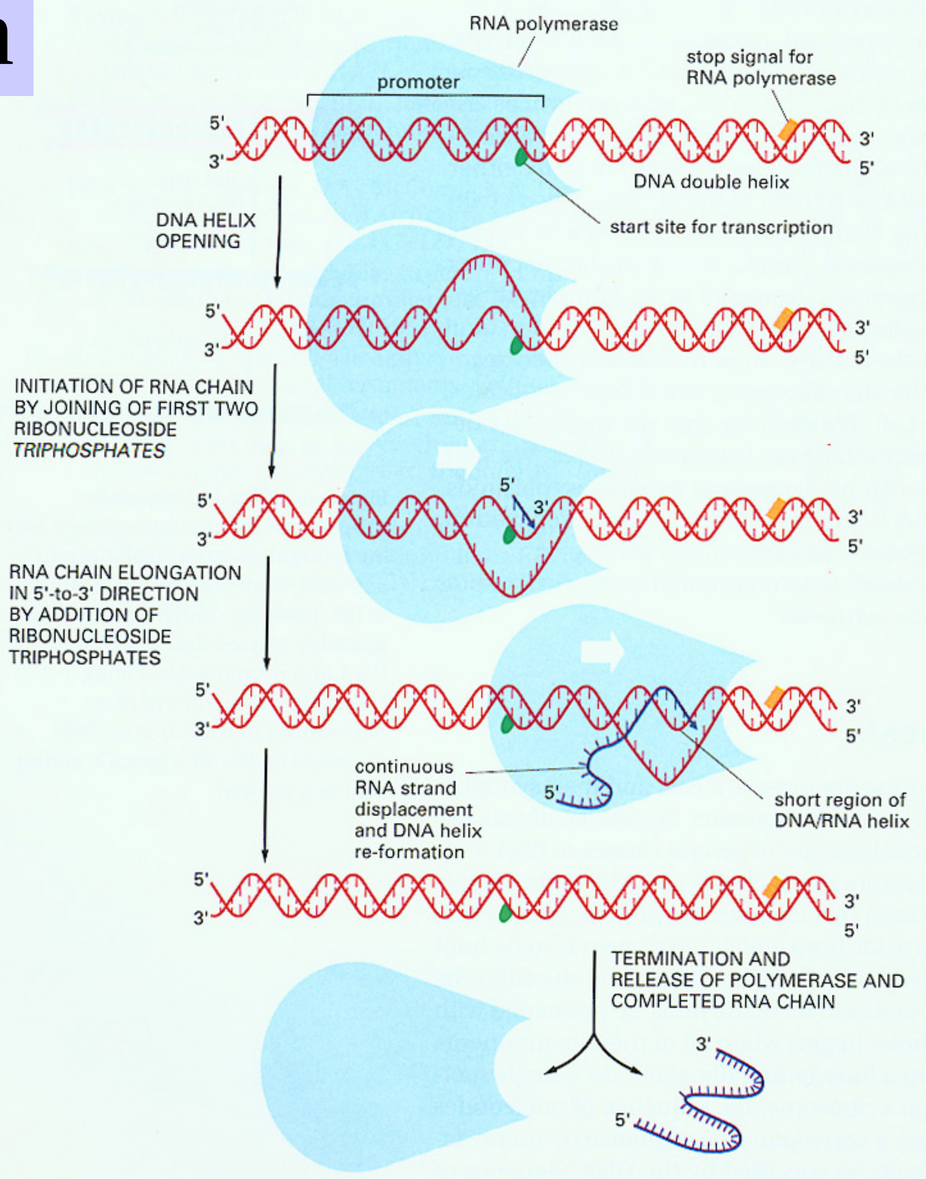
Transcription Regulation



Transcription Initiation



Transcription



RNA polymerase
promoter
stop signal for RNA polymerase
5'
3'
3'
5'
DNA double helix
start site for transcription
DNA HELIX OPENING
INITIATION OF RNA CHAIN BY JOINING OF FIRST TWO RIBONUCLEOSIDE *TRIPHOSPHATES*
RNA CHAIN ELONGATION IN 5'-to-3' DIRECTION BY ADDITION OF RIBONUCLEOSIDE TRIPHOSPHATES
continuous RNA strand displacement and DNA helix re-formation
short region of DNA/RNA helix
TERMINATION AND RELEASE OF POLYMERASE AND COMPLETED RNA CHAIN

Figure 6–2 The synthesis of an RNA molecule by RNA polymerase. The enzyme binds to the promoter sequence on the DNA and begins its synthesis at a start site within the promoter. It completes its synthesis at a stop (termination) signal, whereupon both the polymerase and its completed RNA chain are released. During RNA chain elongation, polymerization rates average about 30 nucleotides per second at 37°C. Therefore, an RNA chain of 5000 nucleotides takes about 3 minutes to complete.

224 **Chapter 6 : Basic Genetic Mechanisms**

Transcription Steps

RNA polymerase needs many transcription factors (TFIIA,TFIIB, etc.)

(A) The promoter sequence (TATA box) is located 25 nucleotides away from transcription initiation site.

(B) The TATA box is recognized and bound by transcription factor TFIID, which then enables the adjacent binding of TFIIB. DNA is somewhat distorted in the process.

(D) The rest of the general transcription factors as well as the RNA polymerase itself assemble at the promoter. What order?

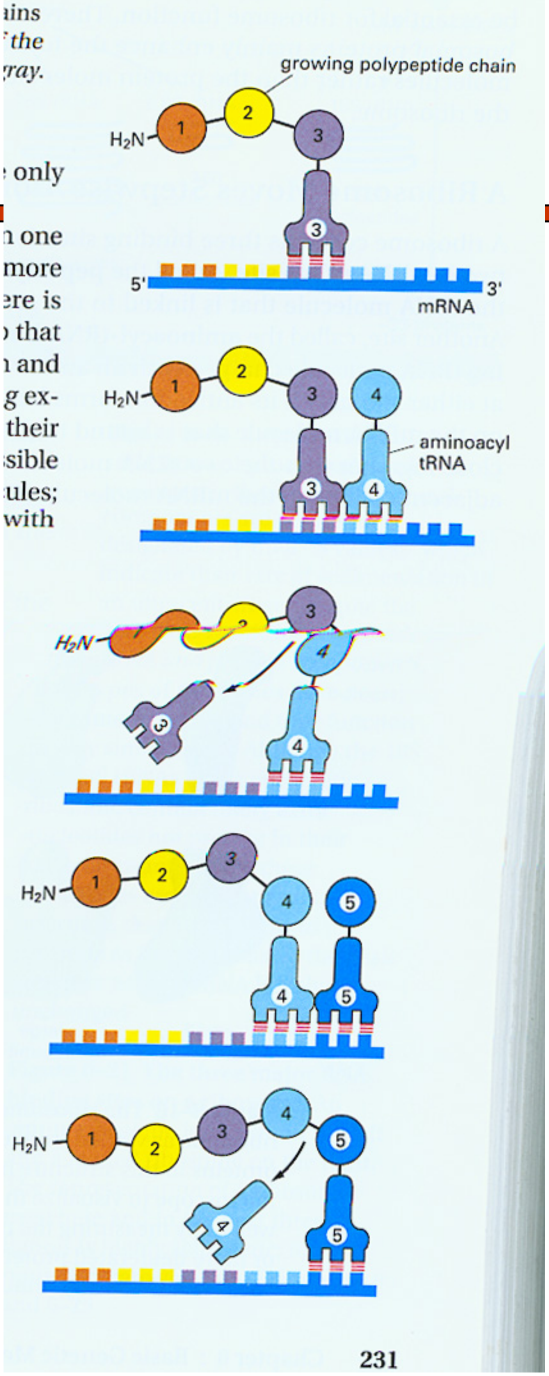
(E) TFIIH then uses ATP to phosphorylate RNA polymerase II, changing its conformation so that the polymerase is released from the complex and is able to start transcribing. As shown, the site of phosphorylation is a long polypeptide tail that extends from the polymerase molecule.

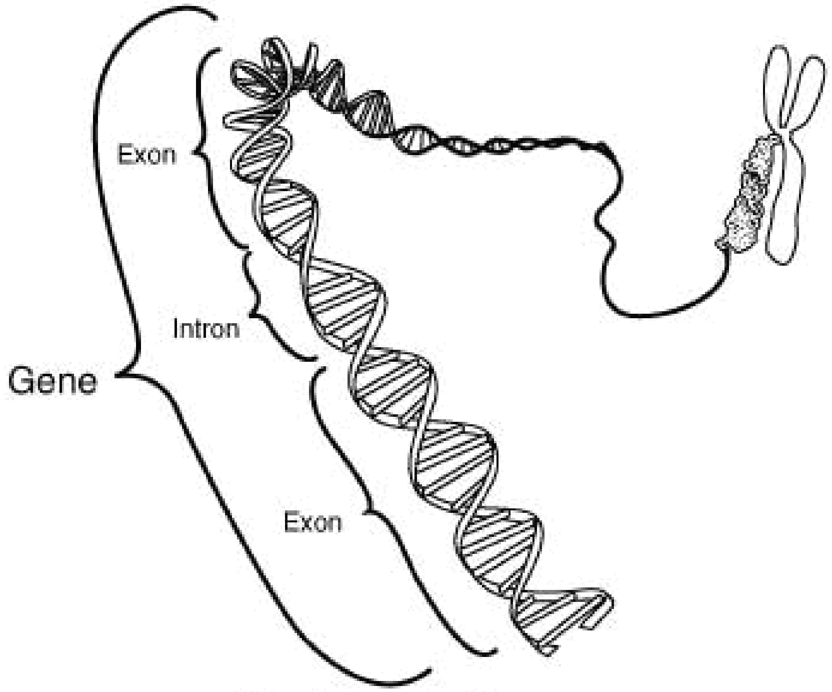
Transcription Factors

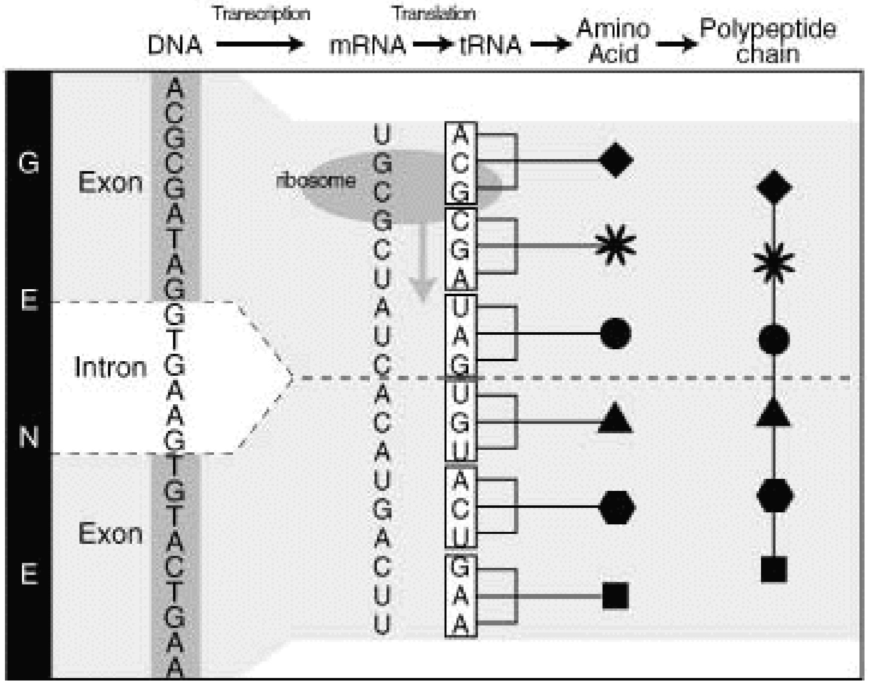
- The general transcription factors have been highly conserved in evolution; some of those from human cells can be replaced in biochemical experiments by the corresponding factors from simple yeasts.

Protein Synthesis:

Incorporation of amino acid into protein




Exon
Intron
Gene
Exon


DNA
Transcription
mRNA
Translation
tRNA
Amino Acid
Polypeptide chain
G E N E
Exon
Intron
Exon
ribosome
ACGCGATAGGTGAAGTGTACTGAA
UGCGCUAUCACAUGACUU
ACG CGA UAG UGU ACU GAA

Perl: Practical Extraction & Report Language

- Created by Larry Wall, early 90s
- Portable, “glue” language for interfacing C/Fortran code, WWW/CGI, graphics, numerical analysis and much more
- Easy to use and extensible
- OOP support, simple databases, simple data structures.
- From interpreted to compiled
- high-level features, and relieves you from manual memory management, segmentation faults, bus errors, most portability problems, etc, etc.
- Competitors: Python, Tcl, Java

Perl Features

- Bit Operations
- Pattern Matching
- Subroutines
- Packages & Modules
- Objects
- Interprocess Communication
- Threads , Process control
- Compiling

Managing a Large Project

- Devise a common data exchange format.
- Use modules that have already been developed.
- Write Perl scripts to convert to and from common data exchange format.
- Write Perl scripts to "glue" it all together.

What is Bioperl?

- Tookit of Perl modules useful for bioinformatics
- Open source; Current version: Bioperl 1.5.2
- Routines for handling biosequence and alignment data.
- Why? Human Genome Project: Same project, same data. different data formats! Different input formats. Different output formats for comparable utility programs.
  - BioPerl was useful to interchange data and meaningfully exchange results. "Perl Saved the Human Genome Project"
- Many routine tasks automated using BioPerl.
- String manipulations (string operations: substring, match, etc.; handling string data: names, annotations, comments, bibliographical references; regular expression operations)
- Modular: modules in any language

Miscellaneous

- pTk – to enable building Perl-driven GUIs for X-Window systems.
- BioJava
- BioPython
- The BioCORBA Project provides an object-oriented, language neutral, platform-independent method for describing and solving bioinformatics problems.

Perl: Examples

```
#!/usr/bin/perl -w
# Storing DNA in a variable, and printing it out

# First we store the DNA in a variable called $DNA
$DNA = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';

# Next, we print the DNA onto the screen
print $DNA;

# Finally, we'll specifically tell the program to exit.
exit;  #perl1.pl
```

Perl: Strings

```
#!/usr/bin/perl -w
$DNA1 = 'ACGGGAGGACGGGAAAATTACTACGGCATTAGC';
$DNA2 = 'ATAGTGCCGTGAGAGTGATGTAGTA';
# Concatenate the DNA fragments
$DNA3 = "$DNA1$DNA2";
print "Concatenation 1):\n\n$DNA3\n\n";
# An alternative way using the "dot operator":
$DNA3 = $DNA1 . $DNA2;
print "Concatenation 2):\n\n$DNA3\n\n";
# transcribe from DNA to RNA; make rev comp; print;
$RNA = $DNA3; $RNA =~ s/T/U/g;
$rev = reverse $DNA3; $rev =~ tr/AGCTacgt/TCGAtgca/;
print "$RNA\n$rev\n";
exit;  #perl2.pl
```

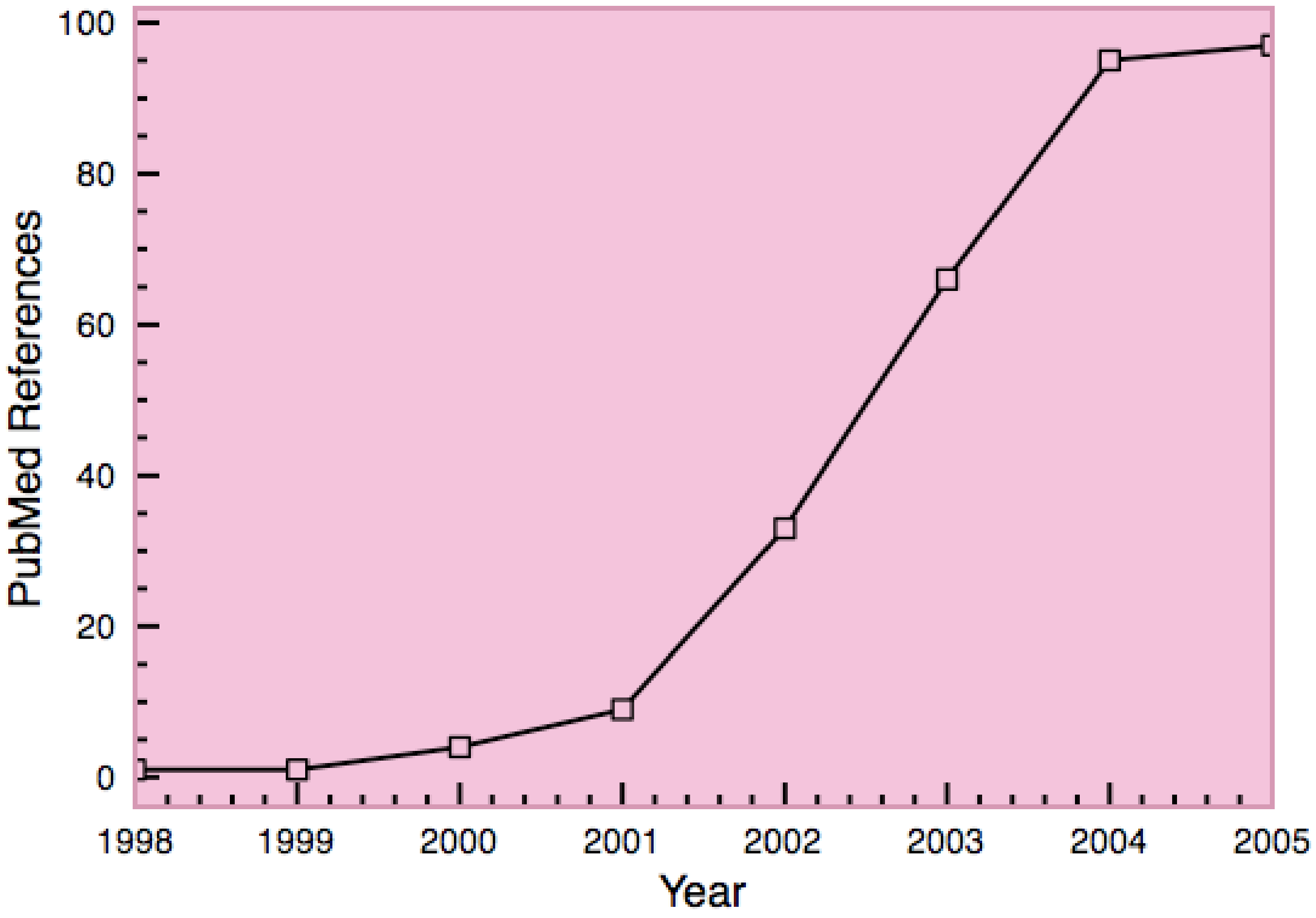
Perl: arrays

```
#!/usr/bin/perl -w
# Read filename & remove newline from string
$protFile = <STDIN>; chomp $protFile;
# First we have to "open" the file
unless (open(PROTEINFILE, $protFile) {
  print "File $protFile does not exist"; exit;}
# Each line becomes an element of array @protein
@protein = <PROTEINFILE>;
print @protein;
# Print line #3 and number of lines
print $protein[2], "File contained ", scalar @protein, "
  lines\n";
# Close the file.
close PROTEINFILE;
exit; #perl3.pl
```

Perl: subroutines

```
#!/usr/bin/perl -w
# using command line argument
$dna1 = $ARGV[0]; $dna2 = $ARGV[1];
# Call subroutine with arguments; result in $dna
$dna = addGACAGT($dna1, $dna2);
print "Add GACAGT to $dna1 & $dna2 to get $dna\n\n";
exit;
##### addACGT: concat $dna1, $dna2, & "ACGT". #####
sub addGACAGT {
    my($dnaA, $dnaB) = @_; my($dnaC) = $dnaA.$dnaB;
    $dnaC .= 'GACAGT';
    return $dnaC;
} #perl4.pl
```

How Widely is Bioperl Used?



What Can You Do with Bioperl?

- Accessing sequence data from local and remote databases
- Transforming formats of database/ file records
- Manipulating individual sequences
- Searching for similar sequences
- Creating and manipulating sequence alignments
- Searching for genes and other structures on genomic DNA
- Developing machine readable sequence annotations

Types of Perl Objects:

- Sequence objects
- Location objects
- Interface objects
- Implementation objects

BioPerl Modules

- **Bio::PreSeq**, module for reading, accessing, manipulating, analyzing single sequences.
- **Bio::UnivAln**, module for reading, parsing, writing, slicing, and manipulating multiple biosequences (sequence multisets & alignments).
- **Bio::Struct**, module for reading, writing, accessing, and manipulating 3D structures.
- Support for invoking **BLAST** & other programs.
- Download URL [http://www.bioperl.org/Core/Latest/]
- Tutorial [http://www.bioperl.org/Core/Latest/bptutorial.html]
- Course [http://www.pasteur.fr/recherche/unites/sis/formation/bioperl/index.html]

BioPerl Sequence Object

```
$seqobj->display_id(); # readable id of sequence
$seqobj->seq(); # string of sequence
$seqobj->subseq(5,10); # part of the sequence as a string
$seqobj->accession_number(); # if present, accession num
$seqobj->moltype(); # one of 'dna','rna','protein'
$seqobj->primary_id(); # unique id for sequence independent
                       # of its display_id or accession number
```

Example 1: Convert SwissProt to fasta format

```
#! /local/bin/perl -w

use strict;
use Bio::SeqIO;
my $in  = Bio::SeqIO->newFh ( -file   => '<seqs.html',
                              -format => 'swiss' );
my $out = Bio::SeqIO->newFh ( -file   => '>seqs.fasta',
                              -format => 'fasta' );

print $out $_ while <$in>;

exit; #bioperl1.pl
```

Example 2 : Load sequence from remote server

```
#!/usr/bin/perl -w
use Bio::DB::SwissProt;

$database = new Bio::DB::SwissProt;

$seq = $database->get_Seq_by_id('MALK_ECOLI');

my $out = Bio::SeqIO->newFh(-fh => STDOUT,
            -format => 'fasta');

print $out $seq;

exit;
```

```
#!/local/bin/perl -w

use Bio::DB::GenBank;

my $gb =
    new Bio::DB::GenBank(
    -retrievaltype=>'tempfile',
    -format=>'Fasta');

my ($seq) = $seq =
    $gb->get_Seq_by_id("5802612");
print $seq->id, "\n";
print $seq->desc(), "Sequence: \n";
print $seq->seq(), "\n";
exit;
```

Sequence Formats in BioPerl

```
#! /local/bin/perl -w
use strict;
use Bio::SeqIO;
my $in  = Bio::SeqIO->new ( -file   => 'seqs.html', -format => 'swiss' );
my $out = Bio::SeqIO->new ( -file   => 'seqs.fas', -format => 'fasta' );

while ($seq = $in->next_seq()) {
    $accNum = $seq->accession_number();
    print “Accession# = $accNum\n”;
    $out->write_seq($seq);
}

exit; #bioperl2.pl
```

BioPerl

```
#!/usr/bin/perl -w
# define a DNA sequence object with given sequence
$seq = Bio::Seq->new('-seq'=>'actgtggcgtcaact',
   '-desc'=>'Sample Bio::Seq object',
   '-display_id' => 'somethingxxx',
   '-accession_number' => 'accnumxxx',
   '-alphabet' => 'dna' );
$gb = new Bio::DB::GenBank();

$seq = $gb->get_Seq_by_id('MUSIGHBA1'); #returns Seq object
$seq = $gb->get_Seq_by_acc('AF303112'); #returns Seq object
# this returns a SeqIO object :
$seqio = $gb->get_Stream_by_batch([ qw(J00522 AF303112)]));
exit; #bioperl3.pl
```

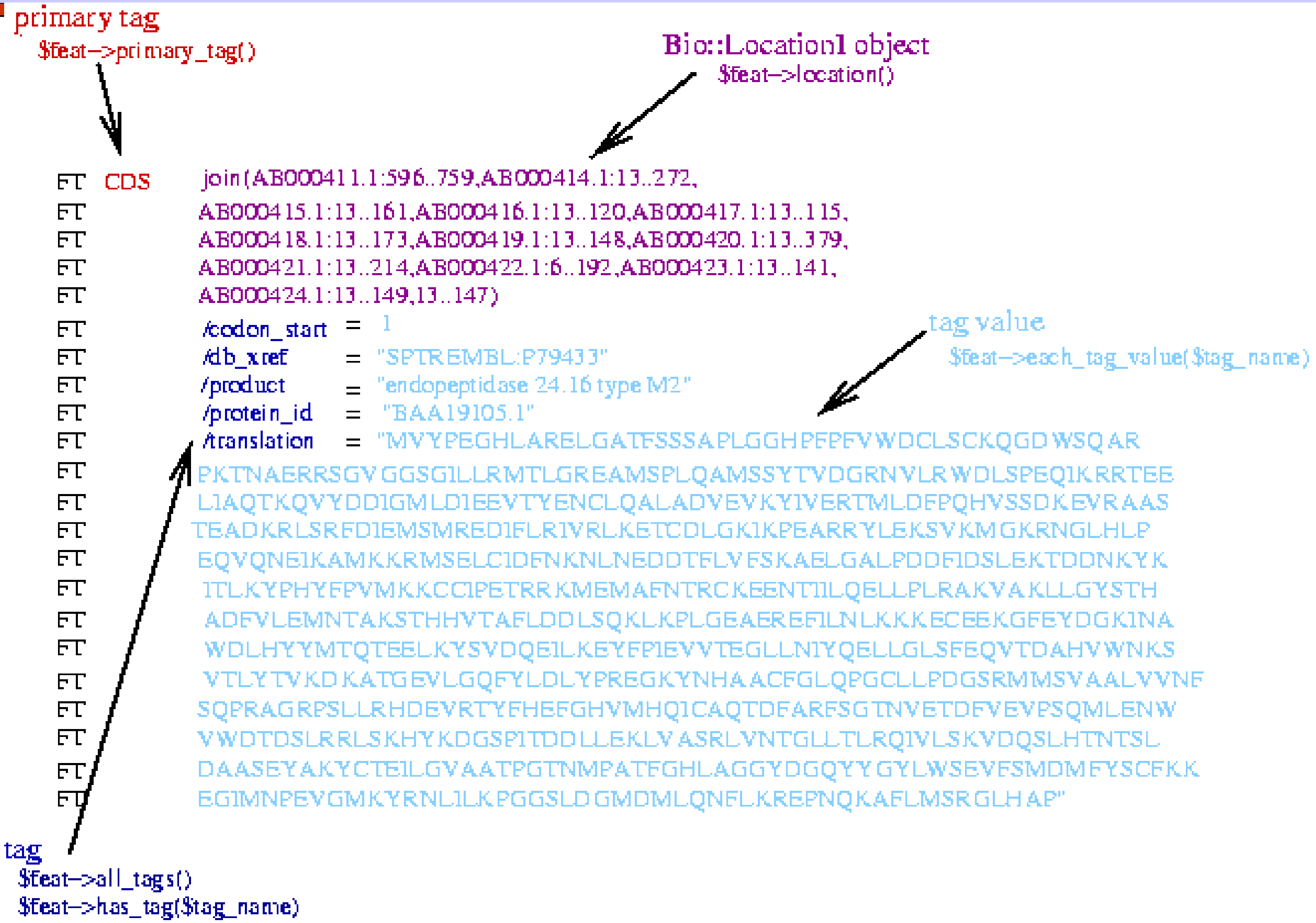
Sequence Manipulations

```
#!/local/bin/perl -w

use Bio::DB::GenBank;

$gb = new Bio::DB::GenBank();

$seq1 = $gb->get_Seq_by_acc('AF303112');
$seq2=$seq1->trunc(1,90);
$seq2 = $seq2->revcom();

print $seq2->seq(), "\n";
$seq3=$seq2->translate;
print $seq3->seq(), "\n";
exit; #bioperl4.pl
```

BioPerl: Structure

```
use Bio::Structure::IO;
$in = Bio::Structure::IO->new(-file => "inputfilename" , '-
   format' => 'pdb');
$out = Bio::Structure::IO->new(-file => ">outputfilename" , '-
   format' => 'pdb');
# note: we quote -format to keep older perl's from
   complaining.
while ( my $struc = $in->next_structure() ) {
   $out->write_structure($struc);
   print "Structure ",$struc->id," number of models: ",
        scalar $struc->model,"\n";
}
```

Sequence Features



primary tag
$feat->primary_tag()

Bio::Location1 object
$feat->location()

tag value
$feat->each_tag_value($tag_name)

```
FT   CDS      join(AB000411.1:596..759,AB000414.1:13..272,
FT            AB000415.1:13..161,AB000416.1:13..120,AB000417.1:13..115,
FT            AB000418.1:13..173,AB000419.1:13..148,AB000420.1:13..379,
FT            AB000421.1:13..214,AB000422.1:6..192,AB000423.1:13..141,
FT            AB000424.1:13..149,13..147)
FT            /codon_start  =  1
FT            /db_xref      =  "SPTREMBL:P79433"
FT            /product      =  "endopeptidase 24.16 type M2"
FT            /protein_id   =  "BAA19105.1"
FT            /translation  =  "MVYPEGHLARELGATFSSSAPLGGHPFPFVWDCLSCKQGDWSQAR
FT            PKTNAERRSGVGGSGILLRMTLGREAMSPLQAMSSYTVDGRNVLRWDLSPEQIKRRTEE
FT            LIAQTKQVYDDIGMLDIEEVTYENCLQALADVEVKYIVERTMLDFPQHVSSDKEVRAAS
FT            TEADKRLSRFDIEMSMREDIFLRIVRLKETCDLGKIKPEARRYLEKSVKMGKRNGLHLP
FT            EQVQNEIKAMKKRMSELCIDFNKNLNEDDTFLVFSKAELGALPDDFIDSLEKTDDNKYK
FT            ITLKYPHYFPVMKKCCIPETRRKMEMAFNTRCKEENTIILQELLPLRAKVAKLLGYSTH
FT            ADFVLEMNTAKSTHHVTAFLDDLSQKLKPLGEAEREFILNLKKKECEEKGFEYDGKINA
FT            WDLHYYMTQTEELKYSVDQEILKEYFPIEVVTEGLLNIYQELLGLSFEQVTDAHVWNKS
FT            VTLYTVKDKATGEVLGQFYLDLYPREGKYNHAACFGLQPGCLLPDGSRMMSVAALVVNF
FT            SQPRAGRPSLLRHDEVRTYFHEFGHVMHQICAQTDFARFSGTNVETDFVEVPSQMLENW
FT            VWDTDSLRRLSKHYKDGSPITDDLLEKLVASRLVNTGLLTLRQIVLSKVDQSLHTNTSL
FT            DAASEYAKYCTEILGVAATPGTNMPATFGHLAGGYDGQYYGYLWSEVFSMDMFYSCFKK
FT            EGIMNPEVGMKYRNLILKPGGSLDGMDMLQNFLKREPNQKAFLMSRGLHAP"
```

tag
$feat->all_tags()
$feat->has_tag($tag_name)

BioPerl: Seq and SeqIO

```
use Bio::SeqIO;
$seqin = Bio::SeqIO->new(-format =>'EMBL', -file=>‘f1');
$seqout= Bio::SeqIO->new(-format =>'Fasta',-file=>'>f1.fa');
while ((my $seqobj = $seqin->next_seq())) {
    print "Seq: ", $seqobj->display_id, ", Start of seq ",
            substr($seqobj->seq,1,10),"\n";
    if ( $seqobj->moltype eq 'dna') {
            $rev = $seqobj->revcom;
            $id = $seqobj->display_id();
            $id = "$id.rev";
            $rev->display_id($id);
            $seqout->write_seq($rev); } #end if
    foreach $feat ( $seqobj->top_SeqFeatures() ) {
            if( $feat->primary_tag eq 'exon' ) {
               print STDOUT "Location ",$feat->start,":",
               $feat->end," GFF[",$feat->gff_string,"]\n";}
    } # end foreach
} # end while
exit; #bioperl6.pl
```

Example 3: Read alignment file using AlignIO class

```
#!/usr/bin/perl -w
use strict;
use Bio::AlignIO;
my $in = new Bio::AlignIO(-file => '<data/infile.aln',
                          -format => 'clustalw');

# returns an alignI (alignment interface)
my $aln = $in->next_aln();
print "same length of all sequences: ",
($aln->is_flush()) ? "yes" : "no", "\n";
print "alignment length: ", $aln->length, "\n";
printf "identity: %.2f %%\n", $aln->percentage_identity();
printf "identity of conserved columns: %.2f %%\n",
$aln->overall_percentage_identity();
```

Example 4: Standalone BLAST

```
#!/usr/bin/perl -w
use strict;
use Bio::SeqIO;
use Bio::Tools::Run::StandAloneBlast;
my $seq_in = Bio::SeqIO -> new(-file => '<data/prot1.fasta',
                              -format => 'fasta');
my $query = $seq_in -> next_seq();
my $factory = Bio::Tools::Run::StandAloneBlast -> new('program' => 'blastp',
                                                     'database' => 'swissprot',
                                                     _READMETHOD => 'Blast');
my $blast_report = $factory->blastall($query);
my $result = $blast_report->next_result();
while (my $hit = $result->next_hit()){
  print "\thit name: ", $hit->name(), "Significance: ", $hit->significance(), "\n";
  while (my $hsp = $hit->next_hsp()){
    print "E: ", $hsp->evalue(), "frac_identical: ", $hsp->frac_identical(), "\n";
  }

}
exit;
```