UNIVERSITY HONORS PROGRAM

SENIOR PROJECT - APPROVAL

Name: Jon Scharff

College: Arts & Sciences Department: Computer Science

Faculty Mentor: Dr. Michael Langston

PROJECT TITLE: Motif-finding and Other Applications in Bioinformatics

I have reviewed this completed senior honors thesis with this student and certify that it is a project commensurate with honors level undergraduate research in this field.

Signed: _____, Faculty Mentor

Date: 9 DEC 03

Comments (Optional):

Summary of Work Performed on Senior Project

Jon Scharff

Motif-finding and Other Applications in Bioinformatics

The following steps have been completed on this project:

Rewrite Dr. Leuze's algorithm for motif finding
This step was achieved during the summer. It allowed me to work on something that did not require much biology knowledge while I began to research biological concepts. During this time, I also listened in to "A Short Course on Bioinformatics" offered by Dr. Jay Snoddy over the summer to familiarize myself with some of the tools and databases available for use. Upon completion of this project, I had successfully improved upon Dr. Leuze's algorithm and had become familiar with some of the bioinformatical concepts necessary to complete the next steps in the project. More information on the results of this step can be found at http://www.cs.utk.edu/~scharff/sr/summary.html.

Write an interface to facilitate viewing mouse and human gene orthologs (Version 1)
In version 1, I had no direct access to any databases at ORNL, so I had to improvise some way of gathering data aside from that provided to me. To start with, I was given two files, one with mouse genes and one with mouse and human orthologs. The mouse gene file contains identifiers of genes for which ORNL currently has expression data. The ortholog file includes information extracted from the DOE's Celera database. Using this information, I created a script to query the LocusLink database via an HTTP connection and to parse out gene identifiers so that I could link the information in the two files. I then wr[illegible] CGI script that takes a gene identifier and shows the gene(s) that match the query as well as their orthologs. During this time, I was also taking a class in bioinformatics taught by Dr. La[illegible]gston that further increased my understanding of some of the needs of the biology community [illegible] bioinformatics is able to fulfill. The CGI script can currently be accessed at http://bio.cokam[illegible].net/cgi-bin/orthoold.pl.

The following steps are future [illegible] of the project

Write an interface that facilitates viewing relative gene locations (Version 2)
This step is in progress. While I still have no direct access to ORNL databases, I was able to find a file that contained most of the data provided in the LocusLink database, enabling me to get much more information on each gene than I was able to parse out of the web pages in the above step. I have already determined what the majority of the data in the file correspond to, have created a preliminary database structure to hold what I feel is the relevant data from the file, and [illegible] the data from the file to the database. Upon completion of the script, I will write a CGI script to perform the same function as version 1 above with the new database structure to replace the version 1 script. Then I will work on CGI scripts to allow for searching by location, viewing groups of genes, and seeing if there is a group of genes on a mouse chromosome with a similar group of genes of orthologs on a human chromosome.

This project is a preliminary step in determining what mouse genes should be further studied in order to learn more about human genes. By finding the mouse orthologs of human genes, we can then study the mouse genes using techniques that cannot be ethically used on humans and infer information about the corresponding human gene. This project is the opening steps in displaying what information is currently available on human and mouse genes, as well as what mouse genes are likely orthologs of certain human genes. While the project currently only displays the possible orthologs based on data provided to us from the Celera database and based on the gene symbols, the final project is envisioned to contain more sources of possible orthologs as well as the ability to visually see the orthologous genes and their surrounding genes to find groupings that are similar in both human and mouse. Also, using the code from this summer to find similar sequences around genes, we can see if there are similar motifs around the mouse-human orthologs identified. But this preliminary step is what is necessary so that we can determine what mouse genes we are interested in so that we can determine on what genes we can get information and for what genes we need to prepare microarrays to gather data.

The code in parseMoulib.pl and parseOrtho.pl contain my method for getting data from the moulib database and Celera database files, respectively. The moulib file tells us what mouse genes we can currently get data on, and the Celera file tells us some probable mouse orthologs of human genes that we are interested in studying further.

The code in pump.pl gets data necessary to link the information in the Celera database with the information in the moulib database. Since both databases use different identifiers for genes, we need to be able to link this data. Pump.pl requests an html page from the LocusLink web interface and parses the html to find the relevant identifiers for the gene.

The code in orthoold.pl uses the data from the Celera and moulib databases and the LocusLink web interface to present genes, their identifiers, and their orthologs in what is intended to be a readable fashion.

The code in llCron.pl is intended to replace pump.pl as the method for getting data from the LocusLink database. While the previous method got the necessary data for determining for what mouse genes we have microarrays and for which mouse genes we need to prepare microarrays, it was slow, and not much more information could easily be gathered from the html pages. The LocusLink data file that I found contains more comprehensive data on genes including the identifiers that pump.pl gathered and location data that will be useful in future versions of the project. LlCron.pl is also able to gather data more quickly as it does not rely on an Internet connection once the data file has been downloaded.

```
#!/usr/bin/perl
# llCron.pl- script to download the LocusLink database file and parse out
#  relevant information
use Net::FTP;
use Compress::Zlib;
use DBI;

my $dbh = DBI->connect('DBI:mysql:bio', 'biouser', 'bio') or
  die('could not connect to database');

my $llserv = 'ftp.ncbi.nih.gov';
#my @files = ('/refseq/LocusLink/LL.out.gz',
#my @files = ('/refseq/LocusLink/LL.out_hs.gz',
#             '/refseq/LocusLink/LL.out_mm.gz',
my @files = (
             '/refseq/LocusLink/LL_tmpl.gz');
#my @localFiles = ('LL.out',
#my @localFiles = ('LL.out_hs', 'LL.out_mm',
my @localFiles = (
                  'LL_tmpl');

downloadFiles($llserv, \@files, \@localFiles);
#populateLlout(@localFiles[0,1]);
pumpLltmpl($localFiles[0]);

$dbh->disconnect;

sub populateLlout(@) {
   my $start = 'INSERT INTO llout (llid, chromosome, cyto, geneName, '.
               'organism, symbol, symType) VALUES ';
   my ($query, $cnt);
   my $prevLlid;

   $dbh->do('DELETE FROM llout');
   foreach my $fn (@_) {
      open(FILE, $fn);
      $query = $start;
      $cnt = 0;
      while(<FILE>) {
         chomp;
         my @row = split('\t', $_);
         my ($org, $sym, $symt);

         next if($row[0] == $prevLlid || $row[0] =~ /\D/);
         $prevLlid = $row[0];

         delete $row[4] if($row[4] eq '');
         delete $row[5] if($row[5] eq '');
         delete $row[6] if($row[6] eq '');

         if($row[7] == 9606) {
            $org = 'human';
            next unless($row[4] =~ /^(?:5|1[69]|)$/);
         } elsif($row[7] == 10090) {
            $org = 'mouse';
         } else {
            next;
         }

         if($row[1] ne '') {
            $sym = $row[1];
```

```
while(<FILE>) {
   chomp;
   if(/^>>\d+$/) {
      $next = 0;
      undef %assmAccn;
   }
   next if($next);

   $llid = $1 if(/^LOCUSID: (\d+)$/);
   if(/^CURRENT_LOCUSID: (\d+)$/) {
      $genellcheck->execute($1);
      if(my ($geneId) = $genellcheck->fetchrow_array) {
         $dbh->do('INSERT INTO genell (geneId, llid, status) VALUES '.
                  "('$geneId', '$llid', 'alias')");
      } else {
         push(@{$llAlias{$1}}, $llid);
      }
      $genellcheck->finish;
      $next++;
      next;
   }
   $llStatus = ($1 eq 'yes')?('confirmed'):('unconfirmed') and next
     if(/^LOCUS_CONFIRMED: (\w+)$/);
   if(/^ORGANISM: ([\w ]+)$/) {
      if($1 eq 'Mus musculus' || $1 eq 'Homo sapiens') {
         my $org = ($1 eq 'Mus musculus')?('mouse'):('human');
         $gene->execute($org);
         $genell->execute($llid, $llStatus);
         for my $alias (@{$llAlias{$llid}}) {
            $genell->execute($alias, 'alias');
         }
         delete $llAlias{$llid};
      } else {
         $next++;
      }
      next;
   }

   $geneStatus = $1 and next if(/^STATUS: (\w+)$/);
   if(/^(N\w|ACCNUM): ([^|]+)\|(\d+)\|([^|]+)(?:\|(\d+|na)\|(\d+|na))?$/) {
      my $type;
      if($1 eq 'NG') {
         $type = 'genomic';
      } elsif($1 eq 'NR') {
         $type = 'RNA';
      } elsif($1 eq 'NM') {
         $type = 'mRNA';
      } elsif($1 eq 'ACCNUM') {
         $assmAccn{'accn'} = $2;
         $assmAccn{'gi'} = $3;
         $assmAccn{'strain'} = ($4 ne 'na')?($4):(undef);
         $assmAccn{'begin'} = ($5 ne 'na')?($5):(undef);
         $assmAccn{'end'} = ($6 ne 'na')?($6):(undef);
         next;
      } else {
         next;
      }
      $accncheck->execute($3);
      if(my ($curAccn, $curVer, $curType) = $accncheck->fetchrow_array) {
         if($curAccn ne $2 || $curType ne $type) {
            print STDERR "Ambiguous gi $3: ($2, $type) and ".
```

```
#          print "skipping $files[$i]\n";
           $mod[$i] = 0;
           next;
        }
        $ftp->get($files[$i], "$lf.gz");
        gunzip("$lf.gz", $lf);
        $mod[$i] = 1;
    }
    $ftp->quit;
}

sub gunzip($$) {
    my ($ifn, $ofn) = @_;
    my $gz = gzopen($ifn, "rb") or return 0;
    my $ln;
    open(OUTFILE, ">$ofn") or return 0;

    while($gz->gzread($ln) > 0) {
        print OUTFILE $ln;
    }

    $gz->gzclose();
    close(OUTFILE);
    1;
}
```

```
#!/usr/bin/perl
# parseMoulib.pl- script to parse the information from the moulib database

use DBI;

my $dbh = DBI->connect('DBI:mysql:bio', 'biouser', 'bio') or
  die('could not connect to database');

$_ = <>;
chomp;
my @header = split(/\s+/, $_);

my $start = 'INSERT INTO moulib ('.join(', ', @header).') VALUES ';
my $sel = $start;
my $cnt = 0;

while(<>) {
   chomp;
   my @vals = split(/\s+/, $_);
   next if(scalar(@vals) != scalar(@header));
   $sel.= '('.join(', ', map { $dbh->quote($_); } @vals).'),';
   $cnt++;
   if($cnt >= 1000) {
      chop($sel);
      $dbh->do($sel);
      $sel = $start;
      $cnt = 0;
   }
}
$dbh->do($sel) if(chop($sel) eq ',');
```

```
#!/usr/bin/perl
# parseOrtho.pl: script to parse the information received form the celera
#  database

use DBI;

my $dbh = DBI->connect('DBI:mysql:bio', 'biouser', 'bio') or
  die('could not connect to database');

$_ = <>;
chomp;
my @header = split("\t", $_);

my $startci = 'INSERT INTO celerainfo (ct, symbol, organism, chromosome, '.
              'orientation, begin, end) VALUES ';
my $startco = 'INSERT INTO celeraortho (human, mouse) VALUES ';
my $sel1 = $startci;
my $sel2 = $startco;
my $cnt = 0;

while(<>) {
   chomp;
   my @val = split("\t", $_);
   die("invalid row") if(scalar(@val) != scalar(@header);
   my %vals = map { $header[$_] => $val[$_] } (0..$#header);

   my @human = ($vals{'Human CT'}, $vals{'Human Symbol'}, 'human',
                $vals{'Human Chromosome'}, $vals{'Human Orientation'},
                $vals{'Human Begin'}, $vals{'Human End'});
   my @mouse = ($vals{'Mouse CT'}, $vals{'Mouse Symbol'}, 'mouse',
                $vals{'Mouse Chromosome'}, $vals{'Mouse Orientation'},
                $vals{'Mouse Begin'}, $vals{'Mouse End'});
   $sel1.= '('.join(', ', map { $dbh->quote($_); } @human).'),';
   $sel1.= '('.join(', ', map { $dbh->quote($_); } @mouse).'),';
   $sel2.= '('.$dbh->quote($human[0]).', '.$dbh->quote($mouse[0]).'),';
   $cnt++;
   if($cnt >= 1000) {
      chop($sel1);
      chop($sel2);
      $dbh->do($sel1);
      $sel1 = $startci;
      $dbh->do($sel2);
      $sel2 = $startco;
      $cnt = 0;
   }
}
$dbh->do($sel) if(chop($sel) eq ',');
```

```
#!/usr/bin/perl
# pump.pl script to request infromation about genes from the LocusLink
#  database and parse the resulting html page for relevant information.

use DBI;
use HTTP::Request;
use HTTP::Response;
use LWP::UserAgent;
use HTML::Parser;

open(LOG, '>pumpout');
my %organisms = ('Mm' => 'mouse', 'Hs' => 'human');

my $dbh = DBI->connect('dbi:mysql:bio', 'biouser', 'bio') or
  die("could not connect to database $!\n");

my $ua = LWP::UserAgent->new;

my @selectreq = (<<'ENDSQL3', <<'ENDSQL1', <<'ENDSQL2');
SELECT c.symbol, c.organism
FROM celerainfo c
WHERE c.symbol IS NOT NULL AND
      c.symbol NOT IN(SELECT gs.symbol
                      FROM genesym gs JOIN gene g ON gs.geneId = g.id
                      WHERE g.organism = c.organism
                     )
ENDSQL3

SELECT GenBankAccn, 'mouse'
FROM moulib
WHERE GenBankAccn NOT IN(SELECT accn FROM geneaccn)
ENDSQL1

SELECT UniGeneId, 'mouse'
FROM moulib
WHERE UniGeneId IS NOT NULL AND UniGeneId NOT IN (SELECT UniGeneId FROM gene)
ENDSQL2

my @selectopt = (' AND symbol NOT IN(\'', ' AND GenBankAccn NOT IN(\'',
  ' AND UniGeneId NOT IN(\'');
my $cnt = 0;

while($cnt < scalar(@selectreq)) {
   my ($query, $org, $prevQuery, @ignores);
   while(sleep(19)) {
      my $qh = $dbh->prepare($selectreq[$cnt].
        ((scalar(@ignores) > 0)?($selectopt[$cnt].
          join("','", @ignores)."')"):('')).
        ' LIMIT 1');
      $qh->execute;
      if(!(($query, $org) = $qh->fetchrow_array)) {
         syswrite LOG, "round $cnt ignored\n".join(', ', @ignores)."\n\n";
         $cnt++;
         last;
      } elsif($query eq $prevQuery) {
         push @ignores, $query;
         syswrite LOG, "problem $query $org\n";
         next;
      } else {
         $prevQuery = $query;
      }
```

```
      $hp->handler('end' => sub {
         if(defined $llid && $_[0] eq 'b') {
            $inb-- if($inb > 0);
         } elsif(defined $llid && $_[0] eq 'em') {
            $inem-- if($inem > 0);
         } elsif($_[0] eq 'td') {
            $intd-- if($intd > 0);
         }
      }, 'tagname');
      $hp->report_tags(qw(input img b em td));
      $hp->parse($resp->content);

      if(defined $llid) {
         pump($llid, $unigene, $organism, $symbol, \@altsym,
              \@refseq, \@accn);
      } else {
         push @ignores, $query;
         syswrite LOG, "ignoring $query\n";
         next;
      }
   }
}

$dbh->disconnect;
close(LOG);

sub pump($$$$\@\@\@) {
   my $llid  = $_[0];
   my $unigene = $_[1];
   my $organism = $_[2];
   my $symbol = $_[3];
   my @altsym = @{$_[4]};
   my @refseq = @{$_[5]};
   my @accn = @{$_[6]};

   my ($recs) = $dbh->selectrow_array(<<'ENDSQL', undef, $llid);
SELECT count(*) FROM gene WHERE LocusLinkId = ?
ENDSQL

   if($recs > 0) {
      return 0;
      next;
   }

   $dbh->do(<<'ENDSQL', undef, $llid, $unigene, $organism);
INSERT INTO gene (locusLinkId, UniGeneId, organism)
VALUES (?, ?, ?)
ENDSQL

   $dbh->do("INSERT INTO geneaccn (geneId, accn) VALUES (LAST_INSERT_ID(), '".
            join("'), (LAST_INSERT_ID(), '", @accn)."')")
     if(@accn);
   $dbh->do('INSERT INTO geneaccn (geneId, accn, refseq) VALUES '.
            "(LAST_INSERT_ID(), '".
            join("', 'true'), (LAST_INSERT_ID(), '", @refseq).
            "', 'true')")
     if(@refseq);
   $dbh->do('INSERT INTO genesym (geneId, symbol, alias) VALUES '.
            "(LAST_INSERT_ID(), '".
            join("', 'true'), (LAST_INSERT_ID(), '", @altsym).
            "', 'true')")
```

```
#!/usr/bin/perl
# orthoold.pl- CGI script to allow for searching for genes based on an
#  identifier and presenting that gene with its orthologs
use CGI qw(:cgi);
use DBI;
use Bio::DB::GenBank;
use Bio::Seq;
use HTTP::Request;
use HTTP::Response;
use LWP::UserAgent;
use HTML::Parser;

my ($symbol, $llid, $refseq, @accn, $unigene);
my $query;
my $dbh;
my $qh;

print header();

if(defined param('query')) {
   $query = param('query');

   if($query ne '') {
      $dbh = DBI->connect('dbi:mysql:bio', 'biouser', 'bio') or
        die("Database error (ortho.pl) $!\n");

      my $type;
      my $sql = <<'ENDSQL';
SELECT g.id, g.LocusLinkId, g.UniGeneId, g.organism, g.chromosome
FROM oldgene g WHERE
ENDSQL
      if($query =~ /^\d+$/) {
         $type = 'LocusLinkId';
      } elsif($query =~ /^[A-Z][a-z]\.\d{1,7}$/) {
         $type = 'UniGeneId';
      } elsif($query =~ /^[A-Z]{1,2}_?\d{5,6}$/) {
         $type = 'accn';
         $sql = <<'ENDSQL';
SELECT g.id, g.LocusLinkId, g.UniGeneId, g.organism, g.chromosome
FROM oldgene g JOIN oldgeneaccn ga ON g.id = ga.geneId WHERE ga.
ENDSQL
      } else {
         $type = 'symbol';
         $sql = <<'ENDSQL';
SELECT g.id, g.LocusLinkId, g.UniGeneId, g.organism, g.chromosome
FROM oldgene g JOIN genesym gs ON g.id = gs.geneId WHERE gs.
ENDSQL
      }
      $qh = $dbh->prepare($sql.$type.' = '.$dbh->quote($query).
        ' ORDER BY g.organism, g.LocusLinkId');
      $qh->execute;
   }
}

print<<ENDHTML;
<html>
<head>
   <title>Gene lookup: $query</title>
</head>
<body>
<form method="post" action="ortho.pl">
```

```
SELECT GROUP_CONCAT(accn SEPARATOR ', ') FROM oldgeneaccn
WHERE geneId = ? AND RefSeq = ?
ENDSQL

   my ($sym) = $dbh->selectrow_array(<<'ENDSQL', undef, $gid, 'false');
SELECT symbol FROM genesym WHERE geneId = ? AND alias = ?
ENDSQL

   my ($symalias) = $dbh->selectrow_array(<<'ENDSQL', undef, $gid, 'true');
SELECT GROUP_CONCAT(symbol SEPARATOR ', ') FROM genesym
WHERE geneId = ? AND alias = ?
ENDSQL

   my ($oligosn) = $dbh->selectrow_array(<<'ENDSQL', undef, $gid);
SELECT GROUP_CONCAT(oligoSn SEPARATOR ', ') FROM geneoligo
WHERE geneId = ?
ENDSQL

   my ($ugorg, $ugcid) = split(/\./, $ugid);
   $type = "($type)" if(defined $type);
   $chrom = "(chromosome $chrom)" if(defined $chrom);

   print<<ENDHTML;
   <table>
   <tr><th valign="top">
      Organism
   </th><th>
      $org$chrom $type
   </th></tr>
   <tr><td valign="top">
      LocusLink
   </td><td>
      <a href="http://www.ncbi.nlm.nih.gov/LocusLink/LocRpt.cgi?l=$llid">$llid
</a>
   </td></tr>
   <tr><td valign="top">
      Accn
   </td><td>
      <a href="http://www.ncbi.nlm.nih.gov/entrez/viewer.cgi?
db=Nucleotide&dopt=GenBank&val=$refseq"><b>$refseq</b></a><br>
      <a href="http://www.ncbi.nlm.nih.gov/entrez/viewer.cgi?
db=Nucleotide&dopt=DocSum&dispmax=1000&val=$accn">$accn</a>
   </td></tr>
   <tr><td valign="top">
      UniGene
   </td><td>
      <a href="http://www.ncbi.nlm.nih.gov/UniGene/clust.cgi?ORG=$ugorg&CID=
$ugcid">$ugid</a>
   </td></tr>
   <tr><td valign="top">
      symbol
   </td><td>
      <b>$sym</b><br>
      $symalias
   </td></tr>
ENDHTML

   if($oligosn) {
      print<<ENDHTML;
   <tr><td valign="top">
      Oligo Serial Number
```

Subject: Jon Scharff
To: Michelle Blackwell <mblackwe@utk.edu>

Michelle

Jon tells me that you'd like more information about his senior project. I'm not sure exactly what is needed, but will remark that he's undertaken a challenging task and done a very professional and admirable job. His work integrates computational tools with large-scale mammalian genomic datasets. All of us here at UT and at ORNL have been extremely pleased with both his work ethic and his results. With just a little more elbow grease, spit and polish, his efforts would almost suffice as an MS thesis. In short, I think Jon has accomplished far and away more than I've seen done in an typical BS senior project.

I hope this is the sort of information you wanted. Please let me know if I can be of more help.

Regards,

Mike Langston

--
Michael A. Langston
Professor, Department of Computer Science
University of Tennessee
Knoxville, TN 37996-3450
USA
865-974-3534
http://www.cs.utk.edu/~langston
--