Virtual Hydrocarbon Databases for use with CAVEAT

Yongliang Yang, Dimitri Nesterenko, Ryan Trump, Ken Yamaguchi, Paul A. Bartlett, and Dale G. Drueckhammer*

Department of Chemistry, Stony Brook University, Stony Brook, NY 11794
Center for New Directions in Organic Synthesis, Department of Chemistry, University of California, Berkeley, CA 94720

Supporting Information

Unix scripts used in database construction:

1. base_conversion

Generates a set of base 4 numbers in which all but three of the digits are 0, converts these base 4 numbers to base 10, and adds 1 to each.

```
#!/usr/bin/perl
use strict;
my $num;
# = $ARGV[0];
#my $base = $ARGV[1] || 2;
sub base_conversion;
sub count_zeros;
sub base_conversion {
	my $num = $_[0];
	my $base = $_[1];
	my $temp;
	my $result;
	#either user specified or 2 if none mentioned
```

```
        while($num >= $base){
                $temp = $num % $base;
                $result = ($temp.$result);
                $num = int($num / $base);
        }
        $result = ($num.$result);
        print ($result);
        print("\n");
        return $result;
       }
 sub count_zeros {
        my $i;
        my $num_array = $_[0];
        my $zero_count = 0;
        for($i=0 ; $i<length($num_array); $i++){
                                #print($num_array[$i]);
                                if(substr($num_array, $i, 1)==0){
                                        $zero_count++;
                                }
                                #print("\n");
                        }
        return ($zero_count);
}
$num = base_conversion($ARGV[0], $ARGV[1]);
print(count_zeros($num)."\n");
```

---

2. find_hydrogen_sd

Usage: find_hydrogen_sd /filepathname > newfilename.sd

This selects the trisubstitued compounds and saves them to the new file name.

```
#!/usr/bin/perl
use strict;
#use base_conversion;
#read the first line of the molecule
#get it's number ie ModelX-N
#find the number of zeros in N in base 4
#if the number is = 5 (or if non_zeros == 3)output the molecule else
#skip to the next one
sub find_hydrogen;
sub base_conversion;
sub count_zeros;
sub count_non_zero;
sub base_conversion {
        my $num = $_[0];
        my $base = $_[1];
        my $temp;
        my $result;
        #either user specified or 2 if none mentioned
                while($num >= $base){
                  $temp = $num % $base;
                  $result = ($temp.$result);
                  $num = int($num / $base);
        }
        $result = ($num.$result);
        #print ($result);
        #print("\n");
        return $result;
        }
        sub count_zeros {
        my $i;
        my $num_array = $_[0];
```

```
	my $zero_count = 0;
	for($i=0 ; $i<length($num_array); $i++){
				#print($num_array[$i]);
				if(substr($num_array, $i, 1)==0){
					$zero_count++;
				}
				#print("\n");
			}
	return ($zero_count);
}
	sub count_non_zero {
	my $i;
	my $num_array = $_[0];
	my $non_zero_count = 0;
	for($i=0 ; $i<length($num_array); $i++){
				#print($num_array[$i]);
				if(substr($num_array, $i, 1)!=0){
					$non_zero_count++;
				}
				#print("\n");
			}
	return ($non_zero_count);
}
sub find_hydrogen{
	my $line = 0;
	my $matched = 0;
	my $molecules_read = 0;
	my $num;
	my $temp_line;
	my $non_hydrogen_num = 3;#$_[0]; #the first parameter passed is hydrogen num
	MOL: while(<>){
```

```
		$line++;
		#read first line
		1==$line && do {
			$temp_line = $_;
			$num = substr($temp_line, rindex($_, "-")+1);
			$num--;
			$molecules_read++;
			#print STDERR "the number is $num \n";
			#print out the molecule if more than 3 non-hydrogens (less than 5
hydrogen)
			if(count_non_zero(base_conversion($num, 4)) ==
$non_hydrogen_num){
				$matched++;
				print ($temp_line);
				while(<>){
				  print;
				  /^\$\$\$\$/ && do{$line = 0 ; next MOL;};
				}
			}
			else
			{
				while(<>){
				  /^\$\$\$\$/ && do {$line = 0; next MOL;};
				}
			}
		};
		#print;
		/^\$\$\$\$/ && do {$line = 0;};
	}
	$molecules_read > 0 and print STDERR "$molecules_read total molecules\n";
	$matched > 0 and print STDERR "$matched molecules found\n";
```

```
}
#find_hydrogen($ARGV[2]);
find_hydrogen;
```

---

3. fix-msi-sd

To the output sd file from Cerius2 this checks for and removes any duplicate molecules, inserts the term “-ISIS-“ in the header information for each molecule, and removes the charge notation.

```
#!/usr/bin/perl –f
#Usage:  fix-msi-sd  sd-file > new-sd-file
#removes duplicate molecules and inserts “-ISIS-“header information

use strict;
my $atom_end;
my $line = 0;
my %names;
my $dups = 0

MOL: while (<> ) {
$line++;

#Header Block
1 == $line && do {
if (exists ($names {$_})) {
$dups++;
While (<>){
/^\$\$\$\$/ && do { $line=0; next MOL; };
                                        }
                                }

 $names {$_}=1;

};

2 == $line && do {
#Catalyst program insists only ISIS files can have stereochemistry
substr($_, 2, 8) = “-ISIS-“ ;
print; next;
```

```
};

$line < 4 && do {
$atom_end = substr($_, 0, 3) + 4;
Substr($_,14,1) = “1” ;
print; next
};

#Atom Block
$line <= $atom_end && do {
#Remove any charge on atoms.
/^.{38}[^0]/&& do {substr($_,38,1)=”0”; };
print; next;
};

#Kill charge lines

/^M CHG/ && next;

#everything else

print;

/^\$\$\$\$/ && do {$line=0 ;};
}

$dups > 0 and print STDERR
“$dups duplicate compound”, ($dups>1 ? “s”: “”), eliminated.\n”;
```

---

4. K shell script to run batch minimization in Macromodel

```
#!/bin/ksh

export MMOD_ROOT=/mnt/troy/disk2/macromodel.old

export LM_LICENSE_FILE=7182@tanager.chem.sunysb.edu

export BATCH_ROOT=$MMOD_ROOT/run/mmdat

for dir in errors Input-1 minimized-1;

 if [ ! -d $dir ]

 then

 mkdir $dir

 fi
```

```
done
echo "Minimizations started: `date`"
for mmod in $1/*.mmod ; do
name=`echo $mmod|sed -e "s/.mmod$//"`
echo "Working on name: $name\n"
#output com file
 cat>$name.com<<_EOD_
$name.mmod
$name.out
 FFLD   10   0   0   1
 DEMX    0 2500 0  0  20.92  41.84
 MULT
 BGIN
 READ
 COMP
 MINI   9    0   5000
 MINI   4    0   5000
 END
_EOD_
#minimize the conformers
#10 chances
cnt=0
while [ $cnt -lt 10 ]; do
echo "Try the command for the $cnt time"
nice $BATCH_ROOT/bmin.auto $name
if [ -f $name.out ]
then
echo "file $name.out found, break"
break
fi
let cnt=$cnt+1
```

```
#give networks a chance to recover
sleep 5
done
if [ -f $name.tmp -o ! -f $name.out ]
then
#abnormal conditions
mv $name.* errors
echo "$name had errors"
else
#clean up
#mv $mmod Input-
mv $name.out minimized-1
rm $name.com $name.log # $name.inf
echo $name
fi
done
echo "Minimization completed:`date`"
```

---

5. fix-header-output

Usage: The Macromodel program does not distinguish conformational structures' names from Batchmin minimization process, thus it is necessary to fix the header information from its output files.

```
#! /usr/bin/ksh
#The first loop for every .out file from minimization process from Macromodel
for out in $1/*.out;
do
name=`basename $out | sed –e "s/.out$//"`
dname=`dirname $out`
cat $out | awk –f count.awk > $name.tmp
```

```
# count.awk is displayed below
# add .out file name+a in front of `CONF` and substitute it.
sed –e “s/CONF/${name}a/”  $name.tmp > $dname/$name.out
rm –rf  $name.tmp
done
count.awk
#The inner loop will add line number behind `CONF`.
{
If (n= = 0) n=n+1
ret=sub(“CONF”, “CONF”n, $0)
if (ret>0) n=n+1
print $0
}
```

---

6. grab-minimized-source

Usage: To grab minimized output files from Macromodel BatchMin process and put them into the database source directory.

```
#!/usr/bin/ksh
#either k shell or c shell
for dir in minimized-all , DBS;  do
if [ ! –d  $dir ]
then
mkdir $dir
fi
done
integer newnumber=$count
echo “set up for CAVEAT source database started: `date`”
#output minimized files from Macromodel
for out in $1/*.out ;  do
```

```
name=`echo $out | sed –e “s/.out$//`
cp $name.out M-$newnumber.out
mv M-$newnumber.out $1
rm $name.out
newnumber=newnumber+1
#every output file is enumerated in sequence
done
echo “copy is completed:`date`”
```