Daylight Chemical Information Systems, Inc.
120 Vantis - Suite 550 - Aliso Viejo, CA 92656
tel +1 949-831-9990 - fax +1 949-831-9902 - www.daylight.com

Daylight Properties Manual

Table of Contents

Daylight Properties Package Reference Manual

Daylight Version 4.9
Release Date 08/01/11

Copyright notice: This document is copyrighted © 2011 by Daylight Chemical Information Systems, Inc. of Aliso Viejo, CA. Daylight explicitly grants permission to reproduce this document under the condition that it is reproduced in its entirety, including this notice. All other rights are reserved.

1. Introduction

1.1 Why use the Daylight Properties Package?

With the advent of virtual screening, there is a requirement for rapid estimation of physical properties directly from molecular structure. The Daylight Properties Package addresses this need by providing a command line program called Dayprop and an interactive program object called Dayproptalk. In addition there are SQL scripts to use the Dayproptalk program with Daycart[tm] within Oracle. The modular architecture of the package allows for additional properties to be added in the future.

2. Prerequisites

2.1 Programming Knowledge

Users of this manual should have general UNIX skills as well as general knowledge of Daylight software. Users of Dayproptalk should be familiar with the concept of program objects, as well as how to use them. Users of DayCart should have general knowledge of Oracle.

2.2 Software Requirements

Dayprop is included with the standard Daylight distribution (versions 4.82 or later), which can be downloaded from Daylight's web site (http://www.daylight.com). In order to use Dayprop, a "dayprop" program license from Daylight is required. Within Oracle, DayCart must be licensed to call the property package functions.

2.3 UNIX Configuration

Users need to have the appropriate environment variables set. For more information see the Daylight Installation Guide instructions.

3. Conventions.

`Teletype` is used for routine names, source code, computer output, object names, and environment variables.

`Bold` is used for commands you type.

Italic is used for filenames, variable names (except in examples), and emphasis in a description.

4. Using Dayprop and Tautomer

Dayprop and Tautomer are a non-interactive program that takes Thor Data Tree (TDT) input containing SMILES and provides output in TDT format.

4.1 Dayprop

Dayprop takes a property to be calculated with the `-PROPERTY` option and outputs the calculated value into the PPROP datatype in the TDT.

Usage:

`% dayprop [options] [infile.tdt [outfile.tdt]]`
If present, Dayprop will read in the input from `infile.tdt` and write it's output to `outfile.tdt`. If filenames are not specified Dayprop will read from standard input and write to standard output.

The output PPROP datatype consists of four parts: the property to be calculated, the calculated value, the method used to calculate the value, and a comment.

For example:

```
% dayprop -property AVERAGE_MOL_WT -propid "MyComment"
$SMI<CC>|
PPROP<1;30.08;0;MyComment;>|
```

In the above example output, `1` is the indirect identifier for the property AVERAGE_MOL_WT, `30.08` is the computed `AVERAGE_MOL_WT` of ethane, `0` is the indirect identifier for the method version used to calculate the property, "`MyComment`" is the user supplied comment.

Dayprop options:
Note: Options are case insenstitive and can not be abbreviated.

`% dayprop -h`
`% dayprop -HELP [TRUE|FALSE]`

Write the help message to standard output and exit. **`% dayprop -version`**
`% dayprop -VERSION [TRUE|FALSE]`

Write the version number of the dayprop program to standard output (ex. 4.95) and exit. If the dayprop program is not accessible for any reason, dayprop outputs `000` **`% dayprop -dump_indirect`**
`% dayprop -DUMP_INDIRECT [TRUE|FALSE]`

Write the indirect data for use in the Thor databases to standard output and exit:

```
% dayprop -dump_indirect
$PI<0;NO_PROPERTY>|
```

```
$PI<1;AVERAGE_MOL_WT>|
$PI<2;MOL_FORM>|
$PI<3;ROTBOND_COUNT>|
$PI<4;HDONOR_COUNT>|
$PI<5;HACCEPTOR_COUNT>|
$PI<6;PARACHOR>|
$PI<7;ACCURATE_MASS>|
$PI<8;MOLAR_VOLUME>|
$PI<9;RING_COUNT>|
$PI<10;RIGIDITY>|
$PI<11;FRAGMENT_COUNT>|
$PI<12;FLEXIBILITY>|
$PI<13;FINGERPRINT>|
$PI<14;DEPICTION>|
$PI<15;STEREOCENTER_COUNT>|
$PI<16;PART_COUNT>|
$PI<17;POLAR_SURFACE_AREA>|
$PI<18;ATOM_COUNT>|
$PI<19;MATCH_COUNT>|
$PI<20;ALL>|
$MI<0;4.95 DCIS Standard>|
```

% **dayprop -PROPERTY *property***

Works with all the properties listed below in the properties section, as well as the property `ALL`, which lists all the properties (except for `MATCH_COUNT`) at once.

Example of a single property calculation:

```
 % dayprop -PROPERTY ATOM_COUNT
$SMI<CC>|
$SMI<CC>PPROP<18;2;0;>|
```

Example using the `ALL` property to calculate all the properties at once:

```
% dayprop -PROPERTY ALL
$SMI<CC>
|
$SMI<CC>
PPROP<1;30.08;0;>
PPROP<2;C2H6;0;>
PPROP<3;0;0;>
PPROP<4;0;0;>
PPROP<5;0;0;>
PPROP<6;110.40;0;>
PPROP<7;30.046951;0;>
PPROP<8;56.00;0;>
PPROP<9;0;0;>
PPROP<10;1.0000;0;>
```

```
PPROP<11;1;0;>
PPROP<12;0.00;0;>
PPROP<13;.....................................E
.............................................
.....................................................................+
.........................20.....................
.................................6.2.........6
.............................................................
.......................1;0;>
PPROP<14;-0.51,1.53,0.51,1.53;0;>
PPROP<15;0;0;>
PPROP<16;1;0;>
PPROP<17;0.00;0;>
PPROP<18;2;0;>
|
```

% dayprop -PROPID "*comment*"

Adds a comment to the comment field in PPROP.

For example:

```
% dayprop -PROPERTY AVERAGE_MOL_WT -PROPID "MyComment"
$SMI<CC>|
$SMI<CC>PPROP<1;30.08;0;MyComment;>| % dayprop -SMARTS 'smarts'
```

Defines the user pattern to be used with the `MATCH_COUNT` property.

For example:

```
% dayprop -PROPERTY MATCH_COUNT -SMARTS '[#6]'
$SMI<CC>|
$SMI<CC>PROP<19;2;0;>| % dayprop -SINGLE_PART [TRUE|FALSE]
```

When `TRUE`, treats the entire input SMILES as a single molecule and computes a single property value. When `FALSE`, computes a comma separated list of property values for the dot separated components within the input SMILES. The default is `FALSE`.
NOTE: `-RXNDIFF TRUE` can not be used with `-SINGLE_PART FALSE` option.

For example:

```
% dayprop -PROPERTY AVERAGE_MOL_WT -SINGLE_PART TRUE
$SMI<CC.CC>|
$SMI<CC.CC>PROP<1;60.16;0;>| % dayprop -RXNDIFF [TRUE|FALSE]
```

Calculates the difference between the property values of the product and reactant. The default for the `RXNDIFF` option is `FALSE`. Note that `-RXNDIFF TRUE` can not be used with `-SINGLE_PART`

FALSE.

For example:

```
% dayprop -PROPERTY AVERAGE_MOL_WT -SINGLE_PART TRUE -RXNDIFF TRUE
$SMI<"CCCN>>CN">|
$SMI<"CCCN>>CN">PPROP<1;-28.06;0;>|
```

4.2 Tautomer

% tautomer [options] [infile.tdt [outfile.tdt]]
If present, Tautomer will read in the input from `infile.tdt` and write it's output to `outfile.tdt`. If filenames are not specified Tautomer will read from standard input and write to standard output. The original SMILES is returned in the list of tautomers.

The output $TAUT datatype consists of one part: the calculated value.

For example:

```
% tautomer
$SMI<Oc1cc(O)ncn1> |
$SMI<Oc1cc(O)ncn1>
$TAUT<O=C1CC(=O)N=CN1>
$TAUT<OC1=NC=NC(=O)C1>
$TAUT<Oc1cc(=O)[nH]cn1>
$TAUT<Oc1cc(=O)nc[nH]1>
$TAUT<Oc1cc(O)ncn1>
|
```

Tautomer options:
Note: Options are case insenstitive and can not be abbreviated.

```
% tautomer -h
% tautomer -HELP [TRUE|FALSE]
```

Write the help message to standard output and exit. `% tautomer -version`
`% tautomer -VERSION [TRUE|FALSE]`

Write the version number of the tautomer program to standard output (ex. 4.95) and exit. If the tautomer program is not accessible for any reason, tautomer outputs `000`
`% tautomer -NO_ENOL [TRUE|FALSE]`

This option allows only heteroatoms to participate as hydrogen donors or acceptors. This option suppresses keto-enol type tautomerism.
The default is `FALSE`.

```
% tautomer -NO_ENOL TRUE
$SMI<Oc1cc(O)ncn1>
|
$SMI<Oc1cc(O)ncn1>
```

```
$TAUT<Oc1cc(=O)[nH]cn1>
$TAUT<Oc1cc(=O)nc[nH]1>
$TAUT<Oc1cc(O)ncn1>
|
% tautomer -ISO [TRUE|FALSE]
```

The -ISO FALSE option returns the tautomers as unique SMILES. When true the program returns the tautomers as absolute SMILES.
The default is `FALSE`.

```
%tautomer -ISO TRUE
$SMI<CC(=O)[C@H](C)C(=O)OC>
|
$SMI<CC(=O)[C@H](C)C(=O)OC>
$TAUT<COC(=C(C)C(=C)O)O>
$TAUT<COC(=C(C)C(=O)C)O>
$TAUT<COC(=O)C(=C(C)O)C>
$TAUT<COC(=O)[C@@H](C)C(=C)O>
$TAUT<COC(=O)[C@@H](C)C(=O)C>
|
% tautomer -KEKULE [TRUE|FALSE]
```

The KEKULE option when TRUE generates kekule structures using dt_xsmiles(). When FALSE the program generates canonical SMILES using dt_cansmiles();
The default is `FALSE`.

```
tautomer -KEKULE TRUE
$SMI<Oc1cc(O)ncn1>
|
$SMI<Oc1cc(O)ncn1>
$TAUT<O=C1CC(=O)N=CN1>
$TAUT<OC1=CC(=O)N=CN1>
$TAUT<OC1=CC(=O)NC=N1>
$TAUT<OC1=NC=NC(=O)C1>
$TAUT<OC=1C=C(O)N=CN1>
|
% tautomer -UNIQUE [TRUE|FALSE]
```

When TRUE the UNIQUE option writes out the canonical tautomer. The canonical tautomer is the one generated by using the relative electronegativities of the atom types (O > S > Se > Te > N > C) as graph invariants to preferentially assign double bond and hydrogen positions in the tautomer. Although this tautomer often corresponds to the lowest energy form, it is not guaranteed, as it's generated from graph theory and does not consider extended electronic factors.
The default is `FALSE`

```
% tautomer -UNIQUE TRUE
$SMI<Oc1cc(O)ncn1>
|
$SMI<Oc1cc(O)ncn1>
$TAUT<Oc1cc(=O)[nH]cn1>
```

```
|
% tautomer -ITERATION_LIMIT [LIMIT]
```

This is the maximum number of donor or acceptor positions to iterate. If a structure has more than LIMIT of either donors or acceptors, then no tautomer enumeration is performed. The default limit is 0, which causes the program to generate tautomers for every input structure until all possible tautomers have been generated. A reasonable value for limit to minimize long-running, pathological cases, is 10.

```
% tautomer -FIXED_SUBSTRUCTURE [SUBSTRUCTURE]
```

A comma-separated list of SMARTS which are matched against each input molecule. Any atoms which match are marked as non-tautomerizable and hence stay fixed throughout the enumeration. Useful for excluding specific functional groups from the calculation.

```
% tautomer
$SMI<O=CCCCC(=O)N>
|
$SMI<O=CCCCC(=O)N>
$TAUT<NC(=CCC=CO)O>
$TAUT<NC(=CCCC=O)O>
$TAUT<NC(=O)CCC=CO>
$TAUT<NC(=O)CCCC=O>
$TAUT<OC(=N)CCCC=O>
$TAUT<OC=CCCC(=N)O>
|

vs:
% tautomer -FIXED_SUBSTRUCTURE 'O=CN'
$SMI<O=CCCCC(=O)N>
|
$SMI<O=CCCCC(=O)N>
$TAUT<NC(=O)CCC=CO>
$TAUT<NC(=O)CCCC=O>
|
```

5. Using Dayproptalk and Tautomertalk

Dayproptalk is an interactive program object that parallels Dayprop.
Tautomertalk is an interactive program object that parallels Tautomer.

5.1 What are Program Objects?

Program objects are used to provide two-way communication with an external process. For example, the clogptalk program for computing hydrophobicity for a structure represented in SMILES. Using program objects, a calling program can start an external program, send input, receive the program's output and perform other tasks while the external program remains running and ready for more input. This is particularly important for many property programs which spend significant amounts of time initializing themselves. A number of programs supporting program objects are supplied with the release of Daylight Software. Most of

these are supplied as contributed code in the progob directory. The commercial programs clogptalk and cmrtalk operate as program objects (in `$DY_ROOT/bin`). The dayproptalk program operates in the same way, using the PIPETALK protocol.

5.2 Dayproptalk program object messages.

Dayproptalk responds to all of the standard program object messages, as defined in the Program Objects section of the Toolkit Programmer's Guide.

One minor difference is the behavior of `Qwerty: Say HELP.` If the current property is set, `HELP` returns a help message specific to that property, otherwise a generic help message is returned.

```
Qwerty: Set PROPERTY UNDEFINED.
Qwerty: Over.
Property type set to UNDEFINED...0
Qwerty: Over.
Qwerty: Say HELP.
Qwerty: Over.
Reads SMILES, computes requested physical property
-- takes one argument, the property to be calculated.
One of AVERAGE_MOL_WT, MOL_FORM, ROTBOND_COUNT, HDONOR_COUNT,
HACCEPTOR_COUNT, PARACHOR, ACCURATE_MASS, MOLAR_VOLUME,
RING_COUNT, RIGIDITY, FRAGMENT_COUNT, FLEXIBILITY,
FINGERPRINT, DEPICTION, STEREOCENTER_COUNT, PART_COUNT,
ATOM_COUNT, POLAR_SURFACE_AREA
Qwerty: Over.
Qwerty: Set PROPERTY AVERAGE_MOL_WT.
Qwerty: Over.
Property type set to AVERAGE_MOL_WT...1
Qwerty: Over.
Qwerty: Say HELP.
Qwerty: Over.
Reads SMILES, calculates molecular weight based on
average atomic weights for naturally occuring elements
Qwerty: Over.
```

Dayproptalk also responds to the following object messages:

"Qwerty: Set PROPERTY *property*."

Sets the property value.

For example:

```
Qwerty: Set PROPERTY AVERAGE_MOL_WT.
Qwerty: Over.
Property type set to AVERAGE_MOL_WT...1
Qwerty: Over.
```

NOTE: The Dayproptalk program optionally takes a single argument, the property name. When present, dayproptalk starts with the property set to the given value.

"Qwerty: Say PROPERTY".

Shows the current property value.

```
Qwerty: Say PROPERTY.
Qwerty: Over.
Property set to AVERAGE_MOL_WT.
Qwerty: Over.  "Qwerty: Set SMARTS SMARTS."
```

Sets the user defined SMARTS pattern.

```
Qwerty: Set SMARTS [#6].
Qwerty: Over.
SMARTS set to [#6]
Qwerty: Over.

 "Qwerty: Say SMARTS."
```

Shows the current user defined SMARTS pattern.

```
Qwerty: Say SMARTS.
Qwerty: Over.
SMARTS set to [#6]
Qwerty: Over.  "Qwerty: Set VALUE_ONLY TRUE."
"Qwerty: Set VALUE_ONLY FALSE."
```

When `TRUE`, sets the output to suppress the SMILES in the output and return only the calculated value. When `FALSE`, sets the output to return the both the input SMILES and the calculated value. The default is `FALSE`.

For example:

```
Qwerty: Set PROPERTY AVERAGE_MOL_WT.
Qwerty: Over.
Property type set to AVERAGE_MOL_WT...1
Qwerty: Over.
CC
Qwerty: Over.
CC 30.08
Qwerty: Over.
Qwerty: Set VALUE_ONLY TRUE.
Qwerty: Over.
Value only output toggled
Qwerty: Over.
CC
Qwerty: Over.
30.08
Qwerty: Over.
Qwerty: Set VALUE_ONLY FALSE.
```

```
Qwerty: Over.
Value only output toggled
Qwerty: Over.
CC
Qwerty: Over.
CC 30.08
Qwerty: Over.

  "Qwerty: Set SINGLE_PART TRUE."
"Qwerty: Set SINGLE_PART FALSE."
```

When TRUE treats the input SMILES as a single molecule and computes a single property value. When FALSE, computes a property value for each dot separated component within a SMILES. SINGLE_PART FALSE can not be used with the option RXNDIFF TRUE. The default is FALSE.

For example:

```
Qwerty: Set SINGLE_PART FALSE.
Qwerty: Set PROPERTY AVERAGE_MOL_WT.
Qwerty: Over.
Property type set to AVERAGE_MOL_WT...1
Qwerty: Over.
CC.CC
CC.CC 30.08,30.08
Qwerty: Over.
Qwerty: Set SINGLE_PART TRUE.
Qwerty: Over.
Single part only output toggled
Qwerty: Over.
CC.CC
Qwerty: Over.
CC.CC 60.16
Qwerty: Over.
 "Qwerty: Set RXNDIFF TRUE."
"Qwerty: Set RXNDIFF FALSE."
```

When TRUE, returns the difference of computed property values between the product and reactant molecules. When FALSE, returns the computed property values for the individual component molecules. The default is FALSE.
NOTE: SINGLE_PART FALSE can not be used with the option RXNDIFF TRUE.

```
Welcome to dayproptalk
Qwerty: Over.
Qwerty: Set PROPERTY AVERAGE_MOL_WT.
Qwerty: Over.
Property type set to AVERAGE_MOL_WT...1
Qwerty: Over.
Qwerty: Set SINGLE_PART TRUE.
Qwerty: Over.
```

```
Single part only output toggled
Qwerty: Over.
Qwerty: Set RXNDIFF TRUE.
Qwerty: Over.
Single part rxn only output toggled
Qwerty: Over.
CCCN>>CN
Qwerty: Over.
CCCN>>CN -28.06
Qwerty: Over.
```

Dayproptalk can be used to succesively to compute a set of properties because the data after the SMILES is preserved with each run of dayproptalk:

```
#!/bin/sh
#
# This shell script produces a space separated table containing:
# SMILES Mol_Wt Rot_Bonds Frag_Count
# suitable for analysis program such as Excel or JMP

# Put in column headers
echo " SMILES Mol_wt Rot_Bonds Frag_Count" > outfile
#
cat $1 \
| pipetalker dayproptalk AVERAGE_MOL_WT \
| pipetalker dayproptalk ROTBOND_COUNT \
| pipetalker dayproptalk FRAGMENT_COUNT \
| grep -v "Welcome" >> outfile
```

```
$ sh test.sh smiles
$ more outfile
SMILES Mol_wt Rot_Bonds Frag_Count
CCCCN 73.16 2 1
CCCN 59.13 1 1
CCN 45.10 0 1
```

5.3 Tautomertalk program object messages.

Tautomertalk responds to all of the standard program object messages, as defined in the Program Objects section of the Toolkit Programmer's Guide.
Tautomertalk also responds to the following object messages:

"Qwerty: Set NO_ENOL TRUE."
"Qwerty: Set NO_ENOL FALSE."

This option allows only heteroatoms to participate as hydrogen donors or acceptors. This option suppresses keto-enol type tautomerism.
The default is `FALSE`.

For example:

```
Welcome to tautomertalk
```

```
Qwerty: Over.
Oc1cc(O)ncn1
Qwerty: Over.
O=C1CC(=O)N=CN1
OC1=NC=NC(=O)C1
Oc1cc(=O)[nH]cn1
Oc1cc(=O)nc[nH]1
Oc1cc(O)ncn1
Qwerty: Over.
Qwerty: Set NO_ENOL TRUE.
Qwerty: Over.
No_enol option toggled
Qwerty: Over.
Oc1cc(O)ncn1
Qwerty: Over.
Oc1cc(=O)[nH]cn1
Oc1cc(=O)nc[nH]1
Oc1cc(O)ncn1
Qwerty: Over.

 "Qwerty: Set ISO TRUE."
"Qwerty: Set ISO FALSE."
```

The ISO FALSE option returns the tautomers as unique SMILES. When true the program returns the tautomers as absolute SMILES.
The default is `FALSE`.

For example:

```
Welcome to tautomertalk
Qwerty: Over.
CC(=O)[C@H](C)C(=O)OC
Qwerty: Over.
COC(=C(C)C(=C)O)O
COC(=C(C)C(=O)C)O
COC(=O)C(=C(C)O)C
COC(=O)C(C)C(=C)O
COC(=O)C(C)C(=O)C
Qwerty: Over.
Qwerty: Set ISO TRUE.
Qwerty: Over.
Iso option toggled
Qwerty: Over.
CC(=O)[C@H](C)C(=O)OC
Qwerty: Over.
COC(=C(C)C(=C)O)O
COC(=C(C)C(=O)C)O
COC(=O)C(=C(C)O)C
COC(=O)[C@@H](C)C(=C)O
COC(=O)[C@@H](C)C(=O)C
```

```
Qwerty: Over.
 "Qwerty: Set KEKULE TRUE."
"Qwerty: Set KEKULE FALSE."
```

The KEKULE option when TRUE generates kekule structures using dt_xsmiles(). When FALSE the program generates canonical SMILES using dt_cansmiles();
The default is `FALSE`.

For example:

```
Welcome to tautomertalk
Qwerty: Over.
Oc1cc(O)ncn1
Qwerty: Over.
O=C1CC(=O)N=CN1
OC1=NC=NC(=O)C1
Oc1cc(=O)[nH]cn1
Oc1cc(=O)nc[nH]1
Oc1cc(O)ncn1
Qwerty: Over.
Qwerty: Set KEKULE TRUE.
Qwerty: Over.
Kekule option toggled
Qwerty: Over.
Oc1cc(O)ncn1
Qwerty: Over.
O=C1CC(=O)N=CN1
OC1=CC(=O)N=CN1
OC1=CC(=O)NC=N1
OC1=NC=NC(=O)C1
OC=1C=C(O)N=CN1
Qwerty: Over.
```

```
 "Qwerty: Set UNIQUE TRUE."
"Qwerty: Set UNIQUE FALSE."
```

When TRUE the UNIQUE option writes out the canonical tautomer.
The default is `FALSE`.

For example:

```
Welcome to tautomertalk
Qwerty: Over.
Oc1cc(O)ncn1
Qwerty: Over.
O=C1CC(=O)N=CN1
OC1=NC=NC(=O)C1
Oc1cc(=O)[nH]cn1
Oc1cc(=O)nc[nH]1
```

```
Oc1cc(O)ncn1
Qwerty: Over.
Qwerty: Set UNIQUE TRUE.
Qwerty: Over.
Unique option toggled
Qwerty: Over.
Oc1cc(O)ncn1
Qwerty: Over.
Oc1cc(=O)[nH]cn1
Qwerty: Over.
 "Qwerty: Set ITERATION_LIMIT [LIMIT]."
```

This is the maximum number of donor or acceptor positions to iterate. If a structure has more than LIMIT of either donors or acceptors, then no tautomer enumeration is performed. The default limit is 0, which causes the program to generate tautomers for every input structure until all possible tautomers have been generated. A reasonable value for limit to minimize long-running, pathological cases, is 10. **"Qwerty: Set FIXED_SUBSTRUCTURE [SUBSTRUCTURE]."**

> The SUBSTRUCTURE is a comma separated list of SMARTS which are matched against each input molecule. Any atoms which match are marked as non-tautomerizable and hence stay fixed throughout the enumeration. Useful for excluding specific functional groups from the calculation.
>
> For example:

```
Welcome to tautomertalk
Qwerty: Over.
NC(=O)CCCC=O
Qwerty: Over.
NC(=CCC=CO)O
NC(=CCCC=O)O
NC(=O)CCC=CO
NC(=O)CCCC=O
OC(=N)CCCC=O
OC=CCCC(=N)O
Qwerty: Over.
Qwerty: Set FIXED_SUBSTRUCTURE O=CN.
Qwerty: Over.
Substructure fixed to O=CN
Qwerty: Over.
O=CCCCC(=O)N
Qwerty: Over.
NC(=O)CCC=CO
NC(=O)CCCC=O
Qwerty: Over.
```

6. Using Dayproptalk and Tautomertalk with DayCart[tm]

The PIPETALK protocol provides an ideal way for the Daylight Oracle cartridge, Daycart[TM] to communicate with external functions. For instance, to populate a table column using property values with a SQL command:

```
UPDATE my_table SET Molwt = average_mol_wt(smiles);
```

6.1 Dayproptalk with DayCart[tm]

The script `dy_props_create.plb` creates the ddprop package to be used with DayCart. The script `dy_props_clean.plb` removes the ddprop package. The singular versions of the functions (i.e. `atom_count`) compute the property with `SINGLE_PART TRUE`. The plural versions of the functions (i.e. `atom_counts`) compute a comma separated list of properties with `SINGLE_PART FALSE`.

For example:

```
SQL> select atom_count('CCC') from dual;

ATOM_COUNT('CCC')
-----------------
3

SQL> select atom_counts('CCC.CCC') from dual;

ATOM_COUNTS('CCC.CCC')
--------------------------------------------------------------------------------
3,3 The following property functions are provided:
```

The following functions take a `VARCHAR2` or `CLOB` and return a single computed value using `SINGLE_PART TRUE`.

```
operator function_name (
smiles in VARCHAR2_OR_CLOB
)
=> NUMBER
```

Where `function_name` is:
`accurate_mass, atom_count, average_mol_wt, hacceptor_count, hdonor_count, flexibility, fragment_count,molar_volume, parachor, part_count, polar_surface_area, rigidity, ring_count, rotbond_count, stereocenter_count`

```
operator function_name (
smiles in VARCHAR2_OR_CLOB
)
=> VARCHAR2_OR_CLOB
```

Where `function_name` is:
`depiction, fingerprint, mol_form`
The following functions take a `VARCHAR2` or `CLOB` and return a comma separated list of computed

values using SINGLE_PART FALSE.

```
operator function_name (
smiles in VARCHAR2_OR_CLOB
)
=> VARCHAR2_OR_CLOB
```

Where function_name is:
accurate_masses, atom_counts, average_mol_wts, depictions, hacceptor_counts, hdonor_counts, flexibilities, fingerprints, fragment_counts, mol_forms, molar_volumes, parachors, polar_surface_areas rigidities, ring_counts, rotbond_counts, stereocenter_counts

The functions match_count and match_counts have the following function prototypes:

```
operator match_count (
smiles in VARCHAR2_OR_CLOB,
smarts in VARCHAR2_OR_CLOB
)
=> NUMBER


operator match_counts (
smiles in VARCHAR2_OR_CLOB,
smarts in VARCHAR2_OR_CLOB
)
=> VARCHAR2_OR_CLOB
```

6.2 Tautomertalk with DayCart[tm]

The PIPETALK protocol provides an ideal way for the Daylight Oracle cartridge, Daycart[TM] to communicate with external functions. For instance, to populate a table column using tautomer values with a SQL command:

UPDATE my_table SET tautomer = compute_tautomer(smiles,tautomer number,iso flag,no_enol flag); The script dy_props_create.plb creates the ddprop package to be used with DayCart. The script dy_props_clean.plb removes the ddprop package.

For example:

```
SQL> select compute_tautomer('Oc1cc(=O)[nH]cn1',1,0,0) from dual;

COMPUTE_TAUTOMER('OC1CC(=O)[NH]CN1',1,'0','0')
--------------------------------------------------------------------------------
O=C1CC(=O)N=CN1
```

The following property functions are provided:

The following functions take a VARCHAR2 or CLOB and return a single computed value. The function count_tautomer counts the total number of tautomers.

```
operator count_tautomer (
sosdata IN VARCHAR2_OR_CLOB, isomer_flag IN VARCHAR2,
```

```
no_enol_flag IN VARCHAR2, kekule_flag IN VARCHAR2)
)
=> NUMBER
```

The function compute_tautomer takes a SMILES as input, the number of the tautomers to be returned, and option flags for isomer and no_enol. Tautomers are computed using aromaticity and are returned as SMILES strings.

operator compute_tautomer (smiles in VARCHAR2_OR_CLOB, tautomer_number in NUMBER, isomer_flag IN VARCHAR2, no_enol_flag IN VARCHAR2)) => VARCHAR2_OR_CLOB For example: To get the number of tautomers for the smiles 'Oc1cc(=O)[nH]cn1'.

SQL> **select count_tautomer('Oc1cc(=O)[nH]cn1',0,0,0) from dual;** COUNT_TAUTOMER('OC1CC(=O)[NH]CN1',0,0) ---------- 5 To get the 5th tautomer returned: SQL> **select compute_tautomer('Oc1cc(=O)[nH]cn1',5,0,0) from dual;** COMPUTE_TAUTOMER('OC1CC(=O)[NH]CN1',5,0,0) ---------- Oc1cc(O)ncn1 The function compute_xtautomer takes a SMILES as input, the number of the tautomers to be returned, and option flags for isomer and no_enol. Tautomers are computed without using aromaticity and their kekule SMILES are returned.

operator compute_xtautomer (smiles in VARCHAR2_OR_CLOB,tautomer_number in NUMBER, isomer_flag IN VARCHAR2, no_enol_flag IN VARCHAR2)) => VARCHAR2_OR_CLOB To get the 5th tautomer returned: SQL> **select compute_xtautomer('Oc1cc(=O)[nH]cn1',5,0,0) from dual;** COMPUTE_XTAUTOMER('OC1CC(=O)[NH]CN1',5,0,0) ---------- OC=1C=C(O)N=CN1 The compute_tautomer and compute_xtautomer functions can be put into a loop to get out all the tautomers. For example:

```
set serveroutput on declare tautomer varchar2(32000); xtautomer
varchar2(32000); taut_num number := 0; n number; begin taut_num :=
ddprop.fcount_tautomer('O=c1[nH]cccc1',0,0,0); n := 1; while n <=
taut_num loop tautomer :=
ddprop.fcompute_tautomer('O=c1[nH]cccc1',n,0,0);
dbms_output.put_line(' tautomer ' || tautomer); n := n + 1; end
loop; taut_num := ddprop.fcount_tautomer('O=c1[nH]cccc1',0,0,1); n
:= 1; while n <= taut_num loop xtautomer :=
ddprop.fcompute_xtautomer('O=c1[nH]cccc1',n,0,0);
dbms_output.put_line(' xtautomer ' || xtautomer); n := n + 1; end
loop; end;
```

The output of the above code is: tautomer O=C1C=CCC=N1 tautomer O=C1CC=CC=N1 tautomer O=c1cccc[nH]1 tautomer Oc1ccccn1 xtautomer O=C1C=CC=CN1 xtautomer O=C1C=CCC=N1 xtautomer O=C1CC=CC=N1 xtautomer OC1=CC=CC=N1 The function smi2tautomer takes in a smiles and returns the unique tautomer.

```
operator smi2tautomer (
smiles in VARCHAR2_OR_CLOB
)
=> VARCHAR2_OR_CLOB
```

For example: SQL> **select smi2tautomer('Oc1cc(O)ncn1') from dual;**
SMI2TAUTOMER('OC1CC(O)NCN1')
---------- Oc1cc(=O)[nH]cn1 The function smi2xtautomer takes in a smiles and returns the unique xtautomer.

```
For example:
SQL> select smi2xtautomer('Oc1cc(O)ncn1') from dual;

SMI2XTAUTOMER('OC1CC(O)NCN1')
--------------------------------------------------------------------------------
OC1=CC(=O)NC=N1
```

The function count_subtautomer counts the total number of tautomers with a fixed substructure. The fixed substructure is a comma-separated list of SMARTS which are matched against each input molecule. Any atoms which match are marked as non-tautomerizable and hence stay fixed throughout the enumeration. Useful for excluding specific functional groups from the calculation.

```
operator count_subtautomer (
sosdata IN VARCHAR2_OR_CLOB, isomer_flag IN VARCHAR2,
no_enol_flag IN VARCHAR2, kekule_flag IN VARCHAR2,
substructure IN VARCHAR2_OR_CLOB )
)
=> NUMBER
```

The function compute_subtautomer takes a SMILES as input, the number of the tautomer to be returned, the option flags for isomer and no_enol, and the fixed substructure. Tautomers are computed using aromaticity and are returned as SMILES strings.

operator compute_subtautomer (smiles in VARCHAR2_OR_CLOB, tautomer_number in NUMBER, isomer_flag IN VARCHAR2, no_enol_flag IN VARCHAR2, substructure IN VARCHAR2_OR_CLOB)) => VARCHAR2_OR_CLOB The function compute_xsubtautomer takes a SMILES as input, the number of the tautomer to be returned, and option flags for isomer and no_enol, and the fixed substructure. Tautomers are computed without using aromaticity and their kekule SMILES are returned.

```
operator compute_xsubtautomer (
smiles in VARCHAR2_OR_CLOB,tautomer_number in NUMBER,
  isomer_flag IN VARCHAR2, no_enol_flag IN VARCHAR2,
  substructure IN VARCHAR2_OR_CLOB)
)
=>  VARCHAR2_OR_CLOB
```

The following example shows the difference in output between compute_tautomer and compute_subtautomer and betwwen compute_xtautomer and compute_subxtautomer.

For example:

```
set serveroutput on
declare
tautomer varchar2(32000);
xtautomer varchar2(32000);
subtautomer varchar2(32000);
subxtautomer varchar2(32000);
```

```
taut_num number := 0;
n number;
begin
 taut_num := ddprop.fcount_tautomer('NC(=O)CCCC=O',0,0,0);
 n := 1;
 while n <= taut_num loop
    tautomer := ddprop.fcompute_tautomer('NC(=O)CCCC=O',n,0,0);
    dbms_output.put_line(' tautomer ' || tautomer);
    n := n + 1;
 end loop;

 taut_num := ddprop.fcount_tautomer('NC(=O)CCCC=O',0,0,1);
 n := 1;
 while n <= taut_num loop
    xtautomer := ddprop.fcompute_tautomer('NC(=O)CCCC=O',n,0,0);
    dbms_output.put_line(' xtautomer ' || xtautomer);
    n := n + 1;
 end loop;

 taut_num := ddprop.fcount_subtautomer('NC(=O)CCCC=O',0,0,0,'O=CN');
  n := 1;
  while n <= taut_num loop
     subtautomer := ddprop.fcompute_subtautomer('NC(=O)CCCC=O',n,0,0,'O=CN');
     dbms_output.put_line(' subtautomer ' || subtautomer);
     n := n + 1;
 end loop;

 taut_num := ddprop.fcount_subtautomer('NC(=O)CCCC=O',0,0,1,'O=CN');
 n := 1;
 while n <= taut_num loop
    subxtautomer := ddprop.fcompute_subtautomer('NC(=O)CCCC=O',n,0,0,'O=CN');
    dbms_output.put_line(' subxtautomer ' || subxtautomer);
    n := n + 1;
 end loop;
end;
```

The output for the above code is:

tautomer NC(=CCC=CO)O
tautomer NC(=CCCC=O)O
tautomer NC(=O)CCC=CO
tautomer NC(=O)CCCC=O
tautomer OC(=N)CCCC=O
tautomer OC=CCCC(=N)O
xtautomer NC(=CCC=CO)O
xtautomer NC(=CCCC=O)O
xtautomer NC(=O)CCC=CO
xtautomer NC(=O)CCCC=O
xtautomer OC(=N)CCCC=O
xtautomer OC=CCCC(=N)O
subtautomer NC(=CCC=CO)O
subtautomer NC(=CCCC=O)O
subxtautomer NC(=O)CCC=CO
subxtautomer NC(=O)CCCC=O

The function set_iteration_limit takes in a number as the limit. The limit is the maximum number of

donor or acceptor positions to iterate. If a structure has more than LIMIT of either donors or acceptors, then no tautomer enumeration is performed. The default limit is 0, which causes the program to generate tautomers for every input structure until all possible tautomers have been generated. A reasonable value for limit to minimize long-running, pathological cases, is 10

```
operator set_iteration_limit(limit in NUMBER) => NUMBER
```

7. Property Names and Descriptions.

The following properties can be calculated using dayprop or dayproptalk:

ACCURATE_MASS

Molecular weight in atomic mass units using the the most common isotope of each element. This version uses IUPAC 1989 values. NOTE: Isotopic atom specifications in the input SMILES are ignored for this calculation.

For example, the weight of the most common isotope for hydrogen is 1.007825 amu:

Dayprop

```
% echo '$SMI<[H]>|' | dayprop -property accurate_mass
$SMI<[H]>PPROP<7;1.007825;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY ACCURATE_MASS.
Qwerty: Over.
Property type set to ACCURATE_MASS...7
Qwerty: Over.
[H] Qwerty: Over.
[H] 1.007825
Qwerty: Over.
```

SQL

```
SQL> select accurate_mass('[H]') from dual;

ACCURATE_MASS('[H]')
--------------------
1.007825
```

ATOM_COUNT

The count of heavy atoms in a molecule. Hydrogens are always ignored. Used to moderate the molecular weight values.

For example, the atom count of methane is 1:

Dayprop

```
% echo '$SMI<C([H])([H])([H])[H]>|' | dayprop -property atom_count
$SMI<C([H])([H])([H])[H]<PPROP<18;1;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY ATOM_COUNT.
Qwerty: Over.
Property type set to ATOM_COUNT...18
Qwerty: Over.
C([H])([H])([H])[H]
Qwerty: Over.
C([H])([H])([H])[H] 1
Qwerty: Over.
```

SQL

```
SQL> select atom_count('C([H])([H])([H])[H]') from dual;

ATOM_COUNT('C([H])([H])([H])[H]')
---------------------------------
1
```

AVERAGE_MOL_WT

Molecular weight based on average atomic weights for naturally occurring elements. NOTE: Isotopic atom specifications in the input SMILES are ignored for this calculation.

For example: the average molecule weight of hydrogen is 1.01 amu:

Dayprop

```
% echo '$SMI<[H]>|' | dayprop -property average_mol_wt
$SMI<[H]>PPROP<1;1.01;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY AVERAGE_MOL_WT.
Qwerty: Over.
Property type set to AVERAGE_MOL_WT...1
Qwerty: Over.
[H]
Qwerty: Over.
[H] 1.01
Qwerty: Over.
```

SQL

```
SQL> select average_mol_wt('[H]') from dual;

AVERAGE_MOL_WT('[H]')
---------------------
1.01
```

DEPICTION

Compute planar coordinates for a depiction using the DEPICT Toolkit. Coordinates are computed for explicit atoms only.

For example, if the stereochemical hydrogen in alanine is implicit, 6 pairs of coordinates are computed:

Dayprop

```
% echo '$SMI<N[C@@](C)C(=O)O>|' | dayprop -property depiction
$SMI<N[C@@H](C)C(=O)O>PPROP<14;-0.10,1.53,-0.36,2.51,-0.63,
3.49,0.62,2.77,0.88,3.75,1.34,2.05;0;>|
```

Daypropталk

```
Qwerty: Set PROPERTY DEPICTION.
Qwerty: Over.
Property type set to DEPICTION...14
Qwerty: Over.
N[C@@H](C)C(=O)O
Qwerty: Over.
N[C@@H](C)C(=O)O

-0.10,1.53,-0.36,2.51,-0.63,3.49,0.62,2.77,0.88,3.75,1.34,2.05
Qwerty: Over.
```

SQL

```
SQL> select depiction('N[C@@H](C)C(=O)O') from dual;

DEPICTION('N[C@@H](C)C(=O)O')
--------------------------------------------------------------------------------
-0.10,1.53,-0.36,2.51,-0.63,3.49,0.62,2.77,0.88,3.75,1.34,2.05
```

In contrast, if the hydrogen is explicit, 7 pairs of coordinates will be computed:

Dayprop

```
% echo '$SMI|' | dayprop -property depiction
$SMI<N[C@@]([H])(C)C(=O)O>PPROP<14;-0.10,1.53,-0.36,2.51,-0.36,2.51,
-0.63,3.49,0.62,2.77,0.88,3.75,1.34,2.05;0;>|
```

Daypropталk

```
Qwerty: Set PROPERTY DEPICTION.
Qwerty: Over.
Property type set to DEPICTION...14
Qwerty: Over.
N[C@@]([H])(C)C(=O)O
Qwerty: Over.
N[C@@]([H])(C)C(=O)O
-0.10,1.53,-0.36,2.51,-0.36,2.51,-0.63,3.49,0.62,2.77,0.88,3.75,1.34,2.05
Qwerty: Over.
```

SQL

```
SQL> select depiction('N[C@@]([H])(C)C(=O)O') from dual;

DEPICTION('N[C@@]([H])(C)C(=O)O')
--------------------------------------------------------------------------------
-0.10,1.53,-0.36,2.51,-0.36,2.51,-0.63,3.49,0.62,2.77,0.88,3.75,1.34,2.05
```

For more information, see the manual page on `dt_calcxy()`.**FINGERPRINT**

> Generate a fingerprint using the FINGERPRINT Toolkit. Default parameters for `MINSTEP`, `MAXSTEP` and `SIZE` are `0`, `7`, and `2048`, respectively. For example, the fingerprint for ethanol is computed as follows:

Dayprop

```
% echo '$SMI<CCO>|' | dayprop -property fingerprint
$SMI<CCO>PPROP<13;.....................................E......2.............
.............................0............U................................+...
+..............+.........................20.....................................
..........2........+....6.2..........6...................6.....................
....U.........................................1;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY FINGERPRINT.
Qwerty: Over.
Property type set to FINGERPRINT...13
Qwerty: Over.
CCO
Qwerty: Over.
CCO
.....................................E......2.............................
.............0............U.................................+...+.............+
.......................20.............................................2......
.+....6.2..........6..................6.........................U.............
.............................1
Qwerty: Over.
```

SQL

```
SQL> select fingerprint('CCO') from dual;

FINGERPRINT('CCO')
--------------------------------------------------------------------------------
.....................................E......2.................................
.........0............U.................................+...+.............+....
.....................20.................................................2........+.
..6.2..........6...................6.........................U.................
.......................1
```

FLEXIBILITY

> The ratio of rotatable bonds to the total count of bonds. Values range from `1.0` (totally flexible) to `0.0` (Rigid structures). Bonds to hydrogen are excluded. For more information on rotatable bonds refer to the ROTBOND_COUNT property:

Dayprop
```
% echo '$SMI<CCCCCC>|' | dayprop -property flexibility
$SMI<CCCCCC>PPROP<12;0.60;0;>|
```

Dayproptalk
```
Qwerty: Set PROPERTY FLEXIBILITY.
Qwerty: Over.
Property type set to FLEXIBILITY...12
Qwerty: Over.
CCCCCC
Qwerty: Over.
CCCCCC 0.60
Qwerty: Over.
```

SQL
```
SQL> select flexibility('CCCCCC') from dual;

FLEXIBILITY('CCCCCC')
---------------------
.6
```

FRAGMENT_COUNT

The count of the number of fragments formed by removal of the isolated carbons from the structure. An isolated carbon is defined by the SMARTS pattern `[$([#6]);!$(C(F)(F)F);!$(c(:[!c]):[!c]);!$([#6]=,#[!#6]);!$([#6;!+0])]` Note that lone isotopic hydrogen atoms are not counted as fragments after removal of the isolating carbons. So CC[2H] and CC both have fragment counts of zero.
For example, the fragment count for ethanol is 1:

Dayprop
```
% echo '$SMI<CCO>|' | dayprop -property fragment_count
$SMI<CCO>PPROP<11;1;0;>
```

Dayproptalk
```
Qwerty: Set PROPERTY FRAGMENT_COUNT.
Qwerty: Over.
Property type set to FRAGMENT_COUNT...11
Qwerty: Over.
CCO
Qwerty: Over.
CCO 1
Qwerty: Over.
```

SQL
```
SQL> select fragment_count('CCO') from dual;

FRAGMENT_COUNT('CCO')
---------------------
```

```
1
```

The fragment count for ethane is 0:

Dayprop

```
% echo '$SMI<CC>|; | dayprop - property fragment_count
$SMI<CC>PPROPlt;11;0;0;gt;|
```

Dayproptalk

```
Qwerty: Set PROPERTY FRAGMENT_COUNT.
Qwerty: Over.
Property type set to FRAGMENT_COUNT...11
Qwerty: Over.
CC
Qwerty: Over.
CC 0
Qwerty: Over.
```

SQL

```
SQL> select fragment_count('CC') from dual;

FRAGMENT_COUNT('CC')
--------------------
0
```

HACCEPTOR_COUNT

Number of hydrogen-bonding acceptor sites as defined by the SMARTS pattern `[$([!#6;+0]);!$([F,Cl,Br,I]);!$([o,s,nX3]);!$([Nv5,Pv5,Sv4,Sv6])]`. Heavy atoms may have multiple hydrogen-bonding sites.

For example, the number of hydrogen-bonding acceptor sites for water is 2 (one for each electron pair):

Dayprop

```
% echo '$SMI<O>|' | dayprop -property hacceptor_count
$SMI<O>PPROP<5;2;0;&gt|
```

Dayproptalk

```
Qwerty: Set PROPERTY HACCEPTOR_COUNT.
Qwerty: Over.
Property type set to HACCEPTOR_COUNT...5
Qwerty: Over.
O
Qwerty: Over.
O 2
Qwerty: Over.
```

SQL
```
SQL> select hacceptor_count('O') from dual;

HACCEPTOR_COUNT('O')
--------------------
2
```

HDONOR_COUNT

Number of hydrogen-bonding donor sites as defined by the SMARTS pattern `[!#6;!H0]`. Heavy atoms may have multiple hydrogen-bonding sites.

For example, the number of hydrogen-bonding donor sites for water is 2 (one for each hydrogen):

Dayprop
```
% echo '$SMI<O>|' | dayprop -property hdonor_count
$SMI<O>PPROP<4;2;0;>|
```

Daypropталk
```
Qwerty: Set PROPERTY HDONOR_COUNT.
Qwerty: Over.
Property type set to HDONOR_COUNT...4
Qwerty: Over.
O
Qwerty: Over.
O 2
Qwerty: Over.
```

SQL
```
SQL> select hdonor_count('O') from dual;

HDONOR_COUNT('O')
-----------------
2
```

MATCH_COUNT

The number of unique matches of a user defined SMARTS in a molecule.
For example, the number of carbons in ethanol is 2:

Dayprop
```
% echo '$SMI<CCO>|' | dayprop -property match_count -smarts '[#6]'
$SMI<CCO>PPROP<19;2;0;>|
```

Daypropталk
```
Qwerty: Set PROPERTY MATCH_COUNT.
Qwerty: Over.
Property type set to MATCH_COUNT...19
Qwerty: Over.
Qwerty: Set SMARTS [#6].
```

```
Qwerty: Over.
SMARTS set to [#6]
Qwerty: Over.
CCO
Qwerty: Over.
CCO 2
Qwerty: Over.
```

SQL

```
SQL> select match_count('CCO','[#6]') from dual;

MATCH_COUNT('CCO','[#6]')
-------------------------
2
```

MOLAR_VOLUME

Average molar volume based on SchrÃ¶dinger's method. Only works for C,H,N,O,S,F,Cl,Br,I. The molar volume is the volume of one mole of compound i.e. the inverse of the density. The additive constitutive method used here is that of Schroeder. Molar volume is used in estimating interfacial tension, liquid viscosity, surface tension and water solubility:

Dayprop

```
% echo '$SMI<CCO>|' | dayprop -property molar_volume
$SMI<CCO>PPROP<8;63.00;0;>|
```

Daypropталk

```
Qwerty: Set PROPERTY MOLAR_VOLUME.
Qwerty: Over.
Property type set to MOLAR_VOLUME...8
Qwerty: Over.
CCO
Qwerty: Over.
CCO 63.00
Qwerty: Over.
```

SQL

```
SQL> select molar_volume('CCO') from dual;

MOLAR_VOLUME('CCO')
-------------------
63
```

MOL_FORM

Calculates molecular formula in Hill order. Charges are ignored:

Dayprop

```
% echo '$SMI<N[C@@](C)C(=O)O>|' | dayprop -property mol_form
$SMI<N[C@@](C)C(=O)O>PPROP<2;C3H6NO2;0;>|
```

Dayproptalk
```
Qwerty: Set PROPERTY MOL_FORM.
Qwerty: Over.
Property type set to MOL_FORM...2
Qwerty: Over.
N[C@@](C)C(=O)O
Qwerty: Over.
N[C@@](C)C(=O)O C3H6NO2
Qwerty: Over.
```

SQL
```
SQL> select mol_form('N[C@@](C)C(=O)O') from dual;

MOL_FORM('N[C@@](C)C(=O)O')
--------------------------------------------------------------------------------
C3H6NO2
```

PARACHOR

Computes molar surface tension in dynes per centimeter using McGowan's method.

In this example, the parachor of ethanol is 127.60 dyn/cm:

Dayprop
```
% echo '$SMI<CCO>|' | dayprop -property parachor
$SMI<CCO>PPROP<6;127.60;0;>|
```

Dayproptalk
```
Qwerty: Set PROPERTY PARACHOR.
Qwerty: Over.
Property type set to PARACHOR...6
Qwerty: Over.
CCO
Qwerty: Over.
CCO 127.60
Qwerty: Over.
```

SQL
```
SQL> select parachor('CCO') from dual;

PARACHOR('CCO')
---------------
127.6
```

This supported set of atoms is `C,H,N,S,P,F,Cl,Br,I`. This property is set to `No Value` for molecules that contain atoms outside the supported set.

For example, the parachor for silicon dioxide cannot be computed:

Dayprop

```
% echo '$SMI<O=[Si]=O>|' | dayprop -property parachor
$SMI<O=[Si]=O>PPROP<6;No_Value;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY PARACHOR.
Qwerty: Over.
Property type set to PARACHOR...6
Qwerty: Over.
O=[Si]=O
Qwerty: Over.
O=[Si]=O No_Value
Qwerty: Over.
```

SQL

```
SQL> select parachor('O=[Si]=O') from dual;
PARACHOR('O=[SI]=O')
--------------------
```

Parachor can be used to estimate surface tension and boiling point. Parachor is also used in estimating soil absorption coefficients, and water solubility. This version uses the atom contributions of McGowan. More complicated schema are available in the review by Quayle.

MacLeod-Sugden method for surface tension estimation:[3],[6]

$$\sigma = (P\rho_L/MW)^4$$

where

σ = surface tension (dynes.cm^{-1})
ρ_L = liquid density (g.cm^{-3})
P = parachor
MW = molecular weight (g.mol^{-1})

For aniline `c1ccccc1N` $o = 43.80$ using estimates from dayprop(), experimental value is 42.9 at 20°C (*Handbook of Chemistry and Physics* CRC Press)

For ethyl acetate `CCOC(=O)C` $o = 22.69$ using estimates from dayprop(), experimental value is 23.9 at 20°C (*Handbook of Chemistry and Physics* CRC Press)

Meissner's method for boiling point estimation:[4]

$$Tb = (637\ MR^{1.47} + B) / P$$

where

T_b = boiling point
MR = molar refractivity
B = a constant dependent on chemical class
P = parachor

For chloroethyl vinyl ether `ClCCOC=C` T_b = `371.9°K` using estimates from dayprop(), experimental value is 381.2°K (*Handbook of Chemistry and Physics* CRC Press)

For nicotine `c1ccc(C2N(C)CCC2)cn1` T_b = `493.0°K` using estimates from dayprop(), the experimental value is 515.7°K (*Handbook of Chemistry and Physics* CRC Press) **PART_COUNT**

Number of components.

For example, the number of components in a mixture of ethanol and water is 2:

Dayprop

```
% echo '$SMI<CCO.O>|' | dayprop -property part_count
$SMI<CCO.O>PPROP<16;2;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY PART_COUNT.
Qwerty: Over.
Property type set to PART_COUNT...16
Qwerty: Over.
CCO.O
Qwerty: Over.
CCO.O 2
Qwerty: Over.
```

SQL

```
SQL> select part_count('CCO.O') from dual;

PART_COUNT('CCO.O')
-------------------
2
```

POLAR_SURFACE_AREA

Compute the topological polar surface area (TPSA) according to the method of Ertl, Rohde, and Selzer.[1]

For example, the TPSA for ethanol is 20.23:

Dayprop

```
% echo '$SMI<CCO>|' | dayprop -property polar_surface_area
$SMI<CCO>PPROP<17;20.23;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY POLAR_SURFACE_AREA.
Qwerty: Over.
Property type set to POLAR_SURFACE_AREA...17
Qwerty: Over.
CCO
Qwerty: Over.
CCO 20.23
Qwerty: Over.
```

SQL

```
SQL> select polar_surface_area('CCO') from dual;

POLAR_SURFACE_AREA('CCO')
-------------------------
20.23
```

RIGIDITY

Compute the Tanimoto similarity value between a molecule and a hypothetical version of itself with the rotatable bonds removed. Values range from 1 (rigid) to 0 (not rigid).

For example, the rigidity value of hexane is 0.5833.

Dayprop

```
% echo '$SMI<CCCCCC>|' | dayprop -property rigidity
$SMI<CCCCCC>PPROP<10;0.5833;0;>|
```

Dayproptalk

```
Qwerty: Set PROPERTY RIGIDITY.
Qwerty: Over.
Property type set to RIGIDITY...10
Qwerty: Over.
CCCCCC
Qwerty: Over.
CCCCCC 0.5833
```

SQL

```
SQL> select rigidity('CCCCCC') from dual;

RIGIDITY('CCCCCC')
------------------
.5833
```

Note: Path lengths greater than 7 are not recognized. Rings with 8 or more atoms are not perceived.

RING_COUNT

Number of smallest set of smallest rings (SSSR).

For example, cubane has a SSSR of 5:

Dayprop

```
% echo '$SMI<C12C3C4C1C1C4C3C21>|' | dayprop -property ring_count
$SMI<C12C3C4C1C1C4C3C21>PPROP<9;5;0;>|
```

Daypropталk

```
Qwerty: Set PROPERTY RING_COUNT.
Qwerty: Over.
Property type set to RING_COUNT...9
Qwerty: Over.
C12C3C4C1C1C4C3C21
Qwerty: Over.
C12C3C4C1C1C4C3C21 5
Qwerty: Over.
```

SQL

```
SQL> select ring_count('C12C3C4C1C1C4C3C21') from dual;

RING_COUNT('C12C3C4C1C1C4C3C21')
--------------------------------
5
```

ROTBOND_COUNT:

Number of rotatable bonds using the SMARTS pattern: `[!$(*#*)&!D1&$(*(-[!#1])~[!#1])]-&!@[!$(*#*)&!D1&$(*(-[!#1])~[!#1])]`. It matches acyclic bonds between two atoms that have additional non-Hydrogen substituents and that are not alkynes. Secondary amides are further excluded with the pattern `([N&H1&D2]-&!@[#6&X3])`.

For example, the number of rotatable bonds in alanine is 1:

Dayprop

```
% echo '$SMI<N[C@@](C)C(=O)O>|' | dayprop -property rotbond_count
$SMI<N[C@@](C)C(=O)O>PPROP<3;1;0;>|
```

Daypropталk

```
Qwerty: Set PROPERTY ROTBOND_COUNT.
Qwerty: Over.
Property type set to ROTBOND_COUNT...3
Qwerty: Over.
N[C@@](C)C(=O)O
Qwerty: Over.
N[C@@](C)C(=O)O 1
Qwerty: Over.
```

SQL

```
SQL> select rotbond_count('N[C@@](C)C(=O)O') from dual;

ROTBOND_COUNT('N[C@@](C)C(=O)O')
--------------------------------
1
```

Note: Symmetrical tri-substituted groups are considered rotatable (ex., N-trimethyl anilinium cation, `[N+](C)(C)(C)c1ccccc1`) and symmetrical non-substituted groups are not (ex., `[N+]([H])([H])([H])c1ccccc1`).
Bonds to explicit (isotopic) Hydrogens are not counted as needed substituents on rotatable bonds. CCC[2H] and CCC both have zero rotatable bonds.
Amidines `(C=C(N)N)` are not recognized rotatable (for example: the carbon-carbon double bond in Zantac, `CN/C(=C\[N+](=O)[O-])/NCCSCc1ccc(CN(C)C)o1`, rotates on the NMR time scale).

Triple bonds and the two adjacent single bonds are not recognized as rotatable bonds.

Sulphonamides `(NS(=O)*)` are not recognized as rotatable (the double bond character of the N-S bond is questionable).

Some other groups not considered rotatable as a unit include adamantyl, barrelenes, propelleranes, and extended cumulenes. **STEREOCENTER_COUNT**

The number of stereogenic centers using the following SMARTS patterns. necessary but not sufficient conditions for stereo:

Atom stereo:	`[$([X4&!v6&!v5;H0,H1]),$([SX3]([#6])([#6])~O)]`
Bond stereo:	`[CX3;!H2]=[CX3;!H2]`
Allene stereo:	`[CX3;H0]=C=[CX3;H0,H1]`

For example, the number of stereogenic centers in alanine is 1:

Dayprop

```
% echo '$SMI<N[C@@H](C)C(=O)O>|' | dayprop -property stereocenter_count
$SMI<N[C@@H](C)C(=O)O>PPROP<15;1;0;>|
```

Daypropталk

```
Qwerty: Set PROPERTY STEREOCENTER_COUNT.
Qwerty: Over.
Property type set to STEREOCENTER_COUNT...15
Qwerty: Over.
N[C@@H](C)C(=O)O
Qwerty: Over.
N[C@@H](C)C(=O)O 1
Qwerty: Over.
```

SQL

```
SQL> select stereocenter_count('N[C@@H](C)C(=O)O') from dual;

STEREOCENTER_COUNT('N[C@@H](C)C(=O)O')
```

```
------------------------------------
1
```

8. Appendix: References

1. Ertl, P.; Rohde, B." Selzer P., "Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment-based Contributions and Its Application to the Prediction of Drug Transport Properties", *J.Med.Chem.*(2000), **43**, 371 4- 3717.

2. Katritzky, A. R.; Lobanov, V. S.; Karelson, M. QSPR: The correlation and quantitative prediction of chemical and physical properties from structure. *Chem. Soc. Rev.* **1995**, 24, 279-287

3. MacLeod, D.B. "On a Relation between Surface Tension and Density" *Trans. Faraday Soc.* **19** 384-42 (1923)

4. Meissner H.P., "Critical Constants from Parachor and Molar Refraction" *Chem. Eng. Prog.* **45** 149-153 (1949)

5. Reid, R.C.; Prausnitz J.M.; and Poling, B.E. *The Properties of Liquids and Gases*, 4th ed., New York: McGraw-Hill Book Company (1987). As an historical note this is identical to the example in the Medchem 3.41 GCL manual

6. Sugden, S., "The Influence of the Orientation of Surface Molecules on the Surface Tension of Pure Liquids" *J. Chem Soc.* **125** 1167-89 (1925)