NAME

MolecularComplexityDescriptors

SYNOPSIS

use MolecularDescriptors::MolecularComplexityDescriptors;

use MolecularDescriptors::MolecularComplexityDescriptors qw(:all);

DESCRIPTION

MolecularComplexityDescriptors class provides the following methods:

new, GenerateDescriptors, GetDescriptorNames, GetMolecularComplexityTypeAbbreviation, MACCSKeysSize, SetAtomIdentifierType, SetAtomicInvariantsToUse, SetDistanceBinSize, SetFunctionalClassesToUse, SetMaxDistance, SetMaxPathLength, SetMinDistance, SetMinPathLength, SetMolecularComplexityType, SetNeighborhoodRadius, SetNormalizationMethodology, StringifyMolecularComplexityDescriptors

MolecularComplexityDescriptors is derived from MolecularDescriptors class which in turn is derived from ObjectProperty base class that provides methods not explicitly defined in MolecularComplexityDescriptors, MolecularDescriptors or ObjectProperty classes using Perl's AUTOLOAD functionality. These methods are generated on-the-fly for a specified object property:

```
    Set<PropertyName>(<PropertyValue>);
    $PropertyValue = Get<PropertyName>();
    Delete<PropertyName>();
```

The current release of MayaChemTools supports calculation of molecular complexity using *MolecularComplexityType* parameter corresponding to number of bits-set or unique keys [Ref 117-119] in molecular fingerprints. The valid values for *MolecularComplexityType* are:

```
    AtomTypesFingerprints
    ExtendedConnectivityFingerprints
    MACCSKeys
    PathLengthFingerprints
    TopologicalAtomPairsFingerprints
    TopologicalAtomTripletsFingerprints
    TopologicalAtomTorsionsFingerprints
    TopologicalPharmacophoreAtomPairsFingerprints
    TopologicalPharmacophoreAtomTripletsFingerprints
```

Default value for *MolecularComplexityType*: *MACCSKeys*.

AtomIdentifierType parameter name corresponds to atom types used during generation of fingerprints. The valid values for *AtomIdentifierType* are: *AtomicInvariantsAtomTypes, DREIDINGAtomTypes, EStateAtomTypes, FunctionalClassAtomTypes, MMFF94AtomTypes, SLogPAtomTypes, SYBYLAtomTypes, TPSAAtomTypes, UFFAtomTypes*. *AtomicInvariantsAtomTypes* is not supported for following values of *MolecularComplexityType*: *MACCSKeys, TopologicalPharmacophoreAtomPairsFingerprints, TopologicalPharmacophoreAtomTripletsFingerprints*. *FunctionalClassAtomTypes* is the only valid value of *AtomIdentifierType* for topological pharmacophore fingerprints.

Default value for *AtomIdentifierType*: *AtomicInvariantsAtomTypes* for all fingerprints; *FunctionalClassAtomTypes* for topological pharmacophore fingerprints.

AtomicInvariantsToUse parameter name and values are used during *AtomicInvariantsAtomTypes* value of parameter *AtomIdentifierType* . It's a list of space separated valid atomic invariant atom types.

Possible values for atomic invariants are: *AS, X, BO, LBO, SB, DB, TB, H, Ar, RA, FC, MN, SM*. Default value for *AtomicInvariantsToUse* parameter are set differently for different fingerprints using *MolecularComplexityType* parameter as shown below:

```
    MolecularComplexityType              AtomicInvariantsToUse


    AtomTypesFingerprints                AS X BO H FC
    TopologicalAtomPairsFingerprints     AS X BO H FC
    TopologicalAtomTripletsFingerprints  AS X BO H FC
    TopologicalAtomTorsionsFingerprints  AS X BO H FC


    ExtendedConnectivityFingerprints     AS X  BO H FC MN
    PathLengthFingerprints               AS
```

FunctionalClassesToUse parameter name and values are used during *FunctionalClassAtomTypes* value of parameter *AtomIdentifierType*. It's a list of space separated valid atomic invariant atom types.

Possible values for atom functional classes are: *Ar, CA, H, HBA, HBD, Hal, NI, PI, RA*.

Default value for *FunctionalClassesToUse* parameter is set to:

```
    HBD HBA PI NI Ar Hal
```

for all fingerprints except for the following two *MolecularComplexityType* fingerints:

```
    MolecularComplexityType                            FunctionalClassesToUse

    TopologicalPharmacophoreAtomPairsFingerprints     HBD HBA P, NI H
    TopologicalPharmacophoreAtomTripletsFingerprints  HBD HBA PI NI H Ar
```

MACCSKeysSize parameter name is only used during *MACCSKeys* value of *MolecularComplexityType* and corresponds to size of MACCS key set. Possible values: *166 or 322*. Default value: *166*.

NeighborhoodRadius parameter name is only used during *ExtendedConnectivityFingerprints* value of *MolecularComplexityType* and corresponds to atomic neighborhoods radius for generating extended connectivity fingerprints. Possible values: positive integer. Default value: *2*.

MinPathLength and *MaxPathLength* parameters are only used during *PathLengthFingerprints* value of *MolecularComplexityType* and correspond to minimum and maximum path lengths to use for generating path length fingerprints. Possible values: positive integers. Default value: *MinPathLength - 1*; *MaxPathLength - 8*.

UseBondSymbols parameter is only used during *PathLengthFingerprints* value of *MolecularComplexityType* and indicates whether bond symbols are included in atom path strings used to generate path length fingerprints. Possible value: *Yes or No*. Default value: *Yes*.

MinDistance and *MaxDistance* parameters are only used during *TopologicalAtomPairsFingerprints* and *TopologicalAtomTripletsFingerprints* values of *MolecularComplexityType* and correspond to minimum and maximum bond distance between atom pairs during topological pharmacophore fingerprints. Possible values: positive integers. Default value: *MinDistance - 1*; *MaxDistance - 10*.

UseTriangleInequality parameter is used during these values for *MolecularComplexityType*: *TopologicalAtomTripletsFingerprints* and *TopologicalPharmacophoreAtomTripletsFingerprints*. Possible values: *Yes or No*. It determines wheter to apply triangle inequality to distance triplets. Default value: *TopologicalAtomTripletsFingerprints - No*; *TopologicalPharmacophoreAtomTripletsFingerprints - Yes*.

DistanceBinSize parameter is used during *TopologicalPharmacophoreAtomTripletsFingerprints* value of *MolecularComplexityType* and corresponds to distance bin size used for binning distances during generation of topological pharmacophore atom triplets fingerprints. Possible value: positive integer. Default value: *2*.

NormalizationMethodology is only used for these values for *MolecularComplexityType*: *ExtendedConnectivityFingerprints*, *TopologicalPharmacophoreAtomPairsFingerprints* and *TopologicalPharmacophoreAtomTripletsFingerprints*. It corresponds to normalization methodology to use for scaling the number of bits-set or unique keys during generation of fingerprints. Possible values during *ExtendedConnectivityFingerprints*: *None or ByHeavyAtomsCount*; Default value: *None*. Possible values during topological pharmacophore atom pairs and triplets fingerprints: *None or ByPossibleKeysCount*; Default value: *None*. *ByPossibleKeysCount* corresponds to total number of possible topological pharmacophore atom pairs or triplets in a molecule.

METHODS

new

```
    $NewMolecularComplexityDescriptors = new MolecularDescriptors::
                                         MolecularComplexityDescriptors(
                                             %NamesAndValues);
```

Using specified *MolecularComplexityDescriptors* property names and values hash, new method creates a new object and returns a reference to newly created MolecularComplexityDescriptors object. By default, the following properties are initialized:

```
    Molecule = ''
    Type = 'MolecularComplexity'
    MolecularComplexityType = 'MACCSKeys'
    AtomIdentifierType = ''
    MACCSKeysSize = 166
    NeighborhoodRadius = 2
    MinPathLength = 1
    MaxPathLength = 8
    UseBondSymbols = 1
    MinDistance = 1
    MaxDistance = 10
    UseTriangleInequality = ''
    DistanceBinSize = 2
    NormalizationMethodology = 'None'
    @DescriptorNames = ('MolecularComplexity')
    @DescriptorValues = ('None')
```

Examples:

```
    $MolecularComplexityDescriptors = new MolecularDescriptors::
                                      MolecularComplexityDescriptors(
                                      'Molecule' => $Molecule);

    $MolecularComplexityDescriptors = new MolecularDescriptors::
                                      MolecularComplexityDescriptors();

    $MolecularComplexityDescriptors->SetMolecule($Molecule);
    $MolecularComplexityDescriptors->GenerateDescriptors();
    print "MolecularComplexityDescriptors: $MolecularComplexityDescriptors\n";
```

GenerateDescriptors

```
$MolecularComplexityDescriptors->GenerateDescriptors();
```

Calculates MolecularComplexity value for a molecule and returns *MolecularComplexityDescriptors*.

GetDescriptorNames

```
@DescriptorNames = $MolecularComplexityDescriptors->GetDescriptorNames();
@DescriptorNames = MolecularDescriptors::MolecularComplexityDescriptors::
                   GetDescriptorNames();
```

Returns all available descriptor names as an array.

GetMolecularComplexityTypeAbbreviation

```
$Abbrev = $MolecularComplexityDescriptors->
              GetMolecularComplexityTypeAbbreviation();
$Abbrev = MolecularDescriptors::MolecularComplexityDescriptors::
              GetMolecularComplexityTypeAbbreviation($ComplexityType);
```

Returns abbreviation for a specified molecular complexity type or corresponding to *MolecularComplexityDescriptors* object.

SetMACCSKeysSize

```
$MolecularComplexityDescriptors->MACCSKeysSize($Size);
```

Sets MACCS keys size and returns *MolecularComplexityDescriptors*.

SetAtomIdentifierType

```
$MolecularComplexityDescriptors->SetAtomIdentifierType($IdentifierType);
```

Sets atom *IdentifierType* to use during fingerprints generation corresponding to *MolecularComplexityType* and returns *MolecularComplexityDescriptors*.

Possible values: *AtomicInvariantsAtomTypes, DREIDINGAtomTypes, EStateAtomTypes, FunctionalClassAtomTypes, MMFF94AtomTypes, SLogPAtomTypes, SYBYLAtomTypes, TPSAAtomTypes, UFFAtomTypes*.

SetAtomicInvariantsToUse

```
$MolecularComplexityDescriptors->SetAtomicInvariantsToUse($ValuesRef);
$MolecularComplexityDescriptors->SetAtomicInvariantsToUse(@Values);
```

Sets atomic invariants to use during *AtomicInvariantsAtomTypes* value of *AtomIdentifierType* for fingerprints generation and returns *MolecularComplexityDescriptors*.

Possible values for atomic invariants are: *AS, X, BO, LBO, SB, DB, TB, H, Ar, RA, FC, MN, SM*. Default value [Ref 24]: *AS,X,BO,H,FC,MN*.

The atomic invariants abbreviations correspond to:

```
AS = Atom symbol corresponding to element symbol

X<n>   = Number of non-hydrogen atom neighbors or heavy atoms
BO<n> = Sum of bond orders to non-hydrogen atom neighbors or heavy atoms
LBO<n> = Largest bond order of non-hydrogen atom neighbors or heavy atoms
SB<n> = Number of single bonds to non-hydrogen atom neighbors or heavy atoms
DB<n> = Number of double bonds to non-hydrogen atom neighbors or heavy atoms
TB<n> = Number of triple bonds to non-hydrogen atom neighbors or heavy atoms
H<n>   = Number of implicit and explicit hydrogens for atom
Ar     = Aromatic annotation indicating whether atom is aromatic
RA     = Ring atom annotation indicating whether atom is a ring
FC<+n/-n> = Formal charge assigned to atom
MN<n> = Mass number indicating isotope other than most abundant isotope
SM<n> = Spin multiplicity of atom. Possible values: 1 (singlet), 2 (doublet) or
        3 (triplet)
```

Atom type generated by AtomTypes::AtomicInvariantsAtomTypes class corresponds to:

```
AS.X<n>.BO<n>.LBO<n>.<SB><n>.<DB><n>.<TB><n>.H<n>.Ar.RA.FC<+n/-n>.MN<n>.SM<n>
```

Except for AS which is a required atomic invariant in atom types, all other atomic invariants are optional. Atom type specification doesn't include atomic invariants with zero or undefined values.

In addition to usage of abbreviations for specifying atomic invariants, the following descriptive words are also allowed:

```
X : NumOfNonHydrogenAtomNeighbors or NumOfHeavyAtomNeighbors
BO : SumOfBondOrdersToNonHydrogenAtoms or SumOfBondOrdersToHeavyAtoms
LBO : LargestBondOrderToNonHydrogenAtoms or LargestBondOrderToHeavyAtoms
SB :  NumOfSingleBondsToNonHydrogenAtoms or NumOfSingleBondsToHeavyAtoms
DB : NumOfDoubleBondsToNonHydrogenAtoms or NumOfDoubleBondsToHeavyAtoms
TB : NumOfTripleBondsToNonHydrogenAtoms or NumOfTripleBondsToHeavyAtoms
H :  NumOfImplicitAndExplicitHydrogens
Ar : Aromatic
RA : RingAtom
```

```
FC : FormalCharge
MN : MassNumber
SM : SpinMultiplicity
```

AtomTypes::AtomicInvariantsAtomTypes module is used to assign atomic invariant atom types.

SetDistanceBinSize

```
$MolecularComplexityDescriptors->SetDistanceBinSize($BinSize);
```

Sets distance bin size used to bin distances between atom pairs in atom triplets for topological pharmacophore atom triplets fingerprints generation and returns *MolecularComplexityDescriptors*.

SetFunctionalClassesToUse

```
$MolecularComplexityDescriptors->SetFunctionalClassesToUse($ValuesRef);
$MolecularComplexityDescriptors->SetFunctionalClassesToUse(@Values);
```

Sets functional classes invariants to use during *FunctionalClassAtomTypes* value of *AtomIdentifierType* for fingerprints generation and returns *MolecularComplexityDescriptors*.

Possible values for atom functional classes are: *Ar, CA, H, HBA, HBD, Hal, NI, PI, RA*. Default value [Ref 24]: *HBD,HBA,PI,NI,Ar,Hal*.

The functional class abbreviations correspond to:

```
HBD: HydrogenBondDonor
HBA: HydrogenBondAcceptor
PI :  PositivelyIonizable
NI : NegativelyIonizable
Ar : Aromatic
Hal : Halogen
H : Hydrophobic
RA : RingAtom
CA : ChainAtom
```

```
 Functional class atom type specification for an atom corresponds to:

    Ar.CA.H.HBA.HBD.Hal.NI.PI.RA or None
```

AtomTypes::FunctionalClassAtomTypes module is used to assign functional class atom types. It uses following definitions [Ref 60-61, Ref 65-66]:

```
HydrogenBondDonor: NH, NH2, OH
HydrogenBondAcceptor: N[!H], O
PositivelyIonizable: +, NH2
NegativelyIonizable: -, C(=O)OH, S(=O)OH, P(=O)OH
```

SetMaxDistance

```
$MolecularComplexityDescriptors->SetMaxDistance($MaxDistance);
```

Sets maximum distance to use during topological atom pairs and triplets fingerprints generation and returns *MolecularComplexityDescriptors*.

SetMaxPathLength

```
$MolecularComplexityDescriptors->SetMaxPathLength($Length);
```

Sets maximum path length to use during path length fingerprints generation and returns *MolecularComplexityDescriptors*.

SetMinDistance

```
$MolecularComplexityDescriptors->SetMinDistance($MinDistance);
```

Sets minimum distance to use during topological atom pairs and triplets fingerprints generation and returns *MolecularComplexityDescriptors*.

SetMinPathLength

```
$MolecularComplexityDescriptors->SetMinPathLength($MinPathLength);
```

Sets minimum path length to use during path length fingerprints generation and returns *MolecularComplexityDescriptors*.

SetMolecularComplexityType

```
$MolecularComplexityDescriptors->SetMolecularComplexityType($ComplexityType);
```

Sets molecular complexity type to use for calculating its value and returns *MolecularComplexityDescriptors*.

SetNeighborhoodRadius

```
$MolecularComplexityDescriptors->SetNeighborhoodRadius($Radius);
```

Sets neighborhood radius to use during extended connectivity fingerprints generation and returns *MolecularComplexityDescriptors*.

SetNormalizationMethodology

```
    $MolecularComplexityDescriptors->SetNormalizationMethodology($Methodology);
```

Sets normalization methodology to use during calculation of molecular complexity corresponding to extended connectivity, topological pharmacophore atom pairs and tripletes fingerprints returns *MolecularComplexityDescriptors*.

StringifyMolecularComplexityDescriptors

```
    $String = $MolecularComplexityDescriptors->
                  StringifyMolecularComplexityDescriptors();
```

Returns a string containing information about *MolecularComplexityDescriptors* object.

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

MolecularDescriptors.pm, MolecularDescriptorsGenerator.pm

COPYRIGHT