

NAME

ExtendedConnectivityFingerprints

SYNOPSIS

```
use ExtendedConnectivityFingerprints;  
  
use ExtendedConnectivityFingerprints qw(:all);
```

DESCRIPTION

ExtendedConnectivityFingerprints [Ref 48, Ref 52] class provides the following methods:

new, GenerateFingerprints, GetDescription, SetAtomIdentifierType, SetAtomicInvariantsToUse, SetFunctionalClassesToUse, SetNeighborhoodRadius, StringifyExtendedConnectivityFingerprints

ExtendedConnectivityFingerprints is derived from Fingerprints class which in turn is derived from ObjectProperty base class that provides methods not explicitly defined in ExtendedConnectivityFingerprints, Fingerprints 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 generation of ExtendedConnectivityFingerprints corresponding to following Atomtoml identifierTypes:

```
AtomicInvariantsAtomTypes, DREIDINGAtomTypes, EStateAtomTypes,  
FunctionalClassAtomTypes, MMFF94AtomTypes, SLogPAtomTypes,  
SYBYLAtomTypes, TPSAAtomTypes, UFFAtomTypes
```

Based on the values specified for Atoml identifierType, Atomicl nvariantsToUse and FunctionalClassesToUse, initial atom types are assigned to all non-hydrogen atoms in a molecule and these atom types strings are converted into initial atom identifier integers using TextUtil::HashCode function. The duplicate atom identifiers are removed.

For NeighborhoodRadius value of 0, the initial set of unique atom identifiers comprises the molecule fingerprints. Otherwise, atom neighborhoods are generated for each non-hydrogen atom up-to specified NeighborhoodRadius value. For each non-hydrogen central atom at a specific radius, its neighbors at next radius level along with their bond orders and previously calculated atom identifiers are collected which in turn are used to generate a new integer atom identifier; the bond orders and atom identifier pairs list is first sorted by bond order followed by atom identifiers to make these values graph invariant.

After integer atom identifiers have been generated for all non-hydrogen atoms at all specified neighborhood radii, the duplicate integer atom identifiers corresponding to same hash code value generated using TextUtil::HashCode are tracked by keeping the atom identifiers at lower radius. Additionally, all structurally duplicate integer atom identifiers at each specified radius are also tracked by identifying equivalent atom and bonds corresponding to substructures used for generating atom identifier and keeping integer atom identifier with lowest value.

For *ExtendedConnectivity* value of fingerprints Type, the duplicate identifiers are removed from the list and the unique atom identifiers constitute the extended connectivity fingerprints of a molecule.

For *ExtendedConnectivityCount* value of fingerprints Type, the occurrence of each unique atom identifiers appears is counted and the unique atom identifiers along with their count constitute the extended connectivity fingerprints of a molecule.

For *ExtendedConnectivityBits* value of fingerprints -m, --mode, the unique atom identifiers are used as a random number seed to generate a random integer value between 0 and --Size which in turn is used to set corresponding bits in the fingerprint bit-vector string.

The current release of MayaChemTools generates the following types of extended connectivity fingerprints vector strings:

```
FingerprintsVector;ExtendedConnectivity:AtomicInvariantsAtomTypes:Radi  
us2;60;AlphaNumericalValues;ValuesString;73555770 333564680 352413391
```



```
FingerprintsVector;ExtendedConnectivity:MMFF94AtomTypes:Radius2;64;AlphaNumericalValues;ValuesString;224051550 746527773 998750766 103704190
2 1239701709 1248384926 1259447756 1521678386 1631549126 1909437580 20
37095052 2104274756 2117729376 8770364 31445800 81450228 314289324 344
041929 581773587 638555787 692022098 811840536 929651561 936421792 988
636432 1048624296 1054288509 1369487579 1454058929 1519352190 17271...
```

```
FingerprintsVector;ExtendedConnectivity:SLogPAtomTypes:Radius2;71;AlphaNumericalValues;ValuesString;78989290 116507218 489454042 888737940 1
162561799 1241797255 1251494264 1263717127 1471206899 1538061784 17654
07295 1795036542 1809833874 2020454493 2055310842 2117729376 11868981
56731842 149505242 184525155 196984339 288181334 481409282 556716568 6
41915747 679881756 721736571 794256218 908276640 992898760 10987549...
```

```
FingerprintsVector;ExtendedConnectivity:SYBYLAtomTypes:Radius2;58;AlphaNumericalValues;ValuesString;199957044 313356892 455463968 465982819
1225318176 1678585943 1883366064 1963811677 2117729376 113784599 19153
8837 196629033 263865277 416380653 477036669 681527491 730724924 90906
5537 1021959189 1133014972 1174311016 1359441203 1573452838 1661585138
1668649038 1684198062 1812312554 1859266290 1891651106 2072549404 ...
```

```
FingerprintsVector;ExtendedConnectivity:TPSAAtomTypes:Radius2;47;AlphaNumericalValues;ValuesString;20818206 259344053 862102353 1331904542 1
700688206 265614156 363161397 681332588 810600886 885767127 950172500
951454814 1059668746 1247054493 1382302230 1399502637 1805025917 19189
39561 2114677228 2126402271 8130483 17645742 32278373 149975755 160327
654 256360355 279492740 291251259 317592700 333763396 972105960 101...
```

```
FingerprintsVector;ExtendedConnectivity:UFFAtomTypes:Radius2;56;AlphaNumericalValues;ValuesString;280305427 357928343 721790579 1151822898 1
207111054 1380963747 1568213839 1603445250 4559268 55012922 180940813
335715751 534801009 684609658 829361048 972945982 999881534 1007655741
1213692591 1222032501 1224517934 1235687794 1244268533 1528120700 162
9595024 1856308891 1978806036 2001865095 2096549435 172675415 18344...
```

METHODS

new

```
$NewExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    %NamesAndValues);
```

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

```
Molecule = ''
Type = 'ExtendedConnectivity'
NeighborhoodRadius = 2
AtomIdentifierType = ''
AtomicInvariantsToUse = ['AS', 'X', 'BO', 'H', 'FC', 'MN']
FunctionalClassesToUse = ['HBD', 'HBA', 'PI', 'NI', 'Ar', 'Hal']
```

Examples:

```
$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Molecule' => $Molecule,
    'AtomIdentifierType' =>
        'AtomicInvariantsAtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivityCount',
    'Molecule' => $Molecule,
    'AtomIdentifierType' =>
```

```
'AtomicInvariantsAtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivityBits',
    'Molecule' => $Molecule,
    'Size' => 1024,
    'AtomIdentifierType' =>
        'AtomicInvariantsAtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivity',
    'Molecule' => $Molecule,
    'NeighborhoodRadius' => 2,
    'AtomIdentifierType' =>
        'AtomicInvariantsAtomTypes',
    'AtomicInvariantsToUse' =>
        ['AS', 'X', 'BO', 'H', 'FC', 'MN'] );

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivity',
    'Molecule' => $Molecule,
    'NeighborhoodRadius' => 2,
    'AtomIdentifierType' =>
        'FunctionalClassAtomTypes',
    'FunctionalClassesToUse' =>
        ['HBD', 'HBA', 'PI', 'NI', 'Ar', 'Hal'] );

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivity',
    'Molecule' => $Molecule,;
    'AtomIdentifierType' =>
        'MMFF94AtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivityCount',
    'Molecule' => $Molecule,;
    'AtomIdentifierType' =>
        'MMFF94AtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivityCount',
    'Molecule' => $Molecule,;
    'AtomIdentifierType' =>
        'SLogPAtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivity',
    'Molecule' => $Molecule,;
    'AtomIdentifierType' =>
        'SLogPAtomTypes');

$ExtendedConnectivityFingerprints = new ExtendedConnectivityFingerprints(
    'Type' => 'ExtendedConnectivity',
    'Molecule' => $Molecule,;
    'AtomIdentifierType' =>
        'SYBYLAtomTypes');

$ExtendedConnectivityFingerprints->GenerateFingerprints();
print "$ExtendedConnectivityFingerprints\n";
```

GenerateFingerprints

```
$ExtendedConnectivityFingerprints->GenerateFingerprints();
```

Generates extended connectivity fingerprints and returns *ExtendedConnectivityFingerprints*.

GetDescription

```
$Description = $ExtendedConnectivityFingerprints->GetDescription();
```

Returns a string containing description of extended connectivity fingerprints fingerprints.

SetAtomIdentifierType

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

Sets atom *IdentifierType* to use during extended connectivity fingerprints generation and returns *ExtendedConnectivityFingerprints*.

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

SetAtomicInvariantsToUse

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

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

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.

SetFunctionalClassesToUse

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

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

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
```

SetNeighborhoodRadius

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

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

StringifyExtendedConnectivityFingerprints

```
$String = $Fingerprints->StringifyExtendedConnectivityFingerprints();
```

Returns a string containing information about *ExtendedConnectivityFingerprints* object.

AUTHOR

Manish Sud <msud@san.rr.com>

SEE ALSO

Fingerprints.pm, FingerprintsStringUtil.pm, AtomNeighborhoodsFingerprints.pm, AtomTypesFingerprints.pm, EStateIndicesFingerprints.pm, MACCSKeys.pm, PathLengthFingerprints.pm, TopologicalAtomPairsFingerprints.pm, TopologicalAtomTripletsFingerprints.pm, TopologicalAtomTorsionsFingerprints.pm, TopologicalPharmacophoreAtomPairsFingerprints.pm, TopologicalPharmacophoreAtomTripletsFingerprints.pm

COPYRIGHT

Copyright (C) 2004-2012 Manish Sud. All rights reserved.

This file is part of MayaChemTools.

MayaChemTools is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 3 of the License, or (at your option) any later version.