

## NAME

AtomTypesFingerprints

## SYNOPSIS

```
use AtomTypesFingerprints;

use AtomTypesFingerprints qw(:all);
```

## DESCRIPTION

AtomTypesFingerprints class provides the following methods:

new, GenerateFingerprints, GetDescription, SetAtomIdentifierType, SetAtomTypesSetToUse, SetAtomicInvariantsToUse, SetFunctionalClassesToUse, SetType, StringifyAtomTypesFingerprints

AtomTypesFingerprints is derived from Fingerprints class which in turn is derived from ObjectProperty base class that provides methods not explicitly defined in AtomNeighborhoodsFingerprints, 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 AtomTypesFingerprints corresponding to following AtomIdentifierTypes:

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

Based on the values specified for AtomIdentifierType along with other specified parameters such as AtomicInvariantsToUse and FunctionalClassesToUse, initial atom types are assigned to all non-hydrogen atoms or all atoms in a molecule.

Using the assigned atom types and specified Type, one of the following types of fingerprints are generated:

```
AtomTypesCount - A vector containing count of atom types
AtomTypesBits - A bit vector indicating presence/absence of atom types
```

For *AtomTypesCount* fingerprints, two types of atom types set size is allowed:

```
ArbitrarySize - Corresponds to only atom types detected in molecule
FixedSize - Corresponds to fixed number of atom types previously defined
```

For *AtomTypesBits* fingerprints, only *FixedSize* atom type set is allowed.

*ArbitrarySize* corresponds to atom types detected in a molecule where as *FixedSize* implies a fix number of all possible atom types previously defined for a specific *AtomIdentifierType*.

Fix number of all possible atom types for supported *AtomIdentifierTypes* in current release of MayaChemTools are:

AtomIdentifier	Total	TotalWithoutHydrogens
DREIDINGAtomTypes	37	34
EStateAtomTypes	109	87
MMFF94AtomTypes	212	171
SLogPAtomTypes	72	67
SYBYLAtomTypes	45	44
TPSAAtomTypes	47	47
UFFAtomTypes	126	124

Combination of Type and AtomTypesSetToUse along with AtomIdentifierType allows generation of following different atom types fingerprints:

Type	AtomIdentifierType	AtomTypesSetToUse
AtomTypesCount	AtomicInvariantsAtomTypes	ArbitrarySize
AtomTypesCount	DREIDINGAtomTypes	ArbitrarySize
AtomTypesCount	DREIDINGAtomTypes	FixedSize
AtomTypesBits	DREIDINGAtomTypes	FixedSize
AtomTypesCount	EStateAtomTypes	ArbitrarySize
AtomTypesCount	EStateAtomTypes	FixedSize
AtomTypesBits	EStateAtomTypes	FixedSize





```

Molecule = '';
Type = ''
AtomIdentifierType = ''
AtomTypesSetToUse = ''
IgnoreHydrogens = 1
AtomicInvariantsToUse = ['AS', 'X', 'BO', 'H', 'FC', 'MN']
FunctionalClassesToUse = ['HBD', 'HBA', 'PI', 'NI', 'Ar', 'Hal']

```

## Examples:

```

$AtomTypesFingerprints = new AtomTypesFingerprints(
    'Molecule' => $Molecule,
    'Type' => 'AtomTypesCount',
    'AtomIdentifierType' =>
        'AtomicInvariantsAtomTypes');

$AtomTypesFingerprints = new AtomTypesFingerprints(
    'Molecule' => $Molecule,
    'Type' => 'AtomTypesCount',
    'AtomIdentifierType' =>
        'AtomicInvariantsAtomTypes',
    'AtomicInvariantsToUse' =>
        ['AS', 'X', 'BO', 'H', 'FC'] );

$AtomTypesFingerprints = new AtomTypesFingerprints(
    'Molecule' => $Molecule,
    'Type' => 'AtomTypesCount',
    'AtomIdentifierType' =>
        'DREIDINGAtomTypes');

$AtomTypesFingerprints = new AtomTypesFingerprints(
    'Molecule' => $Molecule,
    'Type' => 'AtomTypesCount',
    'AtomIdentifierType' =>
        'EStateAtomTypes',
    'AtomTypesSetToUse' =>
        'ArbitrarySize');

$AtomTypesFingerprints = new AtomTypesFingerprints(
    'Molecule' => $Molecule,
    'Type' => 'AtomTypesCount',
    'AtomIdentifierType' =>
        'SLogPAtomTypes',
    'AtomTypesSetToUse' =>
        'FixedSize');

$AtomTypesFingerprints = new AtomTypesFingerprints(
    'Molecule' => $Molecule,
    'Type' => 'AtomTypesBits',
    'AtomIdentifierType' =>
        'MMFF94AtomTypes',
    'AtomTypesSetToUse' =>
        'FixedSize');

$AtomTypesFingerprints->GenerateFingerprints();
print "$AtomTypesFingerprints\n";

```

## GenerateFingerprints

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

Generates atom types fingerprints and returns *AtomTypesFingerprints*.

## GetDescription

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

Returns a string containing description of atom types fingerprints.

## SetAtomIdentifierType

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

Sets atom *IdentifierType* to use during atom types fingerprints generation and returns *AtomTypesFingerprints*.

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

## SetAtomTypesSetToUse

```
$AtomTypesFingerprints->SetAtomTypesSetToUse($Value);
```

Sets *Value* of *AtomTypesSetToUse* and returns *AtomTypesFingerprints*. Possible values: *ArbitrarySize* or *FixedSize*. Default for *AtomTypesCount* value of *AtomTypesSetToUse*: *ArbitrarySize*.

#### SetAtomicInvariantsToUse

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

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

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

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

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

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

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

### SetType

```
$AtomTypesFingerprints->SetType($Type);
```

Sets type of AtomTypes fingerprints and returns *AtomTypesFingerprints*. Possible values: *AtomTypesFingerprintsBits* or *AtomTypesFingerprintsCount*.

### StringifyAtomTypesFingerprints

```
$String = $AtomTypesFingerprints->StringifyAtomTypesFingerprints();
```

Returns a string containing information about *AtomTypesFingerprints* object.

### AUTHOR

Manish Sud <msud@san.rr.com>

### SEE ALSO

Fingerprints.pm, FingerprintsStringUtil.pm, AtomNeighborhoodsFingerprints.pm, EStateIndiciesFingerprints.pm, ExtendedConnectivityFingerprints.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.