Functions by Categories

Contents

Charge Functions

Name	License	Description	Return value	Parameter

atomicPolarizability atomPol pol polarizability	Structural Calculations License	calculates atomic polarizability	the polarizability values	• the atom MolAton • the majo pH (take molecul omitted)
averagePolarizability averagePol avgPol	Structural Calculations License	calculates average molecular polarizability component considering 3D geometry	the polarizability value	• the majo pH (take molecul omitted)

axxPol	Structural Calculations License	calculates principal component of polarizability tensor (a(xx), a (yy), a(zz))	the principal component of polarizability tensor	• the majo pH (take molecul omitted)
ayyPol	Structural Calculations License	calculates principal component of polarizability tensor (a(xx), a (yy), a(zz))	the principal component of polarizability tensor	• the majo pH (take molecul omitted)

azzPol	Structural Calculations License	calculates principal component of polarizability tensor (a(xx), a (yy), a(zz))	the principal component of polarizability tensor	• the majo pH (take molecul omitted)

charge	Structural Calculations License	calculates partial charges on atoms for result types "aromaticsystem" / "aromaticring", calculates the sum of partial charges of the atoms in the aromatic system / smallest aromatic ring containing the atom	the charge values	• the atom MolAtom • the resu (default) "aromati "aromati "aromati "aromati "aromati "aromati • the majo pH (take molecul omitted)

molecularPolarizability molPol	Structural Calculations License	calculates molecular polarizability	the polarizability value	• the majo pH (take molecul omitted)
piOrbitalElectronegativity pOEN	Structural Calculations License	calculates atomic pi orbital electronegativity	the pi orbital electronegativity values	• the atom MolAton • the majo pH (take molecul omitted)

resonantCharge	Structural Calculations License	calculates partial charges on atoms considering resonance effect for result types "aromaticsystem" / "aromaticring", calculates the sum of partial charges of the atoms in the aromatic system / smallest aromatic ring containing the atom	the charge values	• the atom MolAton • the resu (default) "aromati "aromati "aromati "aromati "aromati "aromati • the majo pH (take molecul omitted)

sigmaOrbitalElectronegativitysOEN	Structural Calculations License	calculates atomic sigma orbital electronegativity	the sigma orbital electronegativity values	• the atom MolAtom • the majo pH (take molecule omitted)

Back to Contents

Conformation Functions

Name	License	Description	Return value	Parameters	Exampl

conformer	Structural Calculations License	calculates a conformer of the molecule	the conformer	• the conformer index (0-based)	Molecul conform first con molecul Reactio conform returns conform reactan
conformerCount	Structural Calculations License	returns the number of calculated conformers	the number of calculated conformers	-	Molecul conform the num conform molecul Reactio conform (0)) retu calculat the first
conformers	Structural Calculations License	calculates conformers of the molecule (maximum number of conformers to be calculated can be set, default: 100)	the conformer array	-	Molecul conform conform molecul Reactio conform returns first rea

hasValidConformer	Structural Calculations License	returns true if the input molecule exist in 3D space (has a valid conformer)	true if the input molecule exist in 3D space	-	Molecul hasVali returns molecul space (l conform Reactio hasVali (0)) retu first rea space h (produc the if th exist in
lowestEnergyConformer leconformer	Structural Calculations License	calculates the lowest energy conformer of the molecule	the lowest energy conformer	-	Molecul lowestE returns conform molecul Reactio lowestE (reactar lowest e the first lowestE (produc lowest e the seco

mmff94OptimizedStructure	Structural Calculations License	calculates the MMFF94 optimized lowest energy conformer	the MMFF94 optimized lowest energy conformer	-	Molecul mmff94 returns optimize conform molecul Reactio mmff94 (reactar MMFF9 energy first rea mmff94 (produc MMFF9 energy second

Conformation Functions

Name	License	Description	Return value	Parameters	Exampl
conformer	Structural Calculations License	calculates a conformer of the molecule	the conformer	• the conformer index (0-based)	Molecul conform first con molecul Reactio conform returns conform reactan
conformerCount	Structural Calculations License	returns the number of calculated conformers	the number of calculated conformers	-	Molecul conform the num conform molecul Reactio conform (0)) retu calculat the first

conformers	Structural Calculations License	calculates conformers of the molecule (maximum number of conformers to be calculated can be set, default: 100)	the conformer array	-	Molecul conform conform molecul Reactio conform returns first rea
hasValidConformer	Structural Calculations License	returns true if the input molecule exist in 3D space (has a valid conformer)	true if the input molecule exist in 3D space	-	Molecul hasVali returns molecul space (l conform Reactio hasVali (0)) retu first rea space h (produc the if th exist in
lowestEnergyConformer leconformer	Structural Calculations License	calculates the lowest energy conformer of the molecule	the lowest energy conformer	-	Molecul lowestE returns conform molecul Reactio lowestE (reactar lowest e the first lowestE (produc lowest e the seco

mmff94OptimizedStructure	Structural Calculations License	calculates the MMFF94 optimized lowest energy conformer	the MMFF94 optimized lowest energy conformer	-	Molecul mmff94(returns optimize conform molecul Reactio mmff94((reactar MMFF9 energy first rea mmff94((produc MMFF9 energy second

Back to Contents

Dissimilarity Functions

Name	License	Description	Return value	Parameters	Examples
dissimilarity	-		the dissimilarity value	descriptor: metric or descriptor (with default metric) (optional, chemical fingerprint with Tanimoto metric is taken by default), one or two molecules (if only one is specified then the other one is taken from the context)	Molecule Context dissimilarity ("PF", "c1ccccc1", "C1CCCCC1") returns the dissimilarity value between the benzene ring and cyclohexane, computed with pharmacophore fingerprint and its default metric (Tanimoto) dissimilarity ("c1ccccc1", "C1CCCCC1") returns the dissimilarity value between

		computes the dissimilarity value between two molecules **Note 1:** Dissimilarity function requires JChem. **Note 2:** The dissimilarity values returned by dissimilarity function and similarity search in JChem database may differ because of the applied fingerprints. Dissimilarity function uses the default options of the chemical fingerprints (length=1024, bonds=7, bits=2) optimized for similarity search. Database similarity search uses fingerprints specified during the generation of the given table, optimized for substructure (and related) search types.			the benzene ring and cyclohexane, computed with default fingerprint and its default metric (chemical fingerprint with Tanimoto) dissimilarity("PF: Euclidean", "c1ccccc1") returns the dissimilarity value between the benzene ring and the input molecule, computed with pharmacophore fingerprint and Euclidean metric dissimilarity ("LogD", "c1ccccc1") returns the dissimilarity value between the benzene ring and the input molecule, computed with the LogD descriptor and its default AbsDiff metric Reaction Context dissimilarity("CF: Euclidean", "c1ccccc1", reactant(0)) returns the dissimilarity value between the benzene ring and the first reactant, computed with chemical

					fingerprint and Euclidean metric dissimilarity (reactant(0), product(0)) returns the dissimilarity value between the first reactant and the first product, computed with default fingerprint and its default metric (chemical fingerprint with Tanimoto)

Dissimilarity descriptors and metrics

Descriptor	**Metric**
ChemicalFingerprint (or CF)	Tanimoto (default) Euclidean
PharmacophoreFingerprint (or PF)	Tanimoto (default) Euclidean
ECFP	Tanimoto (default) Euclidean
Burden eigenvalue descriptor (or BCUT) (BCUT is a trademark of Tripos, Inc., used with permission)	Euclidean
HDon	Euclidean
HAcc	AbsDiff
Heavy	AbsDiff
LogD	AbsDiff
LogP	AbsDiff
Mass	AbsDiff
TPSA	AbsDiff

Back to Contents

Elemental Analysis Functions

Name	License	Description	Return value	Parameters
atomCount	-	calculates the number of atoms (all atoms or specific atoms)	the atom count	• atomic nu number (separate number c the numb mass nur atoms ar number i isotope a
composition	-	returns the composition	the composition	-
dotDisconnectedFormula	-	returns the dot-disconnected formula	the dot-disconnected formula	-
dotDisconnectedIsotopeFormula	-	returns the dot-disconnected isotope formula	the dot-disconnected isotope formula	-
dotDisconnectedFormulaWithGrouping	-	returns the grouped dot-disconnected formula	the grouped dot-disconnected formula	-

elementalAnalysis	-	performs elemental analysis on molecule represented by formula	the value calculated by the invoked function (e. g., mass)	• molecula
				• available
				  • atomC
				  • mass
				  • massF
				  • exactM
				  • exactM
				  • formu
				  • isotop
				  • dotDis
				  • dotDis
				  • group

exactMass	-	calculates the exact mass of the molecule	the exact mass	-
formula	-	returns the formula	the formula	-
isotopeComposition	-	returns the isotope composition	the isotope composition	-
isotopeFormula	-	returns the isotope formula	the isotope formula	-
mass	-	calculates the molecule mass	the mass	-

massspectrum	-	calculates the mass spectrum, the m/z: relative abundance plot	the mass spectrum as a series of discrete values	-
sortableFormula	-	returns the fixed digit sortable formula	fixed digit sortable molecular formula	-

Back to Contents

General Functions

Name	License	Description	Return value	Parameters	Exam
abs	-	returns the absolute value of a number	the absolute value	integer or real number	Gene abs(7 (-4.9) Molec abs(7 (-4.9) React abs(7 (-4.9)

agentCount	-	counts the agents in the reaction	the number of agents in the reaction, or -1 if the the input (molecule) is not a reaction	-	Molec agent if the a reac contai agent -1 if th is not React agent if the a reac contai agent (0)) ar (produ not m will re
arom	-	returns if the atom has an aromatic bond	true if the atom has an aromatic bond, false otherwise	atom index or MolAtom object	Molec arom(atom aroma React arom(true if match reacti an aro
array	-	constructs an integer array from its arguments	the integer array	integers or MolAtom objects	Gener array(Molec array(React array((5), ra array((5), pa (8))
atno atomicNumber	-	returns the atomic number	the atomic number	atom index or MolAtom object	Molec atno(0 atomi 0 React atno(r the at the re match

atomProp	-				
atoms	-				
booleanToNumber	-	returns the number representation of a boolean value (true = 1, false = 0), or the number itself, if the input is a number	the number representation of a boolean value	boolean or number	Gener boolea (1<2)r boolea (2+2= boolea return Molec boolea (hasV return having error React boolea (ringB bond((2))))r reacta match 2 in th equati conne bond i corres reacta
connections	-	returns the bond plus implicit H count of an atom	the bond plus implicit H count	atom index or MolAtom object	Molec conne the nu conne React conne return conne reacta match reacti

count	-	determines the number of elements in an array	the number of array elements	integer array or real number array	Gener countt 1.2)) r Molec <ac:st ac:nar wiki-m schen macrc 8de7- 8748f plain-t [CDA ("char return atoms charg ("matc 1")) re of car]]></a body> macrc React <ac:st ac:nar wiki-m schen macrc 0816- 0979e plain-t [CDA (react > 0")) numb positiv secon (filter("matc 1")) re of car in the
elemanal	-				
eval	-				

field property	-	returns a molecule property (SDF field value)	the molecule property	the property key (SDF field name)	Molec field('A return ACTIV (SDF ('ACT return ACTIV bigge 0 othe React field(r 'ACTI' ACTIV value reacta (0), 'A (react 'ACTI' the AC the fir bigge first re otherv
fieldAsString propertyAsString	-	returns a molecule property (SDF field value) as string	the molecule property as string	the property key (SDF field name)	Molec fieldA return ID pro React field(r return ID pro reacta

filter	-	filters target atoms by filtering condition	target atom indices satisfying the filtering condition	target atom indices / objects or index / atom object array (optional, all atoms taken if omitted), filtering condition (boolean expression)	Molec <ac:st ac:nar wiki-m schen macrc 8f40-4 1aebc plain-t [CDA > 0") r indice positiv in the filter(6 ('[#8][(return carbo 6, 7, 8 molec]]></a body> macrc React <ac:st ac:nar wiki-m schen macrc 8d39- 6a8d3 plain-t [CDA (0), "c return atoms partial first re (pator "matc 1)") re carbo of pro match the re (note, are su the sa molec

formalCharge totalCharge	-	calculates formal charge of molecule	the formal charge value	atom index or MolAtom object (optional)	Molec forma the fo the in forma return charg React forma (0)) re charg reacta (raton forma reacta match
fragments	-	converts the molecule to its disconnected fragments	the disconnected fragments of the molecule	-	Molec fragm discor fragm molec React fragm return discor fragm secon
hasAromatizationError	-	determines if there is error in the aromatization of the molecule	true if there is an error in the aromatization of the molecule, false otherwise	-	Molec hasAr return an err aroma molec otherv React hasAr (react true if in the the fir otherv

hasIsotope	-	determines if any atom in the molecule is a specific isotope of the element	true if any atom in the molecule is a specific isotope of the element, false otherwise	-	Molec hasIs true if molec isotop false React hasIs (1)) re atom reacta isotop false
hasRadical	-	determines if any atom in the molecule has radical	true if any atom in the molecule has radical, false otherwise	-	Molec hasRa true if molec false React hasRa (1)) re atom reacta false
hasValenceError	-	determines if any atom in the molecule has valence error	true in case of valence error, false otherwise	-	Molec hasVa return in the valen otherv React hasVa (react true if secon valen otherv
HBDA	Structural Calculations License				

hCount	-	returns the hydrogen count of an atom	the hydrogen count	atom index or array of atom indeces (separated by , or 👑 or MolAtom object	Molec hCour the hy atom React hCour return count atom in the
importMol	-	imports and returns the molecule from its string representation	the string representation of the molecule	the molecule in string representation (e.g. "c1ccccc1")	Molec impor return molec React N / A are nu

in	-	determines whether an integer / atom index is contained in a given integer array	true if the array contains the specified integer, false otherwise	an integer or MolAtom object and an integer array	Gener in(5, a return (3, 5, Molec in(3, r ("char true if charg within partia input minat return polari is with smalle value molec React in(ratc (react 2)) ret partia reacta match reacti within partia first re (1), m (1), "p true if on pro match reacti within smalle value produ
ionCharge	Structural Calculations License				

isEmpty	-	decides whether the given molecule is empty (does not contain any atoms, bonds, or non-empty S-groups)	true if the molecule does not contain any atoms, bonds, or non-empty S-groups, false otherwise	-	Molec isQue the gi does atoms empty otherv React isQue return secon contai bonds group
isQuery	-	decides whether the given molecule contains any query features	true if the molecule contains any query features, false otherwise	-	Molec isQue the gi contai featur otherv React isQue return secon contai featur otherv
map	-	returns the atom map number	the atom map number	atom index or MolAtom object	Molec map(3 atom atom React map(p the at on the match reacti

max	-	takes maximum of its array and /or numerical parameters	the maximum value	integers, real numbers, integer arrays, real number arrays	Gene min(2 max(3 return Molec min(cl (2)) re the pa values 2 max return partia the inp React min(cl charg return partia on rea match 3 in th equat (produ the m charg first pr

maxAtom	-	evaluates objective function for each atom, finds largest value(s)	the atom index / indices corresponding to the largest evaluation result(s)	target atom indices / objects or index / atom object array (optional, all atoms taken if omitted), the objective function (as inner expression string), the number of largest values to be taken (optional, takes only one if omitted)	Molec maxA "charg select partia atoms major pH 7.4 molec the co indice React maxA patom "charg select partia produ match in the micros of the molec atoms corres (note, are su the sa molec

maxValue	-	evaluates objective function for each atom, finds largest value(s)	the largest evaluation result(s)	target atom indices / objects or index / atom object array (optional, all atoms taken if omitted), the objective function (as inner expression string), the number of largest values to be taken (optional, takes only one if omitted)	Molec maxV "charg return partia atoms major pH 7. molec React maxV patom "charg return partia produ match in the micros of the molec atoms atoms be in t molec

min	-	takes minimum of its array and /or numerical parameters	the minimum value	integers, real numbers, integer arrays, real number arrays	Gener min(2 max(3 return Molec min(cl (2)) re the pa values 2 max return partia the inp React min(cl charg return partia on rea match 3 in th equat (produ the m charg first pr
minAtom	-	evaluates objective function for each atom, finds smallest value(s)	the atom index / indices corresponding to the smallest evaluation result(s)	target atom indices / objects or index / atom object array (optional, all atoms taken if omitted), the objective function (as inner expression string), the number of smallest values to be taken (optional, takes only one if omitted)	Molec minAt ('7.4')' atom i corres minim charg micros of the React minAt "charg the at corres minim charg micros of the

minValue	-	evaluates objective function for each atom, finds smallest value(s)	the smallest evaluation result(s)	target atom indices / objects or index / atom object array (optional, all atoms taken if omitted), the objective function (as inner expression string), the number of smallest values to be taken (optional, takes only one if omitted)	Molec minVa ('7.4')' minim charg micros of the React minVa "charg the mi charg micros of the
molAtom	-	creates a MolAtom	the MolAtom object	atomic number	Molec molAt carbo React molAt carbo
molBinFormat molImage	-	returns the binary representation (image, pdf, GZIP compressed molecule file) of the molecule in specified format	the binary representation (image, pdf, GZIP compressed molecule file) of the molecule	the binary format with options (e.g. "jpeg", "png: w150,h150", "pdf", "gzip: sdf")	Molec molIm Q95,# the 10 image molec backg quality React N / A are nu

molString molConvert molFormat	-	returns the string representation of a molecule, or an array of molecules, in specified molecule format	the string representation of the molecule(s)	• the molecule format (e. g. "mol", "sdf", "mrv", "smiles") • the clean dimension	Molec molFc return Marvi forma of the molFc "sdf", SDF f repres tautor molec React N / A are nu
pair bond	-	converts two atoms or 0-based atom indexes into an "index1-index2" 1-based atom index setter string (used for pairing atoms in shortestPath)	the generated string	two atom indexes or two MolAtom objects	Gener pair(2 React pair(ra (2)) re index2 and "i based the re match in the equati (2), pa "index "index are th indexe atoms and 5 equati

productCount	-	counts the products in the reaction	the number of products in the reaction, or -1 if the the input (molecule) is not a reaction	-	Molec produ n if th is a re contai produ -1 if th is not React produ n if th is a re contai produ (0)) ar (produ not m will re
radicalCount	-	returns the radical count of an atom	the radical count	atom index or MolAtom object	Molec radica the ra atom React radica return count atom in the
reactantCount	-	counts the reactants in the reaction	the number of reactants in the reaction, or -1 if the the input (molecule) is not a reaction	-	Molec reacta n if th is a re contai reacta -1 if th is not React reacta n if th is a re contai reacta (0)) ar (produ not m will re

sortAsc	-	sorts an array in ascending order	the sorted array	integer array or real number array	Gener sortAs 1.2)) r 3.4, 5 Molec sortAs return charg ascen React sortAs (react the pa values reacta order
sortDesc	-	sorts an array in descending order	the sorted array	integer array or real number array	Gener sortDe 5.6, 1 (5.6, 3 Molec sortDe return values order React sortDe (0), "b the ba of the desce

sum	-	computes the sum of array elements	the sum	integer array or real number array	Gener sum(a 1.2)) r Molec sum(c the su values return atom values React sum(c (0))) r charg first re (produ the su polari the fir
topanal	Structural Calculations License				
valence	-	returns the sum of bond orders and query H atoms of an atom	the sum of bond orders and query H atoms	atom index or MolAtom object	Molec valenc valenc React valenc return the re match reacti
whereIsValenceError	-	returns the index of the first atom with valence error, or -1 if there is no valence error	the index of the first atom with valence error, or -1 if there is no valence error	-	Molec where return first at error, no val molec React where (react the in atom error i reacta is no v the se

Back to Contents

Geometry Functions

Name	License	Description	Return value	Parameters
aliphaticAtom	Structural Calculations License	checks if the specified atom is aliphatic	true for aliphatic atoms, false for non-aliphatic atoms	• the atom index / MolAtom object
aliphaticAtomCount	Structural Calculations License	calculates the aliphatic atom count	the aliphatic atom count	-
aliphaticBondCount	Structural Calculations License	calculates the aliphatic bond count	the aliphatic bond count	-

aliphaticRingCount	Structural Calculations License	calculates the aliphatic ring count	the aliphatic ring count	-
aliphaticRingCountOfSize	Structural Calculations License	calculates the number of aliphatic rings of given size	the number of aliphatic rings of given size	the ring size
aliphaticRings	Structural Calculations License	identifies the aliphatic rings in the molecule	atom indexes of the aliphatic rings in the molecule (null if the molecule does not contain aliphatic rings)	-
aliphaticRingsOfSize	Structural Calculations License	identifies the aliphatic rings in the molecule having a given size (number of atoms)	atom indexes of the aliphatic rings in the molecule having the given size (null if the molecule does not contain aliphatic rings)	the ring size

angle	Structural Calculations License	calculates the angle between three atoms	the angle between three atoms	• the (1-based) ato indexes of the three atoms in a string: "index1-index2-index3" (e '2-3-5')
aromaticAtom	Structural Calculations License	checks if the specified atom is aromatic	true for aromatic atoms, false for non-aromatic atoms	• the atom index / MolAtom object
aromaticAtomCount	Structural Calculations License	calculates the aromatic atom count	the aromatic atom count	-

aromaticBondCount	Structural Calculations License	calculates the aromatic bond count	the aromatic bond count	-
aromaticRingCount	Structural Calculations License	calculates the aromatic ring count	the aromatic ring count	-
aromaticRingCountOfSize	Structural Calculations License	calculates the number of aromatic rings of given size	the number of aromatic rings of given size	the ring size
aromaticRings	Structural Calculations License	identifies the aromatic rings in the molecule	atom indexes of the aromatic rings in the molecule (null if the molecule does not contain aromatic rings)	-

aromaticRingsOfSize	Structural Calculations License	identifies the aromatic rings in the molecule having a given size (number of atoms)	atom indexes of the aromatic rings in the molecule having the given size (null if the molecule does not contain aromatic rings)	the ring size
ASAHydrophobic	Structural Calculations License	calculates the water accessible molecular surface area of all hydrophobic atoms	the molecular surface area	• the major microspec pH (takes the input molecule itself if omitted)

ASANegative	Structural Calculations License	calculates the water accessible molecular surface area of all atoms with negative partial charge	the molecular surface area	• the major microspec pH (takes the input molecule itself if omitted)
ASAPlus	Structural Calculations License	calculates the water accessible molecular surface area of all atoms with positive partial charge	the molecular surface area	• the major microspec pH (takes the input molecule itself if omitted)

ASAPolar	Structural Calculations License	calculates the water accessible molecular surface area of all polar atoms	the molecular surface area	• the major microspec pH (takes the input molecule itself if omitted)
asymmetricAtom	Structural Calculations License	checks if the specified atom is an asymmetric atom	true for asymmetric atoms, false for symmetric atoms	• the atom index / MolAtom object
asymmetricAtomCount	Structural Calculations License	calculates the number of asymmetric atoms	the asymmetric atom count	-

asymmetricAtoms	Structural Calculations License	determines the asymmetric atoms	indexes of asymmetric atoms	-
balabanIndex	Structural Calculations License	calculates the Balaban index	the Balaban index	-
bondCount	-	calculates the bond count	the bond count	-
bondType	Structural Calculations License	returns the bond type between two atoms	the bond type between two atoms, -1 if there is no bond between the two atoms	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')

carboaliphaticRingCount	Structural Calculations License	calculates the number of carboaliphatic rings in the molecule (aliphatic rings containing carbon atoms only)	number of carboaliphatic rings	-
carboaromaticRingCount	Structural Calculations License	calculates the number of carboaromatic rings in the molecule (aromatic rings containing carbon atoms only)	number of carboaromatic rings	-
carboRingCount	Structural Calculations License	calculates the number of carbocyclic rings in the molecule (rings containing carbon atoms only)	number of carbocyclic rings	-
carboRingCountOfSize	Structural Calculations License	calculates the number of carbocyclic rings of given size (rings containing carbon atoms only)	the number of carbocyclic rings of given size	the ring size

carboRings	Structural Calculations License	identifies the carbocyclic rings in the molecule (rings containing carbon atoms only)	atom indexes of the carbocyclic rings in the molecule (null if the molecule does not contain carbocyclic rings)	-
carboRingsOfSize	Structural Calculations License	identifies the carbocyclic rings in the molecule having a given size (number of atoms)	atom indexes of the carbocyclic rings in the molecule having the given size (null if the molecule does not carbocyclic carbo rings)	the ring size
chainAtom	Structural Calculations License	checks if the specified atom is a chain atom	true for chain atoms, false for non-chain atoms	• the atom index / MolAtom object
chainAtomCount	Structural Calculations License	calculates the chain atom count	the chain atom count	-

chainBond	Structural Calculations License	checks if two atoms are connected by a chain bond	true if the two atoms are connected by a chain bond, false otherwise	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')
chainBondCount	Structural Calculations License	calculates the chain bond count	the chain bond count	-
chiralCenter	Structural Calculations License	checks if the specified atom is a tetrahedral stereogenic center	true for tetrahedral stereogenic center atoms	• the atom index / MolAtom object

chiralCenterCount	Structural Calculations License	calculates the number of tetrahedral stereogenic center atoms	the tetrahedral stereogenic center count	-
chiralCenters	Structural Calculations License	determines the chiral center atoms	indexes of chiral center atoms	-
connected	Structural Calculations License	checks if two atoms are connected	true if the two atoms belong to the same connected component, false otherwise	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')

connectedGraph	Structural Calculations License	checks whether the molecule graph is connected	true if the molecule graph is connected, false otherwise	-
cyclomaticNumber	Structural Calculations License	calculates the cyclomatic number	the cyclomatic number	-
dihedral	Structural Calculations License	calculates the dihedral of four atoms	the dihedral of four atoms	• the (1-based) ato indexes of the four atoms in a string: "index1-index2-index3-index4" (e '2-3-7-4')

distance	Structural Calculations License	calculates the distance between two atoms	the distance between two atoms	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')
distanceDegree	Structural Calculations License	calculates the distance degree of an atom	the distance degree	• the atom index / MolAtom object
dreidingEnergy	Structural Calculations License	returns the Dreiding energy of the input molecule (conformer)	the dreiding energy	-
eccentricity	Structural Calculations License	calculates the eccentricity of an atom	the eccentricity of an atom	• the atom index / MolAtom object

fragmentCount	Structural Calculations License	returns the number of fragments (disconnected parts)	the fragment count	-
fsp3	Structural Calculations License	returns the fsp3 of the molecule	the fsp3 value	-
fusedAliphaticRingCount	Structural Calculations License	calculates the number of fused aliphatic rings (SSSR smallest set of smallest aliphatic rings)	the fused aliphatic ring count	-
fusedAliphaticRingCountOfSize	Structural Calculations License	calculates the number of fused aliphatic rings of given size	the number of fused aliphatic rings of given size	the ring size

fusedAliphaticRings	Structural Calculations License	identifies the fused aliphatic rings in the molecule	atom indexes of the fused aliphatic rings in the molecule (null if the molecule does not contain fused aliphatic rings)	-
fusedAliphaticRingsOfSize	Structural Calculations License	identifies the fused aliphatic rings in the molecule having a given size (number of atoms)	atom indexes of the fused aliphatic rings in the molecule having the given size (null if the molecule does not contain fused aliphatic rings)	the ring size
fusedAromaticRingCount	Structural Calculations License	calculates the number of fused aromatic rings (SSSR smallest set of smallest aromatic rings)	the fused aromatic ring count	-

fusedAromaticRingCountOfSize	Structural Calculations License	calculates the number of fused aromatic rings of given size	the number of fused aromatic rings of given size	the ring size
fusedAromaticRings	Structural Calculations License	identifies the fused aromatic rings in the molecule	atom indexes of the fused aromatic rings in the molecule (null if the molecule does not contain fused aromatic rings)	-
fusedAromaticRingsOfSize	Structural Calculations License	identifies the fused aromatic rings in the molecule having a given size (number of atoms)	atom indexes of the fused aromatic rings in the molecule having the given size (null if the molecule does not contain fused aromatic rings)	the ring size
fusedRingCount	Structural Calculations License	calculates the number of fused rings (SSSR smallest set of smallest rings)	the fused ring count	-

hararyIndex	Structural Calculations License	calculates the Harary index	the Harary index	-
heteroaliphaticRingCount	Structural Calculations License	calculates the number of aliphatic heterocyclic rings (SSSR smallest set of smallest aliphatic rings)	the aliphatic heterocyclic ring count	-
heteroaliphaticRingCountOfSize	Structural Calculations License	calculates the number of aliphatic heterocyclic rings of given size	the number of aliphatic heterocyclic rings of given size	the ring size
heteroaliphaticRings	Structural Calculations License	identifies the aliphatic heterocyclic rings in the molecule	atom indexes of the aliphatic heterocyclic rings in the molecule (null if the molecule does not contain aliphatic heterocyclic rings)	-

heteroaliphaticRingsOfSize	Structural Calculations License	identifies the aliphatic heterocyclic rings in the molecule having a given size (number of atoms)	atom indexes of the aliphatic heterocyclic rings in the molecule having the given size (null if the molecule does not contain aliphatic heterocyclic rings)	the ring size
heteroaromaticRingCount	Structural Calculations License	calculates the number of aromatic heterocyclic rings (SSSR smallest set of smallest aromatic rings)	the aromatic heterocyclic ring count	-
heteroaromaticRingCountOfSize	Structural Calculations License	calculates the number of aromatic heterocyclic rings of given size	the number of aromatic heterocyclic rings of given size	the ring size

heteroaromaticRings	Structural Calculations License	identifies the aromatic heterocyclic rings in the molecule	atom indexes of the aromatic heterocyclic rings in the molecule (null if the molecule does not contain aromatic heterocyclic rings)	-
heteroaromaticRingsOfSize	Structural Calculations License	identifies the aromatic heterocyclic rings in the molecule having a given size (number of atoms)	atom indexes of the aromatic heterocyclic rings in the molecule having the given size (null if the molecule does not contain aromatic heterocyclic rings)	the ring size
heteroRingCount	Structural Calculations License	calculates the number of heterocyclic rings (SSSR smallest set of smallest rings)	the heterocyclic ring count	-

heteroRingCountOfSize	Structural Calculations License	calculates the number of heterocyclic rings of given size	the number of heterocyclic rings of given size	the ring size
heteroRings	Structural Calculations License	identifies the heterocyclic rings in the molecule	atom indexes of the heterocyclic rings in the molecule (null if the molecule does not contain heterocyclic rings)	-
heteroRingsOfSize	Structural Calculations License	identifies the heterocyclic rings in the molecule having a given size (number of atoms)	atom indexes of the heterocyclic rings in the molecule having the given size (null if the molecule does not contain heterocyclic rings)	the ring size
hyperWienerIndex	Structural Calculations License	calculates the Hyper Wiener index	the Hyper Wiener index	-

largestAtomRingSize	Structural Calculations License	calculates the size of the largest ring containing the specified atom	the size of the largest ring containing the specified atom	• the atom index / MolAtom object
largestConjugatedSystem	Structural Calculations License	returns the atom list of the largest conjugated system in the molecule	-	
largestConjugatedSystemSize	Structural Calculations License	returns the number of pi electron pairs in the largest conjugated system	-	
largestConjugatedSystemMol	Structural Calculations License	returns the input molecule with the largest conjugated system coloured in green	This function should be used with the output option *-f mrv* to export colour in the output	
largestRing	Structural Calculations License	identifies the atoms of the largest ring (number of atoms) in the molecule.	atom indexes of the largest ring in the molecule (null when acyclic)	-

largestRingSize	Structural Calculations License	calculates the largest ring size	the largest ring size	-
largestRingSystem	Structural Calculations License	identifies the atoms of the largest ring system (number of rings) in the molecule.	atom indexes of the largest ring system in the molecule (null when acyclic)	-
largestRingSystemSize	Structural Calculations License	calculates the size of the largest ring system (number of rings)	the size of the largest ring system	-
maximalProjectionArea	Structural Calculations License	returns the maximal projection area	the maximal projection area	-
maximalProjectionRadius	Structural Calculations License	returns the maximal projection radius	the maximal projection radius	-

maximalProjectionSize	Structural Calculations License	returns the maximal distance in the molecule perpendicular to the maximal projection area	the maximal distance in the molecule (the actual conformer) perpendicular to its maximal projection area	-
minimalProjectionArea	Structural Calculations License	returns the minimal projection area	the minimal projection area	-
minimalProjectionRadius	Structural Calculations License	returns the minimal projection radius	the minimal projection radius	-

minimalProjectionSize	Structural Calculations License	returns the maximal distance in the molecule perpendicular to the minimal projection area	the maximal distance of the molecule (the actual conformer) perpendicular to its minimal projection area	-
mmff94Energy	Structural Calculations License	returns the MMFF94 energy of the input molecule (conformer)	the MMFF94 energy	-
plattIndex	Structural Calculations License	calculates the Platt index	the Platt index	-
randicIndex	Structural Calculations License	calculates the Randic index	the Randic index	-
ringAtom	Structural Calculations License	checks if the specified atom is a ring atom	true for ring atoms, false for non-ring atoms	• the atom index / MolAtom object

ringAtomCount	Structural Calculations License	calculates the ring atom count	the ring atom count	-
ringBond	Structural Calculations License	checks if two atoms are connected by a ring bond	true if the two atoms are connected by a ring bond, false otherwise	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')
ringBondCount	Structural Calculations License	calculates the ring bond count	the ring bond count	-
ringCount	Structural Calculations License	calculates the ring count	the ring count	-

ringCountOfAtom	Structural Calculations License	calculates the number of rings passing through an atom	the number of rings passing through an atom	• the atom index / MolAtom object
ringCountOfSize	Structural Calculations License	calculates the number of rings of given size	the number of rings of given size	the ring size
rings	Structural Calculations License	identifies the rings in the molecule	atom indexes of the rings in the molecule (null if the molecule is acyclic)	-
ringsOfSize	Structural Calculations License	identifies the rings in the molecule having a given size (number of atoms)	atom indexes of the rings in the molecule having the given size (null if the molecule is acyclic or contains different rings only)	the ring size

ringSystemCount	Structural Calculations License	calculates the number of rings systems	the number of rings systems	-
ringSystemCountOfSize	Structural Calculations License	calculates the number of rings systems of given size	the number of rings systems of given size	the ring syste size
ringSystems	Structural Calculations License	identifies the ring systems in the molecule (fused and spiro rings belong to one ring system)	atom indexes of the ring systems in the molecule (null if the molecule is acyclic)	-
ringSystemsOfSize	Structural Calculations License	identifies the ring systems in the molecule having a given size (number of atoms, fused and spiro rings belong to one ring system)	atom indexes of the ring systems in the molecule having the given size (null if the molecule is acyclic or contains different ringSystems only)	the ring size

rotatableBond	Structural Calculations License	checks if two atoms are connected by a rotatable bond	true if the two atoms are connected by a rotatable bond, false otherwise	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')
rotatableBondCount	Structural Calculations License	calculates the rotatable bond count	the rotatable bond count	-
shortestPath	Structural Calculations License	calculates the length of the shortest path between two atoms	the length of the shortest path between two atoms, Integer. MAX_VALUE if disconnected	• the (1-based) ato indexes of the two atoms in a string: "index1-index2" (e '2-3')

smallestAtomRingSize	Structural Calculations License	calculates the size of the smallest ring containing the specified atom	the size of the smallest ring containing the specified atom	• the atom index / MolAtom object
smallestRing	Structural Calculations License	identifies the atoms of the smallest ring (number of atoms) in the molecule.	atom indexes of the smallest ring in the molecule (null when acyclic)	-
smallestRingSize	Structural Calculations License	calculates the smallest ring size	the smallest ring size	-
smallestRingSystem	Structural Calculations License	identifies the atoms of the smallest ring system (number of rings) in the molecule.	atom indexes of the smallest ring system in the molecule (null when acyclic)	-
smallestRingSystemSize	Structural Calculations License	calculates the size of the smallest ring system (number of rings)	the size of the smallest ring system	-

stereoDoubleBondCount	Structural Calculations License	calculates the number of stereo double bonds	the stereo double bond count	-
stericEffectIndex	Structural Calculations License	calculates the steric effect index of an atom	the steric effect index of an atom	• the atom index / MolAtom object
stericHindrance	Structural Calculations License	calculates the steric hindrance of an atom	the steric hindrance of an atom	• the atom index / MolAtom object
szegedIndex	Structural Calculations License	calculates the Szeged index	the Szeged index	-

topologicalPolarSurfaceArea PSA	-	calculates the topological polar surface area (2D)	the polar surface area (2D)	• the major microspec pH (takes the input molecule itself if omitted)
vanDerWaalsSurfaceArea	Structural Calculations License	calculates the van der Waals surface area	the molecular surface area	• the major microspec pH (takes the input molecule itself if omitted)

waterAccessibleSurfaceArea ASA solventAccessibleSurfaceArea	Structural Calculations License	calculates the solvent accessible / water accessible molecular surface area	the molecular surface area	• the major microspec pH (takes the input molecule itself if omitted)
wienerIndex	Structural Calculations License	calculates the Wiener index	the Wiener index	-
wienerPolarity	Structural Calculations License	calculates the Wiener polarity	the Wiener polarity	-

Back to Contents

HBDA Functions

Name	License	Description	Return value	Parameters	Examples

acceptor acc	Structural Calculations License	calculates atomic hydrogen bond acceptor multiplicity	the atomic hydrogen bond acceptor multiplicity	• the atom index / MolAtom object • the major microspecies pH (takes the input molecule itself if omitted)	Molecule Co acceptor(2) r the hydroger acceptor multiplicity or 2 of the input molecule Reaction Co acceptor(rat returns the hydrogen bo acceptor multiplicity or reactant ato matching ma the reaction equation
acceptorCount	Structural Calculations License	calculates molecular hydrogen bond acceptor count (the number of acceptor atoms)	the molecular hydrogen bond acceptor count	• the major microspecies pH (takes the input molecule itself if omitted)	Molecule Co acceptorCou returns the n of hydrogen acceptor ato the input mol Reaction Co acceptorCou (reactant(0)) returns the n of hydrogen acceptor ato the first react
acceptorSiteCount accSiteCount	Structural Calculations License	calculates molecular hydrogen bond acceptor multiplicity (the sum of atomic multiplicities)	the molecular hydrogen bond acceptor multiplicity	• the major microspecies pH (takes the input molecule itself if omitted)	Molecule Co acceptorSite returns the hydrogen bo acceptor multiplicity of input molecu Reaction Co acceptorSite (reactant(0)) returns the hydrogen bo acceptor multiplicity of first reactant

donor don	Structural Calculations License	calculates atomic hydrogen bond donor multiplicity	the atomic hydrogen bond donor multiplicity	• the atom index / MolAtom object • the major microspecies pH (takes the input molecule itself if omitted)	Molecule Co donor(1, "7.4 returns the hydrogen bo donor multipl on atom 1 of major micros at pH 7.4 Reaction Co returns the hydrogen bo donor multipl on the produ atom matchi map 3 in the reaction equ taking the ma microspecies correspondi product at pH
donorCount	Structural Calculations License	calculates molecular hydrogen bond donor count (the number of donor atoms)	the molecular hydrogen bond donor count	• the major microspecies pH (takes the input molecule itself if omitted)	Molecule Co donorCount(" returns the n of hydrogen b donor atoms major micros at pH 7.4 Reaction Co donorCount((1), "7.4") ret the number o hydrogen bo donor atoms major micros of the first pr taken at pH 7

donorSiteCount donSiteCount	Structural Calculations License	calculates molecular hydrogen bond acceptor / donor multiplicity (the sum of atomic multiplicities)	the molecular hydrogen bond acceptor / donor multiplicity	• the major microspecies pH (takes the input molecule itself if omitted)	Molecule Co donorSiteCo ("7.4") return hydrogen bo donor multipl the major microspecies 7.4 Reaction Co donorSiteCo (product(1), ' returns the hydrogen bo donor multipl the major microspecies first product t at pH 7.4

Back to Contents

Huckel Functions

Name	License	Description	Return value	Paramet
chargeDensity totalChargeDensity	Structural Calculations License	calculates the charge density of atoms	the charge density of the atom, NaN for non-existing values	• he ato index MolAt object • the ma micros pH (ta the inp molec itself i omitte

electronDensity piChargeDensity	Structural Calculations License	calculates the electron density of atoms	the electron density of the atom, NaN for non-existing values	• he ato index MolAt object • the ma micros pH (ta the inp molec itself it omitte
electrophilicity energyNu nucleophilicLocalizationEnergy	Structural Calculations License	calculates electrophilicity of atoms	the electrophilicity of the atom, NaN for non-aromatic atoms	• the ato index MolAt object • the ma micros pH (ta the inp molec itself it omitte
electrophilicityOrder aromaticElectrophilicityOrder orderE	Structural Calculations License	calculates E_{+} order of atoms	the E_{+} order index of the atom (0, 1, 2, ...),	• the ato index MolAt object • the ma micros pH (ta the inp molec itself it omitte

hmoChargeDensity	Structural Calculations License	calculates the HMO charge density of atoms	the charge density of the atom, NaN for non-existing values	• he ato index MolAt object • the ma micros pH (ta the inp molec itself i omitte
hmoElectronDensity	Structural Calculations License	calculates the HMO electron density of atoms	the electron density of the atom, NaN for non-existing values	• he ato index MolAt object • the ma micros pH (ta the inp molec itself i omitte
hmoElectrophilicityOrder hmoOrderE	Structural Calculations License	calculates HMO $E_{\oplus}$ order of atoms	the $E_{\oplus}$ order index of the atom (0, 1, 2, ...),	• the ato index MolAt object • the ma micros pH (ta the inp molec itself i omitte

hmoElectrophilicLocalizationEnergy	Structural Calculations License	calculates HMO localization energy L^+ of atoms	the localization energy L^+ of the atom, NaN for non-aromatic atoms	• the at index MolAt object • the ma micros pH (ta the inp molec itself i omitte
hmoNucleophilicityOrder hmoOrderNu	Structural Calculations License	calculates HMO Nu^- order of atoms	the Nu^- order index of the atom (0, 1, 2, ...),	• the at index MolAt object • the ma micros pH (ta the inp molec itself i omitte
hmoNucleophilicLocalizationEnergy	Structural Calculations License	calculates HMO localization energy L^- of atoms	the localization energy L^- of the atom, NaN for non-aromatic atoms	• the at index MolAt object • the ma micros pH (ta the inp molec itself i omitte

hmoPiEnergy	Structural Calculations License	calculates the HMO pi energy of the molecule	the pi energy of the molecule	• the ma micros pH (ta the inp molec itself if omitte
nucleophilicity electrophilicLocalizationEnergy energyE	Structural Calculations License	calculates nucleophilicity of atoms	the nucleophilicity of the atom, NaN for non-aromatic atoms	• the ato index MolAto object • the ma micros pH (ta the inp molec itself if omitte
nucleophilicityOrder aromaticNucleophilicityOrder orderNu	Structural Calculations License	calculates Nu⊖ order of atoms	the Nu⊖ order index of the atom (0, 1, 2, ...),	• the ato index MolAto object • the ma micros pH (ta the inp molec itself if omitte
piEnergy	Structural Calculations License	calculates the pi energy of the molecule. Deprecated.	the pi energy of the molecule	• the ma micros pH (ta the inp molec itself if omitte

Back to Contents

Isomers Functions

Name	License	Description	Return value	Parameters
allTautomer	Isomers License	returns a tautomeric form from all tautomers	the tautomer	• the tautomer index (0-based)
allTautomers	Isomers License	returns all tautomer forms in an array	the tautomer array	the normalization option (default is false)
canonicalResonant	-	constructs the canonical resonant structure	the canonical resonant structure	-
canonicalTautomer	Isomers License	generates the canonical tautomer structure	the canonical tautomer	option that switches on the normalization (defaults is false)

dominantTautomer tautomer	Isomers License	returns the i-th dominant tautomeric form	the i-th dominant tautomer	• the dominant tautomer index (0-based) • the pH value as string (set if pH effect should be considered)
dominantTautomerCount	Isomers License	calculates the number of dominant tautomers	the number of dominant tautomers	• the pH value as string (set if pH effect should be considered)

dominantTautomers tautomers	Isomers License	constructs all dominant tautomeric forms (ordered by distribution)	the dominant tautomer array	• the pH value as string (set if pH effect should be considered)
doubleBondStereoisomer	Isomers License	generates a double bond stereoisomer of the molecule	the double bond stereoisomer	• the double bond stereoisomer index (0-based)
doubleBondStereoisomerCount	Isomers License	returns the number of generated double bond stereoisomers	the number of generated double bond stereoisomers	-

doubleBondStereoisomers	Isomers License	generates double bond stereoisomers of the molecule (maximum number of double bond stereoisomers to be generated can be set, default: all)	the double bond stereoisomer array	-
genericTautomer	Isomers License	constructs the generic tautomer structure	the generic tautomer structure	-
majorTautomer	Isomers License	constructs the major tautomer structure	the major tautomer structure	• the pH value as string (set if pH effect should be considered)
mostStableTautomer	Isomers License	deprecated, use majorTautomer instead. constructs the most stable tautomer structure	the most stable tautomer structure	-
resonant	-	constructs a resonant structure	the resonant structure	-

resonantCount	-	calculates the number of resonant structures	the number of resonant structures	-
resonants	-	constructs all resonant structures	the resonant structure array	-
stereoAnalysis	-	calculates stereo descriptors of molecule	list of stereo descriptors	• - • the type of stereo descriptor: see the API function names
stereoisomer	Isomers License	generates a stereoisomer of the molecule	the stereoisomer	• the stereoisomer index (0-based)

stereoisomerCount	Isomers License	returns the number of generated stereoisomers	the number of generated stereoisomers	-
stereoisomers	Isomers License	generates stereoisomers of the molecule (maximum number of stereoisomers to be generated can be set, default: all)	the stereoisomer array	-
tautomerCount	Isomers License	calculates the number of tautomers	the number of tautomers	-
tetrahedralStereoisomer	Isomers License	generates a tetrahedral stereoisomer of the molecule	the tetrahedral stereoisomer	-

tetrahedralStereoisomerCount	Isomers License	returns the number of generated tetrahedral stereoisomers	the number of generated tetrahedral stereoisomers	-
tetrahedralStereoisomers	Isomers License	generates tetrahedral stereoisomers of the molecule (maximum number of tetrahedral stereoisomers to be generated can be set, default: all)	the tetrahedral stereoisomer array	-

Back to Contents

Markush Functions

Name	License	Description	Return value	Paramete
isMarkush	-	decides whether the given molecule contains any Markush features	true if the molecule contains any Markush features, false otherwise	-

markushEnumerationCount enumerationCount	Markush Enumeration Plugin	calculates the number of Markush enumerations	the number of Markush enumerations	• the (1-based) atom indexe the que atoms enume (defaul
markushEnumerations enumeration enumerations markushEnumeration	Markush Enumeration Plugin	constructs Markush enumerated structures sequentially	the enumerated structures	• the nur of structu to be returne (defaul • the (1-based) atom indexe the que atoms enume (defaul

markushEnumerationsDisplay	Markush Enumeration Plugin	constructs Markush enumerated structures sequentially with scaffold alignment and scaffold /R-group coloring and enumeration ID	the enumerated structures with alignment and coloring data and enumeration ID	• the nur of structu to be returne (defaul • the (1-based) atom indexe the que atoms enume (defaul
markushLibraryMagnitude	Markush Enumeration Plugin	calculates the Markush library magnitude, no enumeration is done	the Markush library magnitude	• the (1-based) atom indexe the que atoms enume (defaul

markushLibrarySize	Markush Enumeration Plugin	calculates the Markush library size, no enumeration is done	the Markush library size	• the (1-based) atom indexe the que atoms enume (defaul
markushLibrarySizeAsString	Markush Enumeration Plugin	calculates the Markush library size and returns it as string, no enumeration is done	the Markush library size	• the (1-based) atom indexe the que atoms enume (defaul

randomMarkushEnumerations randomEnumeration randomEnumerations randomMarkushEnumeration	Markush Enumeration Plugin	constructs Markush enumerated structures randomly	the enumerated structures	• the nur of structu to be returne (defaul • the (1-based) atom indexe the que atoms enume (defaul
randomMarkushEnumerationsDisplay	Markush Enumeration Plugin	constructs Markush enumerated structures randomly with scaffold alignment and scaffold /R-group coloring and enumeration ID	the enumerated structures with alignment and coloring data and enumeration ID	• the nur of structu to be returne (defaul • the (1-based) atom indexe the que atoms enume (defaul

Back to Contents

Match Functions

Name	License	Description	Return value	Parameters	Examples

disjointMatchCount	-	performs substructure search, returns the maximal number of pairwise disjoint search hits	the maximal number of pairwise disjoint search hits	• target atom index / MolAtom object (optional) • query Molecule object / SMARTS string • query atom map(s) (optional) The function returns the maximal number of pairwise disjoint query structures found in the target molecule. Warning: if the target atom index and optionally query atom maps are specified then the return value can only be 0 or 1, therefore the result is similar to the result of the match function.	Molecule Context <ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="32063639-03ca-4f1c-9f87-82aa66315fac"><ac:plain-text-body><![CDATA[disjointMatchCour("[#8]C=O") counts the maximal number of pairwise disjoint carboxylic groups in the input molecule]]></ac:plain-text-body></ac:structured-macro> Reaction Context <ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="c7696101-9696-47e1-89e4-bd6f6914fb9b"><ac:plain-text-body><![CDATA[disjointMatchCour(reactant(0), "[#8]C=O") counts the maximal number o pairwise disjoint carboxylic groups in the first reactant
match	-				Molecule Context <ac:structured-

		performs substructure search and optionally checks for atom matching	true if matching substructure found, false otherwise	• target atom index / MolAtom object (optional) • query Molecule object / SMARTS string • query atom map(s) (optional) The function returns true if the query structure is found in the target molecule, the hit is required to include the target atom if specified, furthermore if query atom map (s) are specified then these mapped atoms should match the target atom.	macro ac:name=" unmigrated-wiki-markup" ac: schema-version=" 1" ac:macro-id=" e12cb441-0b95-4bd7-b3b7-825793aa9439" ><ac:plain-text-body><![CDATA [match("[#8]C=O") performs substructure search without atom matching requirement, the target is the input molecule, the query is the carboxylic group given in the string parameter match (6, "[#8][C:1]=O", 1) performs substructure search, checks if target atom 6 matches the carbon (atom with map 1) of the carboxylic group query match(6, "[#8:1]C=[O:2]", 1, 2) performs substructure search, checks if target atom 6 of the input molecule is a carboxylic oxygen]]></ac:plain-text-body></ac: structured-macro> Reaction Context <ac:structured-macro ac:name=" unmigrated-wiki-markup" ac: schema-version=" 1" ac:macro-id="

					f39f6396-975c-4b7b-ae76-24f463898f0f"><ac plain-text-body><![CDATA[match (reactant(0), "[#8]C=O") performs substructure search without atom matching requirement, the target is the first reactant, the query is the carboxylic group given in the string parameter match(patom(2), "[#8]C=O") performs substructure search, checks if product atom matching map 2 in the reaction equation matches any atom of the carboxylic group query match(ratom (1), "[#8:1]C=[O:2]", 1, 2) performs substructure search, checks if reactant atom matching map 1 in the reaction equation is a carboxylic oxygen
matchCount	-	performs substructure search and optionally checks for atom matching, counts search hits	the number of search hits		Molecule Context <ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="8a943b00-ca03-44ba-8942-d5b1a2c43197"><ac:plain-text-body><![CDATA

				• target atom index / MolAtom object (optional) • query Molecule object / SMARTS string • query atom map(s) (optional) The function returns the number of query structures found in the target molecule, the hit is required to include the target atom if specified, furthermore if query atom map(s) are specified then these mapped atoms should match the target atom.	[matchCount("[#8]C=O") counts search hits without atom matching requirement, the target is the input molecule, the query is the carboxylic group given in the string parameter matchCount(6, "[#8]C=O") counts search hits with target atom 6 matching any atom in a carboxylic group matchCount(6, "[#8:1]C=[O:2]" 1, 2) counts search hits with target atom 6 of the input molecule being a carboxylic oxygen]]></ac:plain-text-body></ac:structured-macro> Reaction Context <ac:structured-macro ac:name="unmigrated-wiki-markup" ac:schema-version="1" ac:macro-id="c77d67a8-f859-48a3-a482-c99ca63b1d49" ><ac:plain-text-body><![CDATA [matchCount(reactant(0), "[#8]C=O") counts search hits without atom matching requirement, the target is the first reactant, the query is the carboxylic group given in the

					string parameter matchCount(patom (2), "[#8]C=O") counts search hits, checks if product atom matching map 2 in the reaction equation matches any atom of the carboxylic group query matchCount(ratom (1), "[#8:1]C=[O: 2]", 1, 2) counts search hits with reactant atom matching map 1 in the reaction equation being a carboxylic oxygen
matchFirst	-	performs substructure search and optionally checks for atom matching	index of the first matching substructure (1-based indexing)	• target atom index / MolAtom object (optional) • query Molecule objects / SMARTS strings in collection (e.g. {amine, amide,"[#8][C:1]=O"}) • query atom map(s) (optional)	Molecule Context `matchFirst ({amine,amide, alcohol})` performs substructure search without atom matching requirement, the target is the input molecule, the queries are the amine, amide, and alcohol groups given in the string parameter. Function returns 1 if there is an amine group in the input molecule, 2 if there is no amine, but there is an amide group in the input molecule, and returns 3 if there is no amine or amide group, but there is an alcohol group in the input molecule

					The function returns the index of the first matching query structure found in the target molecule, the hit is required to include the target atom if specified, furthermore if query atom map(s) are specified then these mapped atoms should match the target atom.	Returns `0` if none of the listed groups is found in input molecule. `matchFirst(6, {"[#8][C:1]=O","[NX3:2][CX3:1]=[OX1:3]"}, 1)` performs substructure search, checks if target atom 6 matches the carbon (atom with map `1`) of the carboxylic or amide group in query. Returns the index of the query if match found, `0` otherwise. `matchFirst(6, {"[#8:1]C=[O:2]","[#6][OX2:1][CX3](=[O:2])[#6]"}, 1, 2)` performs substructure search, checks if target atom 6 of the input molecule is a carboxylic oxygen or an oxigen in an ester group. Returns the index of the query if match found, `0` otherwise. Reaction Context `matchFirst(reactant(0), {amine,amide,alcohol})` performs substructure search without atom matching requirement, the

					target is the first reactant, the queries are the amine, amide, and alcohol groups given in the string parameter. Function returns `1` if there is an amine group in the first reactant, `2` if there is no amine group, but there is an amide group in the first reactant, and returns `3` if there is no amine or amide group, but there is an alcohol group in the first reactant. Returns `0` if none of the listed groups is found in first reactant. `matchFirst (patom(2), {"[#8][C]=O","[NX3][CX3]=[OX1]"})` performs substructure search, checks if product atom matching map `2` in the reaction equation matches any atom of the carboxylic or amide group query. Returns the index of mathing query, or `0` if no match found. `matchFirst (ratom(1), {"[#8:1]C=[O:2]","[#6][OX2:1][CX3](=[O:2])[#6]"}, 1,`

					2) performs substructure search, checks if reactant atom matching map 1 in the reaction equation is a carboxylic oxygen or an oxigen in an ester group. Returns the index of the query if match found, 0 otherwise.

Predefined Molecules and Molecule Sets

It is sometimes easier to refer molecules by names rather than explicit SMARTS strings or molecule file paths. For example, you may want to write nitro or carboxyl as query in a match function. Frequently used queries are pre-defined in the built-in functional groups file (chemaxon/marvin/templates/functionalgroups.cxsmi within MarvinBeans-templates.jar).
You can also define your favourite query SMARTS in marvin/config/marvin/templates/functionalgroups.cxsmi file and in $HOME\chemaxon\marvin\templates\functionalgroups.cxsmi (Windows) or $HOME/.chemaxon/marvin/templates/functionalgroups.cxsmi (UNIX / Linux) file where marvin is the Marvin istallation directory, $HOME is your user home directory.
Back to Contents

Name Functions

Name	License	Description	Return value	Parameters	Examples

name	Structure To Name License	returns the name(s) of the molecule specified by the export option	the name (s) of the molecule	either empty (IUPAC export option) or the specified name export option	Molecule Context name() returns the preferred IUPAC name of the input molecule name('CAS#') returns CAS Registry Number (s) of the input molecule Reaction Context N / A (reaction rules are numerical)
traditionalName	Structure To Name License	returns the traditional name of the molecule	the traditional name of the molecule	-	Molecule Context traditionalName() returns the traditional name of the input molecule Reaction Context N / A (reaction rules are numerical)

Back to Contents

Partitioning Functions

Name	License	Description	Return value	Parameters	Examples

HLB number	Partitioning License	calculates the HLB number	the HLB number	• the calculation method or empty	Molecule Context hlb() returns the HLB number of the input molecul using the default calculation method hlb ('davies') returns the HLB number of the input molecule using the Davies calculation method Reaction Context hlb(reactant (1), 'griffin') returns the HLB number of the second reactant using the Griffin method
logD	Partitioning License	calculates log*D* at specified pH	the log*D* value	• the pH value	Molecule Context logD('7.4') returns the lo *D* at pH 7.4 o the input molecule Reaction Context logD(reactant (1), '7.4') returns the lo *D* at pH 7.4 o the second reactant

logDKLOP	Partitioning License	calculates log*D* at specified pH using method "KLOP"	the log*D* value	• the pH value	Molecule Context logDKLOP ('7.4') returns the log*D* at pH 7.4 of the input molecul Reaction Context logDKLOP (reactant(1), '7.4') returns the log*D* at pH 7.4 of the second reactant
logDPHYS	Partitioning License	calculates log*D* at specified pH using method "PHYS"	the log*D* value	• the pH value	Molecule Context logDPHYS ('7.4') returns the log*D* at pH 7.4 of the input molecul Reaction Context logDPHYS (reactant(1), '7.4') returns the log*D* at pH 7.4 of the second reactant

logDUser	Partitioning License	calculates log*D* at specified pH using the user defined method	the log*D* value	• the pH value	Molecule Context logDUser('7.4 returns the lo *D* at pH 7.4 o the input molecule Reaction Context logDUser (reactant(1), '7.4') returns the log*D* at pH 7.4 of the second reactant
logDVG	Partitioning License	calculates log*D* at specified pH using method "VG"	the log*D* value	• the pH value	Molecule Context logDVG('7.4') returns the lo *D* at pH 7.4 o the input molecule Reaction Context logDVG (reactant(1), '7.4') returns the log*D* at pH 7.4 of the second reactant

logDWeighted	Partitioning License	calculates log*D* at specified pH using weighted method	the log*D* value	• the pH value	Molecule Context logDWeighted ('7.4') returns the log*D* at pH 7.4 of the input molecul Reaction Context logDWeighted (reactant(1), '7.4') returns the log*D* at pH 7.4 of the second reactant

logP	Partitioning License	calculates log*P*	the log*P* value	• the result type:
				  • "logPMicro": the log*P* of the input molecule itself
				  • "logPNonionic": the log*P* of the nonionic species
				  • "logDpI": log*D* at p*I*
				  • "logPTrue": the most typical from the above (default)
					Molecule Context logP() returns the most typical log*P* o the input molecule log*P* the log*P* of th nonionic species and log*D* at pI log ('logPMicro') returns the lo *P* of the input molecule itsel Reaction Context logP(reactant (1), 'logDpI') returns the lo *D* at p*I* of the second reactant logP (product(1), 'logPNonionic returns log*P* o the nonionic species of the second produ

logPincrement logPi	Partitioning License	calculates the atomic log*P* increment	the atomic log*P* increment	• the atom index / MolAtom object	Molecule Context logPincremer (2) returns the log*P* increment on atom 2 of the input molecul Reaction Context logPincremer (ratom(1)) returns the lo *P* increment on the reactant atom matching map 1 in the reaction equation

logPKLOP	Partitioning License	calculates log*P* using method "KLOP"	the log*P* value	• the result type:
				  • "logPMicro": the log*P* of the input molecule itself
				  • "logPNonionic": the log*P* of the nonionic species
				  • "logDpI": log*D* at p*I*
				  • "logPTrue": the most typical from the above (default)
					Molecule Context logPKLOP() returns the most typical lo *P* out of the input molecul log*P*, the log*P* of the nonioni species and log*D* at p*I* logPKLOP ('logPMicro') returns the log *P* of the input molecule itsel Reaction Context logPKLOP (reactant(1), 'logDpI') returns the log *D* at p*I* of the second reactant logPKLOP (product(1), 'logPNonionic' returns log*P* o the nonionic species of the second produ

logPPHYS	Partitioning License	calculates log*P* using method "PHYS"	the log*P* value	• the result type:
				  • "logPMicro": the log*P* of the input molecule itself
				  • "logPNonionic": the log*P* of the nonionic species
				  • "logDpI": log*D* at p*I*
				  • "logPTrue": the most typical from the above (default)
					Molecule Context logPPHYS() returns the most typical lo*P* out of the input molecul log*P*, the log*P* of the nonioni species and log*D* at p*I* logPPHYS ('logPMicro') returns the log*P* of the input molecule itsel Reaction Context logPPHYS (reactant(1), 'logDpI') returns the log*D* at p*I* of the second reactant logPPHYS (product(1), 'logPNonionic returns log*P* o the nonionic species of the second produ

logPUser	Partitioning License	calculates log*P* using the user defined method	the log*P* value	• the result type:
				  • "logPMicro": the log*P* of the input molecule itself
				  • "logPNonionic": the log*P* of the nonionic species
				  • "logDpI": log*D* at p*I*
				  • "logPTrue": the most typical from the above (default)
					Molecule Context logPUser() returns the most typical lo*P* out of the input molecul log*P*, the log*P* of the nonioni species and log*D* at p*I* logPUser ('logPMicro') returns the log*P* of the input molecule itsel Reaction Context logPUser (reactant(1), 'logDpI') returns the log*D* at p*I* of the second reactant logPUser (product(1), 'logPNonionic returns log*P* o the nonionic species of the second produ

logPVG	Partitioning License	calculates log*P* using method "VG"	the log*P* value	• the result type:
				  • "logPMicro": the log*P* of the input molecule itself
				  • "logPNonionic": the log*P* of the nonionic species
				  • "logDpI": log*D* at p*I*
				  • "logPTrue": the most typical from the above (default)
					Molecule Context logPVG() returns the most typical lo*P* out of the input molecul log*P*, the log*P* of the nonioni species and log*D* at p*I* logPVG ('logPMicro') returns the log*P* of the input molecule itsel Reaction Context logPVG (reactant(1), 'logDpI') returns the log*D* at p*I* of the second reactant logPVG (product(1), 'logPNonionic returns log*P* o the nonionic species of the second produ

logPWeighted	Partitioning License	calculates log*P* using weighted method	the log*P* value	• the result type:
				  • "logPMicro": the log*P* of the input molecule itself
				  • "logPNonionic": the log*P* of the nonionic species
				  • "logDpI": log*D* at p*I*
				  • "logPTrue": the most typical from the above (default)
					Molecule Context logPWeighted returns the most typical l*P* out of the input molecul log*P*, the log*P* of the nonioni species and log*D* at p*I* logPWeighted ('logPMicro') returns the lo*P* of the input molecule itsel Reaction Context logPWeighted (reactant(1), 'logDpI') returns the lo*D* at p*I* of the second reactant logPWeighted (product(1), 'logPNonionic returns log*P* o the nonionic species of the second produ

Back to Contents

Protonation Functions

Name	License	Description	Return value	Parameters

acidicpKa apKa	Protonation License	calculates acidic pK_a values	the acidic pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges Note, that the str index is specifiec between quotatic marks.
acidicpKaLargeModel	Protonation License	calculates acidic pK_a values using large model (this model is optimized for a large number of ionizable atoms)	the acidic pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges Note, that the str index is specifiec between quotatic marks.

acidicpKaUseCorrection	Protonation License	calculates acidic pK_a values using the correction library	the acidic pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges Note, that the str index is specifiec between quotatic marks.
averageMicrospeciesCharge	Protonation License	calculates the average microspecies charge (weighted sum of charges of all the microspecies of the molecule) at the given pH	the average charge	• the major micros pH
basicpKa bpKa	Protonation License	calculates basic pK_a values	the basic pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges Note, that the str index is specifiec between quotatic marks.

basicpKaLargeModel	Protonation License	calculates basic pK_a values using large model (this model is optimized for a large number of ionizable atoms)	the basic pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges Note, that the str index is specifiec between quotatic marks.
basicpKaUseCorrection	Protonation License	calculates basic pK_a values using the correction library	the basic pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges Note, that the str index is specifiec between quotatic marks.
isoelectricPoint pI	Protonation License	calculates isoelectric point	the isoelectric point	-

majorMicrospecies majorMs	Protonation License	calculates major microspecies at specified pH	the major microspecies	• the pH value as s
microspecies ms	Protonation License	calculates microspecies at specified pH	the microspecies	• the pH value as s • the microspecies by descending or microspecies distributions
microspeciesCount msCount	Protonation License	calculates the number of microspecies at specified pH	the number of microspecies	-

microspeciesDistribution msDistr	Protonation License	calculates microspecies distribution at specified pH	the microspecies distribution	• the pH value as s • the microspecies by descending or microspecies distributions

pKa	Protonation License	calculates pK_a values	the pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g., '1' fo strongest, '2' for second stronges • the result type ("a "basic", or "pka" (default)) Note, that the str index is specifiec between quotatic marks. In case o strength index th type can be spec "acidic" or "basic case of "pka"resu the returned pK_a are acidic or bas (mixed!), depend the acidic or basi character of the atom. Acidic pK_a returned for an a *acidicpKa() 14.8* *basicpKa()*, othe basic pK_a is retu Specifying "acidi "basic" result typ required to get th or basic pK_a valu (also acidicpKa c basicpKa functio be used alternati

pKaUseCorrection	Protonation License	calculates pK_a values using the correction library	the pK_a values	• the atom index / MolAtom object, • the strength inde string (e.g. '1' for strongest, '2' for second stronges • the result type ("a "basic", or "pka" (default)) Note, that the str index is specifiec between quotatic marks. In case o strength index th type can be spec "acidic" or "basic case of "pka"resu the returned pK_a are acidic or bas (mixed!), depend the acidic or basi character of the atom. Acidic pK_a returned for an a *acidicpKa() 14.8* *basicpKa()*, othe basic pK_a is retu Specifying "acidi "basic" result typ required to get th or basic pK_a val (also acidicpKaUseCo or basicpKaUseCor functions can be alternatively).

Back to Contents

Refractivity Functions

Name	License	Description	Return value	Parameters	Examples
refractivity refrac	Structural Calculations License	calculates molar refractivity	the refractivity value	-	Molecule Cont refractivity() re the molar refra of the input molecule Reaction Cont refractivity(rea (1)) returns the molar refractiv the second rea
refractivityIncrements refraci	Structural Calculations License	calculates atomic refractivity increments	the atomic refractivity increment	• the atom index / MolAtom object	Molecule Cont refractivityIncr (2) returns the refractivity incr on atom 2 of th input molecule Reaction Cont refractivityIncr (ratom(1)) retu the refractivity increment on t reactant atom matching map the reaction ec

Back to Contents

Solubility Functions

Name	License	Description	Return value	Parameters	Examples

logS	Solubility License	Calculates intrinsic solubility or solubility at a specified pH	the log *S* value	• the pH value • the solubility unit: logS, mol/l, mg /ml, category	Molecule Context logS() returns the intrinsic solubility of the input molecule in logS unit (default) logS('7.4', 'category') returns the qualitative log *S* at pH 7.4 of the input molecule logS('7.4', 'mol/l') returns the log*S* at pH 7.4 of the input molecule in mol/l unit

Back to Contents

StructuralFrameworks Functions

Name	License	Description	Return value	Parameter
bmf	Structural Calculations License	Returns the Bemis-Murcko framework of the input structure	BMF of the input structure	-

bmfl bemisMurckoFrameworkLoose	Structural Calculations License	Returns the Bemis-Murcko loose framework of the input structure. Calculated by removing side chains. Exocyclic non single bonded atoms are kept. Remaining atom and bond types are not changed.	Loose BMF of the input structure	-

bmflp bemisMurckoFrameworkLoosePruned	Structural Calculations License	Returns the Bemis-Murcko loose framework of the input structure. Calculated by removing side cains. Atom and bond types are generalized by replacing all atoms with carbons and setting all bond types to single. Exocyclic non single bonded atoms are kept as single bonded carbons.	Generalized loose BMF of the input structure	-

Back to Contents

Structure Checker Functions

Name	License	Description	Return value	Parameters	Examples

check	-	checks the structure for errors, according to the configuration	the error report	Structure checker/fixer configuration as action string or XML string	Molecule Context (All) • check("aromaticity.. • check('"<?xml versi encoding=\"UTF-8\" ><checkers><Arom fixMode=\"fix\" fixer chemaxon.fixers.Re type=\"general\" /><ValenceErrorCh allowTraditionalNitr fixMode=\"fix\" fixer chemaxon.fixers.Va /checkers> '") checks for aromatic errors, and returns Reaction Context check(reactant(0), ' valence") checks fo valence errors in fir returns the error rep
checkErrorCount	-	checks the structure for errors, according to the configuration	the total error count	Structure checker/fixer configuration as XML or action string	Molecule Context checkErrorCount('"<?x encoding=\"UTF-8\" sta ><checkers><Aromatic fixMode=\"fix\" fixerCla chemaxon.fixers.Rearo type=\"general\"/><Val allowTraditionalNitroge fixMode=\"fix\" fixerCla chemaxon.fixers.Valer /checkers> '") or check ("aromaticity..valence" aromaticity and valenc returns the total error c Reaction Context check(reactant(0), "aro checks for aromaticity in first reactant, and re report

fix	-	checks the structure for errors, according to the configuration, and then fixes the errors	the fixed molecule	Structure checker/fixer configuration as action string or XML string	Molecule Context (All) • fix("chiralflag..isotop >converttoelementa • fix('"<?xml version= UTF-8\" standalone ><checkers><Chira fixMode=\"fix\" fixer chemaxon.fixers. RemoveChiralFlagF /><IsotopeChecker fixerClassName=\"c ConvertToElementa /checkers>'") searches for ciral fla and removes them elemental form Reaction Context fix(product(1), "chira >converttoelementa for ciral flags and is product, and remov converts them to el

isValid	-	checks the structure for errors, according to the configuration	true, if the structure is valid (has no errors), false otherwise	Structure checker/fixer configuration as action string or XML string	Molecule Context (All) • isValid("aromaticity • isValid("'<?xml vers encoding=\"UTF-8\" ><checkers><Arom fixMode=\"fix\" fixer chemaxon.fixers.Re type=\"general\" /><ValenceErrorCh allowTraditionalNitr fixMode=\"fix\" fixer chemaxon.fixers.Va /checkers> "') checks for aromatic errors, and returns valid Reaction Context isValid(reactant(0), valence") checks fo valence errors in fir returns if it is valid

Back to Contents

Notes

1. If in reaction context in the parameter list of a Chemical Terms function atoms are referred by atom index - including cases when a Chemical Terms function is used to convert reactant or product atom maps to atom indexes - then the molecule (reactant / product) parameter always have to be specified (e.g. shortestPath(reactant(0), pair(ratom(1), ratom (2)))) and must contain all the atoms referred by atom indexes.
2. **Although the function/calculation names are case-sensitive, the lowercase versions are always accepted.** For example, aromaticAtomCount() is equivalent to aromaticatomcount(), but AromaticAtomCount() and AROMATICATOMCOUNT() are not recognized by the parser.
3. match, matchCount, disjointMatchCount and dissimilarity functions are not available in Marvin, they can be used only if JChem software package is installed.
4. Structure based calculations are performed using Calculator Plugins (these calculations are also referred as "plugin calculations").Since Marvin 5.2, all of these functions can have an additional string argument that specifies plugin parameters in "key1:value1 key2:value2 key3:value3 ..." fashion. For example: charge(8, "type:pi pH:3.5") will compute the pi charge of atom 8 in the major microspecies at pH=3.5 of the input molecule, markushEnumerations("code:true max:4","1,2") will generate maximum 4 enumerated

structures with enumeration ID, enumerating only atoms 1 and 2 of the input Markush structure. Note, that this feature cannot be combined with the former possibility of specifying the pH in a string argument, for example: charge(8, "type:pi pH:3.5") cannot be written in the form charge(8, "type:pi", "3.5"), while charge("pi", 8, "3.5") is accepted (backward compatibility).