semi-global, and global alignment. The algorithms work both for proteins and ribonucleic acids, whereas t-RNA molecules are currently not supported because they contain modified bases. This module can be very helpful when used in conjunction with the *AlignMolecules* module.

7.5 Visualizing Molecular Trajectories and Metastable Conformations

A *MolTrajectory* can be visualized via animation of single time steps by attaching a *Molecule* module to the *MolTrajectory* and then attaching a *MoleculeView* or *BondAngleView* to the *Molecule*. The animation is controlled via the *Molecule's Time* port.
For *MolTrajectories* that represent metastable conformations, the modules *MeanMolecule*, *PrecomputeAlignment*, and *ConfigurationDensity* can be used. The *MeanMolecule* module aligns all steps of a trajectory and computes a mean molecule by averaging every atomic coordinate over all time steps. Instead of computing the mean molecule to a reference, as with the *MeanMolecule* module, the *PrecomputeAlignment* module allows you to find the optimal transformation of each time step to minimize the overall sum of squared distances. With the *RankTimeStep* module you can search in a trajectory for a desired time step, using different criteria, such as the *rmsd* value to a given reference.
The *ConfigurationDensity* module gives an impression of the fuzziness of the conformation by computing a probability density for the positions of atoms and bonds within a molecular trajectory. This density can then be visualized with the *Isosurface* and *Voltex* modules.

7.6 Atom Expressions

7.6.1 Overview

Atom expressions are a query language to find and select atoms of certain properties in the molecule for further action. The most important application of atom expressions in Amira is the highlighting section of the *Selection Browser*.
In Amira, a molecule is separated into groups of different levels. Each group contains a set of attributes. (More details of this concept can be found in the description of the *Attribute Editor*). Atom expressions are a simple form of a relational

query language which accesses these attributes.
The simplest form of an atom expression is an *atom specifier*. This is a literal defining a level, one of its attributes, and a condition for this attribute. For example, `atoms/atomic_number=8` defines all atoms whose atomic number attribute equals 8 (i.e., all oxygens).
Such atom specifiers can be combined with logical operators like `AND`, `OR` and `NOT`. For example, `atoms/atomic_number=8 AND NOT atoms/charge=0` will select all charged oxygen atoms.
There are additional operators like `WITHIN`, `BONDED` and `GROUP` which apply certain conditions to coordinates or bond structure. These are explained in section on *operators*.

7.6.2 Grammar

All possible syntaxes of atom expressions are shown in the following grammar. The different literals and operators are further explained in the following sections.

atomExpr →	(atomExpr)
	| NOT atomExpr
	| atomExpr AND atomExpr
	| atomExpr OR atomExpr
	| WITHIN (atomExpr,float)
	| BONDED (atomExpr[,int])
	| GROUP (groupParts)
	| CS
	| atomSpecifier
atomSpecifier →	hierName/[attrName=]ID
groupParts →	groupPart
	| groupPart,groupParts
groupPart →	groupSpecifier
	| groupSpecifier [bondOrderSymbolChar] groupSpecifier
groupSpecifier →	[!] elementSymbol index

7.6.3 Literals

As mentioned in the overview, the simplest form of an atom expression is an atom specifier. An atom specifier consists of three literals: *hierName*, the optional *attrName*, and *ID*.

hierName stands for a name of a hierarchy level (e.g., `residues`). The following abbreviations can be used for the most common levels:
a=atoms, r=residues, b=bonds, s=secondary_structure, c=chains
attrName is optional and specifies the name of an attribute (e.g., `temperature, occupancy, type, ...`) of the given level. If it is omitted, the ID is assumed to specify the attribute name or index as shown by the list command. If an attribute name is given, the ID is assumed to stand for values of the attribute.
To see which hierarchy levels and respective attributes are defined for a given molecule, take a look at the *Color port* which is used in several modules. The right pull-down menu will show all available attributes for the level chosen in the left pull-down menu.
ID specifies an identifier of a member of the given level. If an attribute name is given, ID specifies a value of this attribute. It may contain wildcards such as * (any substring will match) and ? (any single charackter will match). For integer and float attributes ranges can be used by delimiting the boundaries with a colon (i.e. atoms/index=5:10 selects all atoms with an index within this range). If no attribute name is given, the name attribute is used as a default. In this case, specifying a range will select all atoms whose index is between the index of the selected boundaries. (as an example, for a moelcule imported form PDB the residue name is the chain identifier plus the residue index, thus r/L1:L5 selects residues 1 to 5 on the L chain). An exception to this is the atoms level. Molecules in Amira contain an atomic number attribute instead of an element symbol attribute. To select atoms via their element symbol you can simply type *a/element*. Thus the atom specifier for all oxygen atoms *a/O* is equivalent to the atom specifier *a/atomic_number=8*.
Instead of the '=' comparison you can also use the comparison operators $'<','>'$,$'>='$ *and* $'<='$. These, however, are only available for index, integer and float attributes, not for string attributes.
Another atom expression form involving several literals is the *groupParts* expression which is used with the `GROUP` operator. Operators will be explained in the next section.

7.6.4 Operators

Logical Operators
Several *atomSpecifier* combinations can be used in one expression by linking them logically via the operators `AND`, `OR`, and `NOT` (&, |, and !). Priorities can be specified using usual parentheses (and).

`WITHIN(atomExpr,float)`
This operator selects all atoms which are nearer than *float* Å to any of the atoms specified by *atomExpr*.
`BONDED(atomExpr[,int])`
With this operator, all atoms that are recursively connected to any atoms specified by *atomExpr* will be chosen. You can optionally specify an integer value defining the maximal bond steps. If this is omitted, there will be no limit.
`CS`
`CS` specifies all currently selected (highlighted) atoms.
`GROUP(groupParts)`
The `GROUP` operator is a powerful tool to find functional groups by searching for a certain atom and bond pattern in the molecular graph. To define a graph as a search pattern (*groupParts)*, you must divide it into linear pieces of sequential atoms (a *groupPart*). Each atom must be defined by its element symbol and an index which distinguishes it from other atoms with the same symbol but other indices (*groupSpecifier*). Thus, `C1C2C3` would be a *groupPart* consisting of three different *groupSpecifiers* representing a chain of three carbons. This group part can now be combined with another group part by using one of its *groupSpecifiers* as branch point (e.g., `C2O1H1` for a hydroxyl group branching from the second carbon of the chain). At the beginning of each group specifier, there can be an optional '!'. If it is given, the groupSpecifier is only used for matching, but the corresponding atoms will not be selected. (Thus `!C1O1H1` would find the hydroxyl groups without selecting the flanking carbon, which must be given, however, to avoid selecting structures, such as OH groups in H2O). If the atomSpecifier in question appears several times (to define branch points), it is sufficient to mark it with the exclamation mark once.
Consecutive *atomSpecifiers* in a *groupPart* are usually not divided from each other. This means that there must be a bond of any type between the consecutive atoms. If you want to define the bond order further, you can give an optional *bondOrderSymbol*. ('-' for a single, '=' for a double, '#' for a triple and '~' for an aromatic bond).
Take a carboxyl group as an example. To define the group pattern, the central carbon atom (C2) needs to be connected to three atoms: two oxygens (O1, O2), and one other carbon (C1). This can be done in the following way: `GROUP(!C1C2O1,C2O2H1)`. Thus, the hydroxyl group (O2H1) is given as a branch of the CCO chain. There are, of course, several other ways to split this branch into linear pieces which you can easily find yourself. If your molecule contains the bond type attribute, you can also make use of the double bond. Thus the

expression becomes `GROUP(!C1-C2=O1,C2-O2-H1)`.
Notice: The keywords do not need to be in capital letters. Lower case letters, even a combination of lower and upper case letters, works as well.

7.6.5 Shortcuts

Molecular Option also provides pre-defined shortcuts that have been assembled using the previously mentioned syntactical elements. The shortcuts can be found in your local Amira directory in `share/molecules/atomExpr.cfg`, and can be edited and supplemented.
The standard aliases included in the current Amira release are listed in Table 7.1.

7.6.6 Further Examples

- `all`
  selects all atoms.
- `atoms/5-8`
  all atoms whose index is in the range 5 to 8, inclusive.
- `atoms/atomic_number>1`
  all atoms, except hydrogens
- `s/type=helix AND NOT (a/C OR a/N)`
  all atoms which belong to helices, except C and N atoms.
- `r/type=A*`
  all atoms which belong to residues whose type name begins with the letter A.
- `BONDED(a/4 OR a/100,6)`
  all atoms which are connected via at most 6 steps to the two specified atoms
- `WITHIN(r/A11,3.1) AND C`
  all carbon atoms which are not away further than 3.1 angtroms from atoms of residue 11 on chain A
- `GROUP(C1C2C3C4C5C6C1)`
  all cycles consisting of 6 carbons (e.g., cyclohexane).
- `acidic AND helix`
  all atoms of acidic amino acids which belong to helices

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.