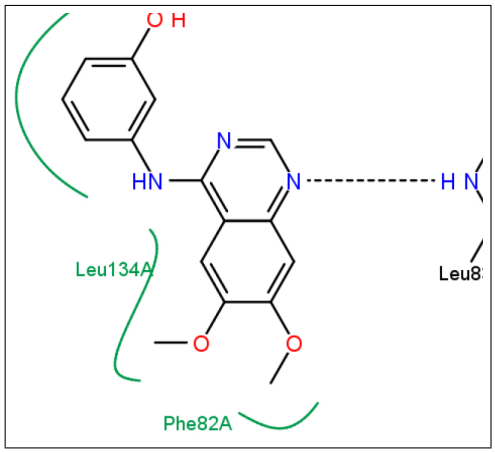
PoseView

2D Sketches of Protein-Ligand Complexes



User Reference
Version 1.9 (updated)

Katrin Stierand
Marcus Gastreich

Fraunhofer Institut Algorithmen und Wissenschaftliches Rechnen

BioSolveIT

Development of *PoseView* has been initiated by Katrin Stierand and Matthias Rarey (ZBH, Univ. Hamburg, Germany). *PoseView* is now available solely through BioSolveIT, St. Augustin, Germany.

Phone ++49-2241-2525-0, support@biosolveit.de

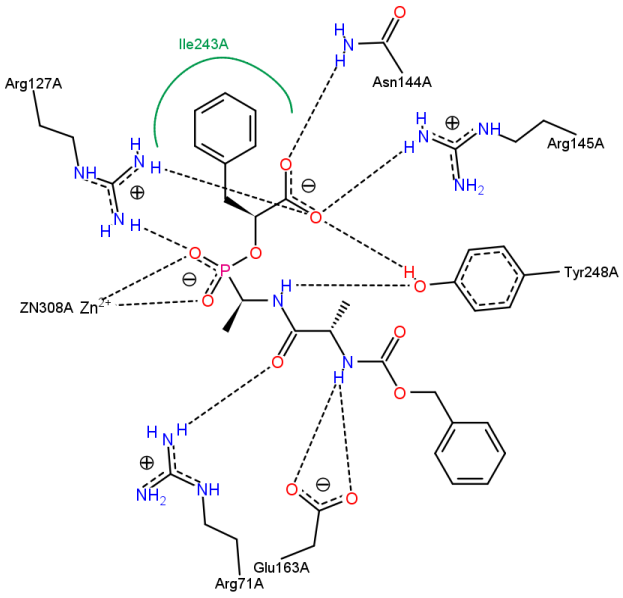
Contents

1 Introduction

1.1 About *PoseView*

PoseView is a computer program which generates 2D pictures of protein-ligand complexes for fast and intuitive perception of interactions; it automatically generates the layouts from given input files.
A diagram contains

- the ligand
- dashed lines representing hydrogen bonds and metal interactions
- the corresponding residues of the protein
- green residue labels for amino acids with hydrophobic contacts to the ligand
- spline segments which highlight the hydrophobic contact areas of the ligand



For details about the implementation and a thorough coverage of *PoseView*'s concept, please see the original publication:

Stierand, K., Rarey, M. (2007)
From Modeling to Medicinal Chemistry:
Automatic Generation of Two-Dimensional Complex Diagrams.
ChemMedChem, Vol. 2, No. 6, pp. 853-860.
http://www3.interscience.wiley.com/journal/114210232/abstract

Stierand, K., Maaß, P., Rarey, M. (2006)
Molecular Complexes at a Glance:
Automated Generation of Two-Dimensional Complex Diagrams.
Bioinformatics, Vol. 22, No. 22, pp. 1710-1716.
http://bioinformatics.oxfordjournals.org/cgi/content/abstract/22/14/1710
Contact: poseview@biosolveit.de

1.2 Installation

1.3 Directory Structures, Files

Unpack the archive. Upon unpacking, there will be a directory structure as below.

```
poseview
|-- examples
`-- static_data
```

The executable name under Linux is `poseview_32` for 32-bit systems and `poseview_64` for 64-bit systems.
You will need to specify where your license resides. There is an extra section on how to do this below (7).

1.4 Additional Files & Libraries

PoseView needs a couple of additional files:

1. A set of ASCII data files containing background knowledge like chemical constants and bond angle patterns. For evaluators, these are usually encrypted and can only be changed after decrypting by BioSolveIT or customers upon request. These files are collected in a folder called static_data.

2. An ASCII configuration file `config.dat`. This file is mainly used to define:

   - where the license resides (optional; see the section on licensing please)
   - where the static data reside (cp. above)
   - under what conditions ligands are read in ("initialized")

   It is commented inside the file. The `tmp` and `|predict|` directories are not used and in there for compatibility reasons. An alternative config.dat file can be specified with the `-c` command line option.

PoseView uses the `cairo` library for complex diagram drawing. If you do not have this library on your system, then it can be downloaded at `http://www.cairographics.org`.

The `doc/` directory contains this documentation (`poseview_ug.pdf`).

1.5 Known Bugs and Limitations

1.5.1 Operating System Dependent Issues

- Linux
  PoseView requires the glibc version 2.3 or later. Please consult systems administration if this error (or alike) occurs:

```
/lib/ld-linux.so.2: version 'GLIBC_2.3' not found (required by poseview)
/lib/i686/libpthread.so.0: version 'GLIBC_2.3.2' not found (required by poseview)
/lib/i686/libc.so.6: version 'GLIBC_2.3' not found (required by poseview)
```

1.5.2 Known Bugs

- With this version, interactions of planar H-bond donors or acceptors may not always be fully complete. We are working to resolve this issue with one of the next versions.
- FlexX Interaction Model deactivated. This model is under heavy development and therefore currently deactivated.
- Splash Screen / Help Screens: There are several minor inconsistencies in the splash and help screens which are due to continuing massive development of the associated engines underneath.

1.6 License Scheme

Our software is license key protected. Please be aware that you cannot run *PoseView* under any circumstances without a valid license.
Please use the form at `https://www.biosolveit.de/license` to request a license for our software or to request further information about the licensing procedure. Alternatively you may mail us at `mailto:license@biosolveit.de`.
To obtain a valid license you must determine and send us the processor or system ID of your computer: To do so, please:

1. Call *PoseView* with the `-i` switch.

   - Linux:
     `poseview -i`

Alternatively, you can use the small ID generation tool 'FlexIDgen' from `https://www.biosolveit.de/download`. If it does not have the 'x' flag, set it with `chmod +x FlexIDgen`.

2. Please send the output to `poseview@biosolveit.de`

3. After you receive your license keys from us, you will need to set your environment variable `BIOSOLVE_LICENSE_FILE`.

   - Linux: Use the `setenv` or `export` mechanisms of your shell.

4. A successfully found license will be uttered like this:

```
...
   ___              __   ___
  | _ \___ ___ __\ \ / (_)_____ __ __
  |  _/ _ (_-</ -_) V /| / -_) V  V /
  |_| \___/__/\___|\_/ |_\___|\_/\_/
...
The BioSolveIT processor ID of machine 'alpha' is 'COMPOSITE=0BD14E0EB2B9'
```

If no license has been found, an error will be output:

```
>> FlexX_base license check (BioSolveIT keys):  failed.
>>    Cannot find license file.
The license files (or license server system network addresses) attempted are
listed below.  Use LM_LICENSE_FILE to use a different license file,
or contact your software provider for a license file.
```

Further details are described below or may be obtained from the BioSolveIT knowledge base page at `https://www.biosolveit.de/faq`.

1.6.1 BioSolveIT License Scheme for Linux/Windows

For the Linux and Windows versions of *PoseView* we use the FlexLM license management system which allows flexible administration of your *PoseView* licenses. Each line of a license file includes the name of the licensed tool or module, and – among other information – the version number and the expiration date of the license.

`Example`

```
INCREMENT PoseView BIOSOLVE 2.0 28-jun-2005 uncounted \
        HOSTID=COMPOSITE=803C749ABB01 SIGN="0078 AD79 1445 A900 \
        906E 00A4 AB51 F600 AFC8 4394 9356 3C87 EBD2 4A7B E88D"
```

You should simply set the `BIOSOLVE_LICENSE_FILE` variable to point to your license file or the directory in which it resides; see above (Sec. 1.6) for Windows instructions.

1.6.2 Running a FlexLM License Server

Instead of specifying a license file directly, it is also possible to serve your *PoseView* licenses from a license server using the FlexLM license management system. Your FlexLM administrator must add the *PoseView* license file to the directory from which FlexLM takes its licenses and add the BIOSOLVE vendor daemon to the directory where the `lmgrd` (license manager daemon) resides. A current version of the license manager daemon and the BIOSOLVE vendor daemon can be downloaded from `https://www.biosolve.it/download`. Use the command

```
lmgrd -c <path_to_licensefiles>
```

to start the license server.
When using a license server, you must inform *PoseView* what the server's name is. You should simply set the environment variable (cp. above) `BIOSOLVE_LICENSE_FILE` to `@myserver`.
Using FlexLM, it is also possible to use *floating licenses*, i.e. licenses that are hosted by a server and distributed to *PoseView* clients on demand. Floating licenses have a special layout. A floating license is always locked to the name or the IP address of the server and its processor ID. Again, this processor ID can be obtained by running FlexIDgen or `poseview -i` on the server.

Example

```
SERVER myserver COMPOSITE=103C749ABB91
USE_SERVER
VENDOR BIOSOLVE
INCREMENT PoseView BIOSOLVE 2.0 28-jun-2005 100 SIGN="00C6 1440 7772 \
        E4A8 116C FFFB EDC8 F400 B648 5413 6ECA 8852 4A2E 29B8 E5D6"
```

In this example a site has 100 *PoseView* licenses. Every time a new instance of *PoseView* is launched the server transfers the license to the application. When *PoseView* finishes its computations the license is returned to the server.

2

Commands and Options

2.1 Calling Help

Calling *PoseView* with the flag `-h` starts the splash screen with a help text:

```
                          ___              __   ___
                         | _ \___ ___ __\ \ / /(_)_____ __ __
                         |  _/ _ (_-</ -_) V /| / -_) V  V /
                         |_| \___/__/\___|\_/ |_\___|\_/\_/

 ZBH - Center for        Automatic generation of 2D complex diagrams
 Bioinformatics
 University of Hamburg   Version:   1.9.1   (26.05.09)
 Bundesstrasse 43
 20146 Hamburg
 Germany                 Authors:   Katrin Stierand, Matthias Rarey

 BioSolveIT GmbH         Copyright: ZBH, University of Hamburg, Germany
 An der Ziegelei 79                 BioSolveIT GmbH, Sankt Augustin, Germany
 53757 St. Augustin      Contact:   poseview@zbh.uni-hamburg.de
 Germany                            www.zbh.uni-hamburg.de/poseview

 For information about additional contributors and copyright notes
 please consult the user guide or type 'help about'.


>>> SYNOPSIS:
    poseviewer [OPTIONS]

>>> OPTIONS:
    -a              : Show complete amino acids
    -c <configfile> : Configuration file. If not specified, PoseView
                      looks for config.dat in the current directory.
                      If not found, PoseView searches in $POSEVIEW_HOME.
    -e              : Lower energy boundary for interactions in percent.
                      0.0 <= value <= 100.0. Default value is 33%.
    -f <filename>   : Read input data from complex file (.mol2)
    -h              : Printing of this help text
    -i              : Generate processor id for this machine
    -l <ligfile>    : The ligand is read from <ligfile>.
                      Default format is .mol2
    -m <file>       : List of ligand and protein file names.
                      Format: <ligfilename> <protfilename> [<outfilename>]
```

```
-o <outfile>    : Prints the diagram to <outfile>
                  possible file formats: .png, .pdf, .ps, .fig
-p <protfile>   : The protein is read from <protfile>.
                  Default format is .rdf
-s <w> <h>      : Size of the .png or .fig output file.
                  png: 100 <= w|h <= 1000, fig 1000 <= w|h <= 50000
-t <text>       : Info text printed in output file.
-v              : Prints the version number and compilation
                  information.
-w <filename>   : Print complex data to file "filename.mol2"
                  Requires -l and -p or -f
```

Most of the options will be self-explanatory. More detailed option descriptions will follow further below.

2.2 Options and command line arguments

2.2.1 `-a`: Show complete amino acids

The drawer will draw interactions to parts of amino acids only, depending on what part of an amino acid forms an interaction. With this switch you can override this behavior and enforce drawing entire amino acids.
An exception are Gly and Pro which are always drawn as entire amino acids.

2.2.2 `-e`: Energy boundary (DEACTIVATED)

If you use the FlexX/-SIS interaction model (cp. p. 18), this switch can be used to suppress certain interaction which do not exceed a certain energy threshold. Because we are currently improving the handling of the model and the model itself, this is currently deactivated.

2.2.3 `-f`: Complex files reading

Please see Section 2.3.1 on p.15.

2.2.4 `-i`: Processor ID Generation

To be able to send you a license file for *PoseView*, we need to know the so-called *BioSolveIT processor ID*. This ID will be reported upon calling *PoseView* with the `-i` flag. Please see the section on installation and licensing to obtain further details on this.

2.2.5 `-l`: Ligand Input

This reads ligands in mol2 and sdf format. The suffix decides. “SD” or “sd” are not accepted as valid sdf format extensions. The extensions are not case sensitive (whereas the filename is, under Linux). Please see p. 15 for the full usage description.

2.2.6 `-m`: List file processing

This option gives you the possibility of processing multiple protein-ligand complexes at the same time. It works as follows:
The argument to `-m` is an ASCII file one or more lines forming an ASCII "table". In every line, entries are separated by a blank or a tab. The first "column" of the file has a contents identifier, being one out of `p`(rotein), `l`(igand), or `f`(ile). The following column contents is then defined by the preceding identifier. Here's an example:
Contents of the file `mycomplexes.txt` (extension does not matter):

```
pl   myproteinA.pdb   myligandA.mol2
f    complexfile.mol2
```

The file `complexfile.mol2` carries the contents of the entire complex in terms of a mol2 file with a special interaction section (cp. Sec **??**).
The optional `outfilename` refers to the picture file written out in a format determined by its extension (pdf, ps, png, fig).

2.2.7 `-o` and `-t`: Outputting images, captions

The default output is the presentation in the integrated GUI browser. The GUI is self-explanatory and gives you plenty of choices to activate or de-activate certain labels, energy values etc.
Possible output file formats are:

- png
- pdf
- ps
- fig (Xfig program format)

Using Xfig you can change the full appearance of *PoseView* images. Within Xfig, you can change fonts, colors etc.

```
poseview -l <ligand> -p <protein> -o <output> [-t <caption>]
```

The caption is optional and defaults to the complex name.
The output format depends on the suffix of the given filename (.png, .pdf, .ps, .fig). Other suffixes will be rejected..
To avoid loading a ligand and a protein more than once it is possible to write a complex file as follows:

```
poseview -l <ligand> -p <protein> -w <complexfile>
```

2.2.8 `-p`: Protein file input

Reading proteins can proceed in two ways:

- Straight through a pdb:
  Using the distance/angle based interaction model, this path parses the pdb file and optimizes the hydrogen positions using a fast and new, internal algorithm.

- Using an rdf file (CURRENTLY DEACTIVATED):
  This mode leaves hydrogens at their positions by default, but enables the user to tweak (add, delete, modify) parts of the protein, to load additional, external mol2 files etc. The rdf file format has its history in FlexX's receptor description file format. Specifications and syntax of this file can be found in the FlexX/-SIS user guides at `https://www.biosolveit.de/FlexX/download/flexx_ug.pdf`.

2.2.9 `-s`: Size of the images

With this option, you specify the size of the image to be created in points. For pdf, this defaults to 300x300 pixels. `w|h` stands for width or height. You may specify both, separated by blanks.

2.2.10 `-t`: Text of your choice in output file

Using this option with a string enclosed by quotation marks (`"`) you can add that very string to the end of your output files.

2.2.11 `-v`: Version information

At times, it is useful for us to know the *exact* version with which you work. The output generated with this option is a verbose block of information which we might request from you during support.

2.2.12 `-w`: Complex file write-out

Please see Section 2.3.1 on p.15.

2.3 Basic Usage, Examples

A protein-ligand complex often comes with a separate ligand file (often in SDF format), and a protein file (usually in PDB format). *PoseView* runs in two different modes which is connected to how *PoseView* determines the interactions (cp. the Sec. 3.1 on p. 17).
N.B.:
At this point it is not possible to use a PDB file which contains both the protein and a co-crystallized ligand in the same file and display them in *PoseView*.

2.3.1 Separate files vs. complex Files

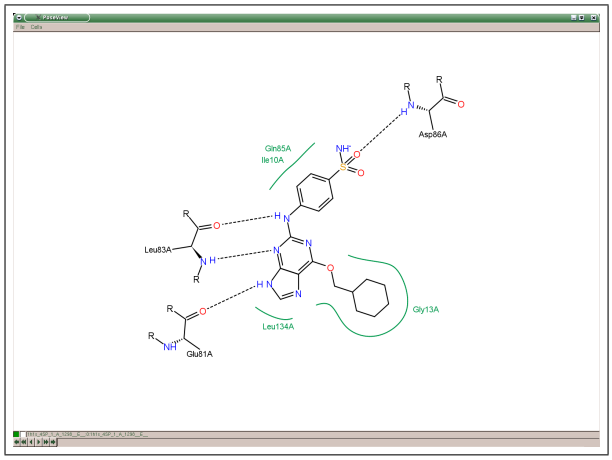
Mode 1: Using separate ligand and protein files

You have two input files, one for the ligand, one for the protein. Here's the quick overview of allowed file formats and some algorithmic details:

- Ligand input file formats: SDF or mol2
- Receptor input file formats: PDB or RDF (FlexX internal format)
- Interactions between ligand and protein are estimated based on geometric criteria
- Interacting residues are cut from the receptor by *PoseView*

poseview `-l ligand_file -p protein_file`

A respective call would accordingly look like this:
poseview `-l examples/1h1s.sdf -p examples/1H1S.pdb`



File formats are detected by the file name suffixes (case insensitive). If no suffix is given, ligand file names are extended by `.mol2` and protein file names by `.rdf`.

Mode 2: A Complex file; the `-f` command line option

A complex file is a multi mol2 format containing all information for one complex

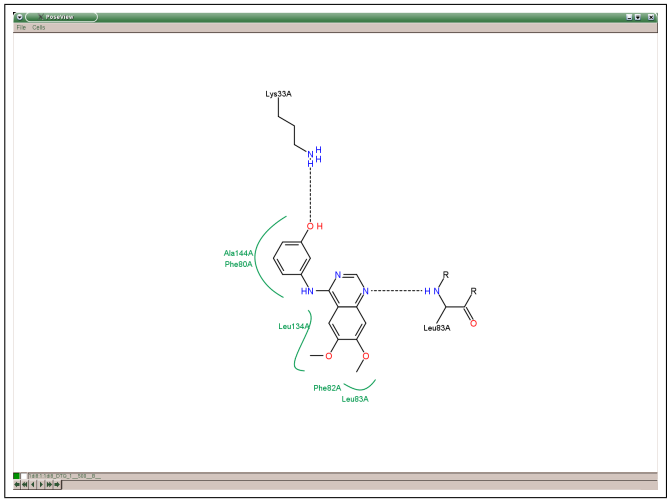
- A comment block contains all needed interaction information (see example below)
- The first molecule has to be the ligand
- The interacting (hydrogen bonds and metal interactions) residues are listed below

poseview `-f multi_mol2_file`

The respective information which *PoseView* needs is written in a special comment section as this header of a file below:

```
@<TRIPOS>COMMENT
ACETYLCHOLINESTERASE (E.C.3.1.1.7) COMPLEXED WITH TACRINE
%AMINO_ACIDS 1
# Format: <mol_id> <mol_nr> <chain_id> <name> <m_id>
2 437 * His440  440
%HYDROGEN_BONDS 1
# Format: <lig_ia_atm> <lig_ia_center> <aa_nr> <aa_ia_atm> <aa_ia_center> <type> <energy>
7 20 437 1 1 0 -4.002900
%HYDROPHOBIC_CONTACTS 2
# Format: <aa_name> <aa_nr> <aa_pdb_nr> <aa_chain_id><nof_cons> <con1> <con2> ... <conn>
PHE 327 330 * 6 1 2 3 4 5 6
TRP 81 84 * 8 3 4 5 6 11 12 13 14
```

This mode is mainly used in environments where another BioSolveIT tool or a corporate/third party tool fills the needed information into a multi-mol2 file. Here is an example for a call:
poseview `-f examples/1di8.mol2`



3

Chemistry

3.1 Interaction Models

From the user perspective, *PoseView* has two tasks:

1. determine where interactions are

2. draw the found interactions in 2D

Since there is no straightforward way of a unique definition of interactions, *PoseView* has several modes to determine interactions:

A) Distance & Angle Based: (vdW = van der Waals)
This model has (currently hard-wired) distance and angle thresholds to determine interactions:

   - hydrogen bonds: There are two criteria which have to be met: If heavy atoms are closer than 1.9$\pm$0.5Å + vdW-radius of the protein atom and the angle between bond to the donor and the interaction is not smaller than 130°
   - hydrophobic interactions: are drawn if the distance of ligand and protein atom is smaller than the sum of vdW-radius(ligand atom), vdW(protein atom), and 0.8Å
   - metal interactions: are drawn if metal and ligand atom have a distance smaller than the sum of 2.4Å and the metal's vdW-radius.

B) Using the FlexX/-SIS Interaction Model (**currently deactivated**):
Whenever the FlexX/-SIS docking engine determines that there is an interaction, that interaction will be drawn by *PoseView* as well. Since the interaction model in FlexX-SIS is This function currently undergoes a re-work and is therefore deactivated.

C) Using your generic description in a so-called *complex-file*:
Using a special section in a preprocessed mol2 file, it is possible to pipe information about where interactions should be drawn to *PoseView* directly. This requires the `-f` option at startup time. There is a commented example of a *complex file* in the `examples` directory for your convenience. Typically this would be used if you would like to have your own interaction model interfaced to *PoseView*.

3.2 Tweaking the Chemistry: Show/Hide Interactions

This section will be about adding or deleting interactions that you would like to see or suppress to see.
The three interaction determination modes have different access to tweaking them.

3.2.1 Distance & Angle Based Model

Currently, the associated distances and angels are hard-wired in the program. We currently collect user feedback on whether this is tolerable, or whether users need to edit these thresholds. Therefore, please do let us know by writing a mail to `poseview@biosolveit.de`.
There are, however, some points at which the user can interfere with the chemistry:

- Ligand:
  The ligands are initialized, i.e., checked for protonation, atom types, aromaticity etc. This is controlled using the static data file `transform.dat`. Please refer to the FlexX-SIS user guide for more information on this mechanism.

- Protein:
  The protein protonation protonation is controlled by the so-called templates. These templates reside in the static data file `amino.dat`. Here, too, we ask you to please refer to the FlexXSIS user guide for more information on the syntax of this file (`https://www.biosolveit.de/FlexX/download/flexx_ug.pdf`).

3.2.2 FlexX/-SIS Interaction Model (CURRENTLY DEACTIVATED!)

Since, with this model, all chemistry used is encoded in the so-called static_data ASCII files, you can change, delete, or add interactions with a little effort.
Interaction specifications in *PoseView* are made on the grounds of the well-known FlexX interaction scheme.
This has two parts to consider:

- The protein

- The ligand

Protein Initialization

Proteins are made of 'known' amino acids, therefore no totally unexpected molecular subgroups are to be expected here. If, however, you have untypical amino acids, or should you want an unusual protonation, then please enter a subgraph and interactions accordingly in the file `amino.dat`. The syntax is described in an older FlexX manual available upon request from BioSolveIT.
In addition to the subgraph-related information, the FlexX interaction model contains the definition of what *types* of protein interactions are compatible with what types on the side of the ligand. The compatibility of such types is set in a static data file called `contype.dat`; we will have more on this below (Sec. 3.2.2.)

Ligand Initialization

Ligands are much more complicated with respect to the variability of their subgraphs. There are three files controlling interactions formed from the perspective of a ligand:

- The so-called interaction types are coded in a corresponding file called `contype.dat` in the `static_data` directory. In this file, you encode classes and/or types of interactions. Usually, you will not even need to modify this file upon adding another interaction because it will be one of acceptor/donor, hydrophobic spherical, or hydrophobic non-spherical.

  In addition to the pure definition, the intercompatibility between these classes are defined by the pipe sign (`|`): Everything on the left side of a pipe is compatible (i.e., may form an interaction) with every interaction type on the right side of the pipe.

- The file `geometry.dat` This is where the interaction geometries are defined. For example, the ketone-O has an acceptor 'surface' with which it can form interactions. If you'd like to add a new geometry, e.g., for a halide acceptor property, you can often copy the geom try of another acceptor. In addition to the FlexX/-SIS syntax of subgraphs, you can also use SMARTS expressions. The very details of the format can be found in the FlexX/-SIS documentation which is freely available on our web page `www.biosolveit.de`.

- The file `contact.dat` Here, the subgraphs are defined which are employed as names in `geometry.dat`. For example, the entry:

```
@subgraph 1 1 acceptor_SO
  atom  1 O.* NOF_BONDS == 1
  atom  2 S.*
  atom  3 R
  bond  1 2 un
  bond  2 3 un
data
  iact  1       2       3       -       h_acc           co
  iact  1       2       -       -       metal_acc       metal_co
end
```

  defines a molecular subgraph (= subgroup) `acceptor_SO` which has an acceptor surface geometry attributed to its oxygen atom in `geometry.dat`.

More documentation about this interaction scheme inspired by LUDI and first used in FlexX, the files `amino.dat`, `contact.dat`, and `contype.dat` is available from BioSolveIT (`https://www.biosolveit.de`).

4

Availability, Technical Remarks

4.1 Operating Systems

PoseView is available for Linux and Windows.

The tool's web page is at: `https://www.biosolveit.de/poseview`