

iMolTalk Workshop II

Alexander V. Diemand
UNIL – University of Lausanne
ISB-SIB – Swiss Institute of Bioinformatics

Agenda

- Update and introduction to new toolchains
- Example demonstrations
- Questions and Answers
- break
- Introduction to MolTalk scripting capabilities
- Example practice

New Toolchains

- Sequence to Structure alignment
 - align a SP sequence to sequence of PDB co-ordinates
 - implications (1:1 relation of SP annotation to PDB residues)
- Structure to Structure alignment
 - implemented algorithms (LSQ fitting, geometrical MAMMOTH alignment)
 - implications (structural alignment and its verification)
- Updated residue contacts search includes image of het. group as provided by MSD (EBI)

Examples

- Sequence alignment of Swiss-Prot entries to PDB co-ordinates derived sequence
- Structural alignment of very distantly related proteins (unknown homology)
- Verification of structural alignment using the newly introduced differential CA contact map
- Search for (structural) domain boundaries using CA contact map



Q & A

- Addressing questions from last meeting
- ?

MolTalk principles

- **Classes** are the meta definition of **Objects** which are the real existing instances. **Classes** are defined within MolTalk while **Objects** are created by your program.
- **Objects** do understand **Messages** (with **Parameters**); this is the only way to talk to **Objects** and achieve action

aChain := someStrx getChain: 65.

- '=' is the predefined assignment operator

MolTalk scripting

- Related to Smalltalk programming language
 - based on GNUstep (open source)
 - purely object-oriented: messaging
 - not typed (int, char*, ...)
 - garbage collection
- Principles (message passing, object creation, memory management, ...)
- Building blocks (syntax, subroutines, loops, ...)
- Examples

MolTalk principles

- Life time of **Objects**: become released when nobody holds a reference to them
- Implicitly created **Objects**:
 - Numbers: **anObject := 333.3 .**
 - Strings: **aString := 'hello world.'**
 - Arrays: **anArr := #(33 'P00508' anObj anotherObj)**

Building blocks

```
[ |
main
| aVar |
  aVar := 3.33 .
  self subRoutine: aVar. “ call ourselves “
!
subRoutine: aParam
  Transcript showLine: aParam description.
  ^self
]
```

Building blocks

- Conditional execution of code blocks:

```
(anObj >= 42) ifTrue: [ result := 'done' ]
ifFalse: [ result := 'not yet' ].
```

- Foreach entry in an array:

```
anArray do: [ :item |
  Transcript showLine: item description ].
```

- Iteration:

```
0 to: 33 do: [ :index | self doIt:index ].
```

Scripting Examples

- Getting more information for structure
 - keywords
 - getSEQRES
- Print connectivity of heterogenous group
- Extract ranges of residues from PDB file, then superimpose them
- Transform structural chain by trafo matrix, then save superimposed structures in one file