2008

Introducing legacy program scripting to molecular biology toolkit (MBT)

Todd Newell

Introducing Legacy Program Scripting to Molecular Biology Toolkit (MBT)

Graduate Project Defense for MS Computer Science Degree

Todd A. Newell

October 28, 2005

Table of Contents

- Introduction
  - Description of Problem
  - Chime, RasMol, Molecular Biology Toolkit (MBT)
  - Goals
- Scripting
  - Definition
  - Background
  - Present in Chime and RasMol, lacking in MBT
- Using MBT for Visualization
  - How MBT is used (2D, 3D, text)
  - Third-Party Applications
- Architecture
  - JavaCC
    - Tokens
    - Grammars
    - Token Manager and Parser
    - LOOKAHEAD
  - MBT
    - Packages
    - Classes
    - StructureStyles and StructureMap
- Execution
- Future Work

Abstract

- Ever-changing landscape of molecular visualization requires a common thread
- Introduction of powerful visualization language, MBT, gives users a comprehensive collection of libraries with the freedom to develop unique applications
- Lacks scripting capabilities, however, something that older, popular languages such as Chime and RasMol have
- MBT lacks handholds for widespread acceptance without scripting
- This project will use JavaCC to parse legacy commands from Chime and RasMol into single, simple commands usable by MBT – scripting
- Bridging to a user-centric and friendly method of use for end-users not entrenched in software development

Introduction

- Description of Problem
  - Visualization software is always changing and it is tough for users to keep up
  - These users are often chemists, biologist, and physicists instead of computer scientists
  - Desire to change with the times comes at a price - users have to learn new application with potentially long learning curve
  - Methods of use should be transferable
    - Scripting
    - .pdb Files

Chime, RasMol, and MBT

- Two popular, older languages have scripting abilities and are widely used
  - Chime: free but incomplete version supplied, not open source.
  - Chime: popular tool leads to many scripts being created
  - RasMol: older and unsupported
  - RasMol: free, open source, command list is subset of Chime’s
  - RasMol: also large number of scripts existing for application
- New toolkit, MBT, incorporates many of the benefits from Chime and RasMol, however not all of them
  - Open source, free
  - Toolkit library that needs applications to render images and interact with molecule
  - Uses .pdb files to input compound data
    - Chemical information listed for each atom
    - .pdb files are the same wherever used

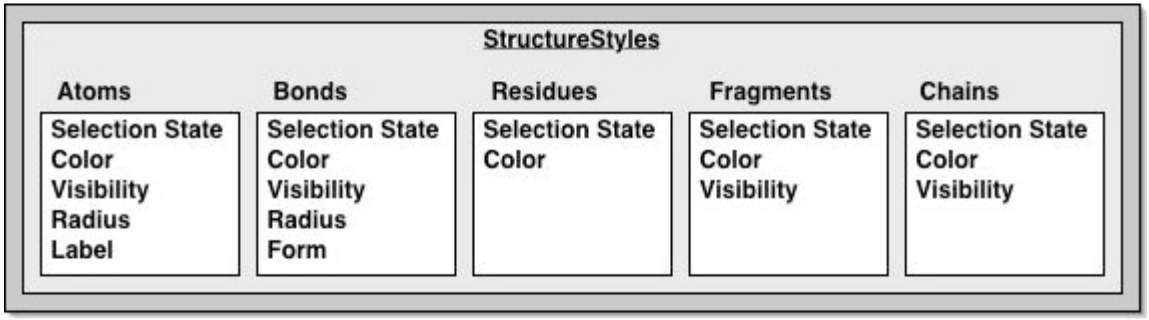
Sample .pdb File

```
ATOM      1  OH  OSP     1      -0.332   0.440  -0.016  1.00  0.00
ATOM      2 1HH  OSP     1       0.596   0.456   0.228  1.00  0.00
ATOM      3 2HH  OSP     1      -0.596  -0.456  -0.228  1.00  0.00
TER       4      OSP     1
END
```

- Type of Component, Atom Serial Number, Atom Name, Coordinates of Atom
- Conforms to standards – identical each time used
- Read by the viewer to manufacture initial view of the molecule.

MBT

- Requires additional application to run
- MBT’s libraries arranged in hierarchy which allows only a few classes to affect molecular components
- StructureStyles.java have hooks into atoms, bonds, residues, fragments, and chains



StructureStyles

Atoms	Bonds	Residues	Fragments	Chains
Selection State Color Visibility Radius Label	Selection State Color Visibility Radius Form	Selection State Color	Selection State Color Visibility	Selection State Color Visibility

Goals

- Last component needed is a parser to evaluate input languages – JavaCC
- Allows us to bridge from older languages to the new, via scripting
- Goals
  - “This project was designed to create a parsing language that accepts scripting commands from two ancestor programs, checking them for correctness, and eventually using the commands to affect the chemical components in the modified molecular visualization toolkit”

Scripting

- What is scripting?
  - Classic method of execution – there is no way to change variables or run methods after compiling without a portal into the program
  - Graphical programs have pull-down menus
  - Scripting allows users to make changes without the knowledge of target applications
  - Easy to remember, accessible commands that run more complex pieces of code

Scripting Example

- Scripting command to select all components in a molecule: “select all”
- MBT method to select all components:

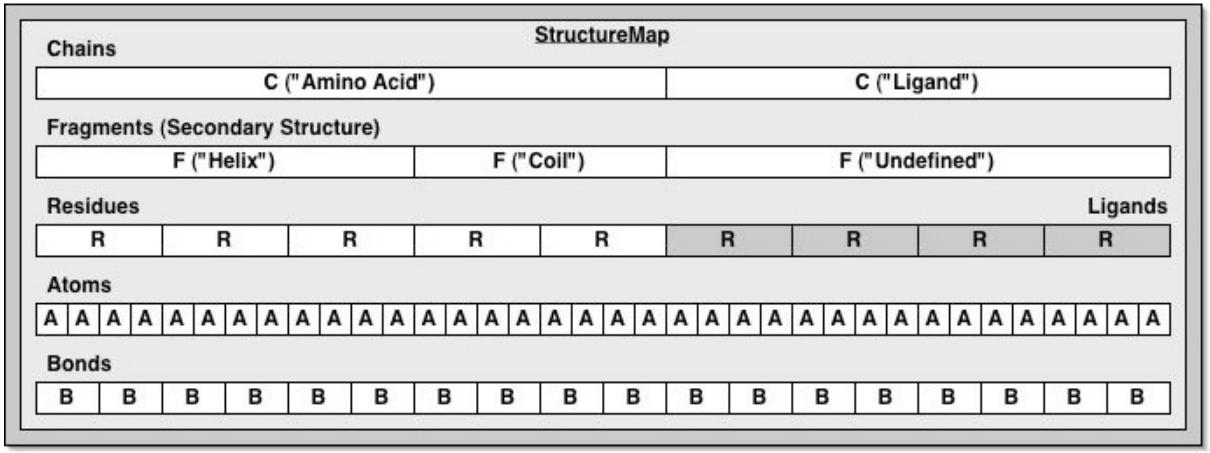
```
public void selectAll() {
  selection.clearAll();
  setAtomSelection( selection.getMin(),
           selection.getMax(), true );
  bondSelection.clearAll();
  setBondSelection( bondSelection.getMin(),
           bondSelection.getMax(), true ); }
```

Importance of Scripting

- Chime and RasMol have scripting ability, while MBT does not
- It was the void we decided to fill – commands that were used to run Chime and RasMol could be made to execute MBT methods
- In this way, users without a comprehensive background in programming can jump to a toolkit like MBT by reusing familiar commands
- Gives the user a simple method of interaction once application has been compiled and executed

Using MBT for Visualization

- MBT is a library and not a free-standing application
- Hierarchies are arranged to be intuitive
- Users can create applications to fit specific needs
- MBT can be used to render and display molecules either 2D or 3D – using Java3D
- Also allows non-graphical manipulation of molecules – loading the same .pdb files lets the user create a virtual representation and change parts of the components without ever seeing the molecule
- Scripting can act on either type of representation



StructureMap

Chains

C ("Amino Acid") | C ("Ligand")

Fragments (Secondary Structure)

F ("Helix") | F ("Coil") | F ("Undefined")

Residues | Ligands

R | R | R | R | R | R | R | R | R

Atoms

A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A | A

Bonds

B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B | B

Architecture

- JavaCC
  - Designed to compile programming languages
  - Can analyze any left-decent grammar
  - Tokens are the lowest common denominator
  - Consume tokens, process them, and analyze them for correctness
  - Tokens act as reserve words that disallow their use elsewhere in the program

Architecture - JavaCC

- JavaCC
  - User defines the grammar which JavaCC program uses to analyze
  - Rules are added as needed – making the grammar as flexible as needed
  - Only requirement: grammar is non-ambiguous so no infinite loops

Architecture - JavaCC

- JavaCC
  - Tokens can be anything – regular expressions, keywords, or literals
  - 2 distinct functions
    - Token Manager – separates commands into tokens (as defined by grammar – whatever size)
    - Parser – examines the rules to try to match tokens, in order, to the functions defined in the program

Architecture - JavaCC

- JavaCC
  - LOOKAHEAD – used to notify the program of a way to avoid ambiguity
  - By specifying number of tokens ahead of consumed token to check, parsing path directed around spots with similar command structures
    - E.g.: LOOKAHEAD(2) color() – two or more functions begin with the same token < COLOR >. Alert parser to check 2 tokens ahead of consumed token, instead of 1
  - Most of the program can operate in a processor efficient manner, only slowing down when meets with stumbling block

Architecture – JavaCC / ChimeParser.jj

```
λ   PARSER_BEGIN( ChimeParser )
    /* IMPORTS – entire MBT library, each package, and any needed Java imports */
        public class ChimeParser {
            /* global variable declaration – must be static */
            private static Explorer explorer = new Explorer();
            public static void main( String args[] ) throws ParseException {
                …
                explorer.initialize();
                try {
                    parser.cmdList();
                } catch ( ParseException e ) {
                    System.out.println( e.getMessage() );
                    System.out.println
                    ( "ChimeParser: Encountered errors during parse.");
                }
            }
        }
PARSER_END( ChimeParser )
```

Architecture – JavaCC / ChimeParser.jj

-
  ```
  TOKEN : /* command list */ {
  < ANIM : "anim" | "animation" >
   | < DIRECTION : "direction" > }
  ```

-
  ```
  void anim() :
  {
          String animStr;
          Token animTok, lastTok;
  }
  {
          animTok = < ANIM >
          animStr = animParser()
          {
                         System.out.println( animTok.image + " "  + animStr );
          }
          [ lastTok = < SEMICOLON >
          {
                        insertSemiColon();
          } ]
  }
  ```

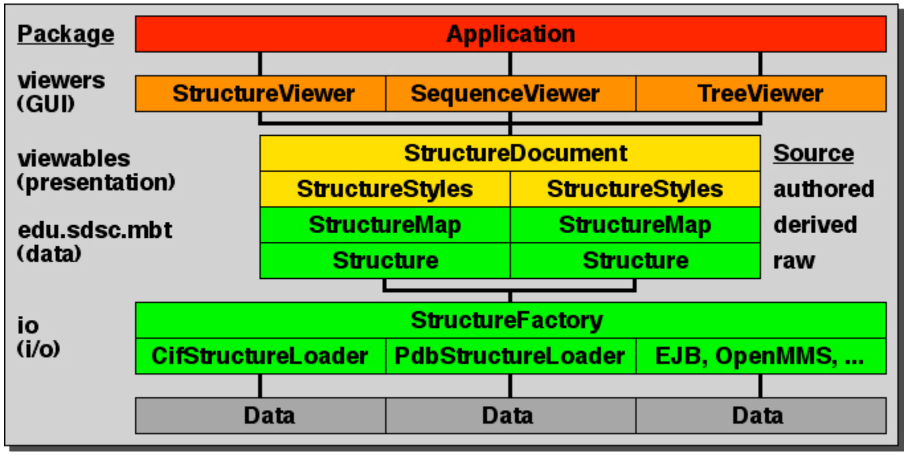
Architecture - MBT

- MBT - packages
  - *edu.sdsc.mbt:* This package provides classes which define the core data storage containers for the MBT toolkit. **StructureMap**
  - *edu.sdsc.mbt.filters:* This package provides classes which enable filtering or subsetting of a structure's constituent components.
  - *edu.sdsc.mbt.gui:* This package provides classes which implement graphical user interface (GUI) component classes for MBT.
  - *edu.sdsc.mbt.io:* This package provides classes which enable molecular biology data sets (such as protein structures, sequence data, etc) to be loaded into an MBT application as a Structure object.
  - *edu.sdsc.mbt.util:* This package provides classes which implement extra functionality on top of the core MBT classes (that is, capabilities that are not absolutely required by, or would otherwise overcomplicate, the core classes).
  - *edu.sdsc.mbt.viewables:* This package provides classes which enable molecular biology data to be represented as visible/renderable viewable objects, plus, this package defines a top-level StructureDocument object to encapsulate the complete state of these objects and properties.
  - *edu.sdsc.mbt.viewers:*This package provides classes which enable data to be graphically visualized or to be processed by other means based upon coordinated events (that is, any MBT component that wishes to observe data and then respond and interact to changes in that data).

Architecture - MBT

- MBT
  - Classes StructureStyles and StructureMap used the most when developing ChimeParser.jj
  - Spans the hierarchy – can make direct calls to the basic components as well as accessing collections
  - StructureStyles – contains most of the methods to make changes such as color, selection, and visibility
    - Method calls directly from Explorer.java : structureStyles.setBondSelection (index, index, true);
  - StructureMaps – edu.sdsc.mbt package and arranges the components after a Structure object is created
    - Accessed whenever need to isolate a component, like the atoms, and check each one in turn:

```
for( int index = 0; index < bondCount; index++ ) {
        Bond bond = structureMap.getBond( index );
}
```



Package

Application

viewers (GUI)

StructureViewer | SequenceViewer | TreeViewer

viewables (presentation)

StructureDocument

StructureStyles | StructureStyles

edu.sdsc.mbt (data)

StructureMap | StructureMap

Structure | Structure

Source

authored

derived

raw

io (i/o)

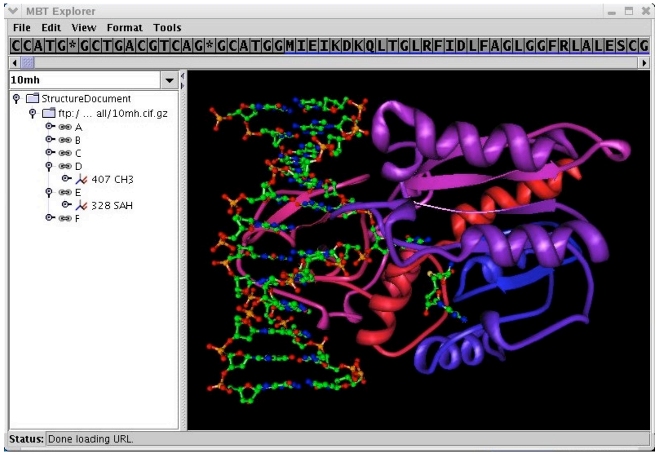
StructureFactory

CifStructureLoader | PdbStructureLoader | EJB, OpenMMS, ...

Data | Data | Data

Architecture - MBT

- Explorer.java
  - The viewer chosen to render the molecule
  - Not fully implemented, but was designed to make basic changes to the molecule
  - Other viewers developed to access specific type of component or chemical interaction (i.e.: hydrogen bonds or visualizing single nucleotide polymorphisms (SNP))
  - Additional methods added to support ChimeParser.jj
    - hbonds(boolean selected)
    - labels(boolean selected)
    - selectAtoms(boolean selected)
    - setBonds(boolean selected) / setAllBonds(boolean selected)
    - setAtomColorByRGB(float[] colors) / setBondColorByRGB(float[] colors)
    - updateAtomColors(StructureStyles ss, boolean selected) / updateBondColors(StructureStyles ss, boolean selected)



CCATG*GCTGACGTCAG*GCATGGMIEIKDKQLTGLRFIDLFAGLGGFRLALESCG

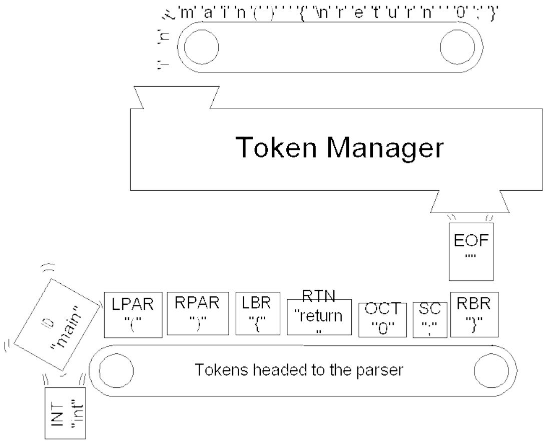
10mh

- StructureDocument
  - ftp:/ ... all/10mh.cif.gz
    - A
    - B
    - C
    - D
      - 407 CH3
    - E
      - 328 SAH
    - F

Status: Done loading URL.

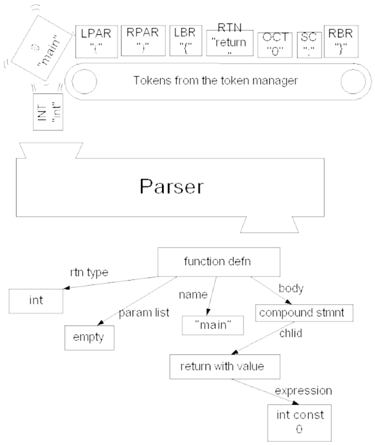
Execution

- ChimeParser.jj – first point of execution, responsible for parsing input commands
  - Running: % java ChimeParser
  - Await input command to begin token manager
  - i.e.: color list {listname} [color] {list of colors} broken down to defined tokens: <color>, <list>, <allowable string regular expression>, [<color>], (<valid color>)+


'i' 'n' 't' ' ' 'm' 'a' 'i' 'n' '(' ')' ' ' '{' '\n' 'r' 'e' 't' 'u' 'r' 'n' ' ' '0' ';' '}'
Token Manager
EOF
""
ID
"main"
LPAR
"("
RPAR
")"
LBR
"{"
RTN
"return"
OCT
"0"
SC
";"
RBR
"}"
INT
"int"
Tokens headed to the parser

Execution

- ChimeParser.jj
  - Parser was then applied to token stream to check for correctness
  - Initializes an Explorer object when executes the main method
  - Drops into the command functions, checking each for possible entrance
    - Once finds matching first token, enters into that command's parsing tree (left-decent), comparing to operating grammar



Execution

- Program execution
  - % javacc ChimeParser.jj
    - Creates 7 .java files:
      - ChimeParser.java – transformation of grammar definitions to Java code
      - ChimeParserConstants.java
      - ChimeParserTokenManager.java
      - ParseException.java
      - SimpleCharacterStream.java – because using default options with ChimeParser.jj
      - Token.java
      - TokenMgrError.java
  - % javac *.java – compiles all .java in directory (including Explorer.java)
  - % java ChimeParser [‘<‘] [<command filename>]
    - ‘<‘: force commands from file to be taken individually from Standard.in
    - <command filename>: file user wishes to parse
    - with no “load” command, commands will echo back to user if correct
  - % <commands user wishes to parse>

Future Work

- Implement full Chime and RasMol libraries
  - Improve the depth of the chemistry
  - Cross-department work to redesign effect of commands on MBT
- Expanding supported input languages
  - PyMol – amazing graphics, interface is lacking
  - Use ChimeParser.jj as backend for any visualization language that conforms to grammar rules (non-ambiguous)
- Artificial Intelligence
  - Protein active site anticipation and extrapolation
  - Study protein interactions with hypothetical molecules
  - Drug research: i.e.: G-protein coupled receptors
- Add scripting ability directly to MBT
  - Eliminate ChimeParser.jj and JavaCC – tokenless?
- Additional implementation techniques
  - Web-based
  - Databases

References

- 1. MDL Chime
  http://www.mdl.com

- 2. RasMol Home Page
  http://www.umass.edu/microbio/rasmol/

- 3. Norvell, Theodore S. (2004). *The JavaCC Faq*. Retrieved from Memorial University of Newfoundland Web site: http://www.engr.mun.ca/~theo/JavaCC-FAQ/javacc-faq.pdf

- 4. JavaCC Tutorial
  http://www.cs.rit.edu/~jmg/javacc/

- 5. The Molecular Biology Toolkit. J.L. Moreland, A.Gramada, O.V. Buzko and P.E. Bourne 2005 The Molecular Biology Toolkit (MBT): A Modular Platform for Developing Molecular Visualization Applications. BMC Bioinformatics, 6:21. Funded by Grant NIGMS 1P01GM63208. Web site: http://mbt.sdsc.edu/

- 6. Kenekin, T. (2005). New Bull’s-Eyes For Drugs, *Scientific American*. October 2005, pp. 50-57.