BIOINFORMATICS APPLICATIONS NOTE Vol. 26 no. 8 2010, pages 1127–1128
doi:10.1093/bioinformatics/btq083

Structural bioinformatics

Advance Access publication February 24, 2010

ESBTL: efficient PDB parser and data structure for the structural and geometric analysis of biological macromolecules

Sébastien Loriot[1], Frédéric Cazals[1] and Julie Bernauer[1,2,*]

[1]ABS, INRIA Sophia Antipolis, 2004 route des Lucioles, 06902 Sophia-Antipolis and [2]AMIB INRIA - Bioinformatics group, LIX, École Polytechnique, 91128 Palaiseau, France

Associate Editor: Anna Tramontano

ABSTRACT

Summary: The ever increasing number of structural biological data calls for robust and efficient software for analysis. Easy Structural Biology Template Library (ESBTL) is a lightweight C++ library that allows the handling of PDB data and provides a data structure suitable for geometric constructions and analyses. The parser and data model provided by this ready-to-use include-only library allows adequate treatment of usually discarded information (insertion code, atom occupancy, etc.) while still being able to detect badly formatted files. The template-based structure allows rapid design of new computational structural biology applications and is fully compatible with the new remediated PDB archive format. It also allows the code to be easy-to-use while being versatile enough to allow advanced user developments.

Availability: ESBTL is freely available under the GNU General Public License from http://esbtl.sf.net. The web site provides the source code, examples, code snippets and documentation.

Contact: julie.bernauer@inria.fr

Received on December 2, 2009; revised on February 3, 2010; accepted on February 20, 2010

1 INTRODUCTION

Robust implementation of new methods is a major challenge in modern computational structural biology. For new ideas and biological hypotheses to be tested *in silico*, programs have to be fast, robust and quick to develop. Using a general-purpose compiled language provides high performance. C++ is thus suitable for computationally expensive techniques and/or large amount of data. The use of template libraries makes C++ a very versatile language, used in a wide variety of applications. Indeed templates allow functions and classes to operate with generic types (so that they do not need to be rewritten to work on a different data type). The success of the Boost C++ library[1] providing a general purpose extension of the C++ standard library shows the wide use of a library extensively using templates among a large community.

Many solutions for handling macromolecules and the PDB format have already been developed. Most of these, such as BTL (Pitt *et al.*, 2001), BioC++ (Zhang, 2010), PDBlib (Chang *et al.*, 1994), were developed some time ago and are either not maintained or do not fully benefit from the features of C++ in terms of genericity and versatility. The MMDB parser developed by RCSB (Ohkawa *et al.*, 1995) and the CCP4 parser (Krissinel *et al.*, 2004) are mainly C-style and are very application specific, making them very difficult to use for other large-scale applications. BALL (Kohlbacher and Lenhof, 2000) is an exhaustive C++ advanced compiled[2] library. Thus it is large and requires several external components. Other recent tools for handling PDB data such as the BioPython PDB Parser (Hamelryck and Manderick, 2003), the MMTK toolkit (Hinsen, 2000), were mainly designed in Python, making the code easy to use but relatively slow because of the use of an interpreter. These libraries were not designed for heavy geometric computations and thus do not provide easy access to geometric constructions such as Voronoi diagrams which are widely used in structural bioinformatics (Poupon, 2004).

Here, we present Easy Structural Biology Template Library (ESBTL) which provides a large variety of template classes and functions to parse PDB data. ESBTL offers a data model suitable for the structural and geometric analysis of biological macromolecules. These can be used with 3D geometric algorithms and constructions such as those provided by the CGAL library[3]. CGAL is intended to geometry algorithm development and provides a wide range of robust geometric code and tools. However its point-based data structures are not suited to represent macromolecular objects (there is no convenient way to attach information to points). Not only does ESBTL provide a robust PDB parser, but also an optional integrated library interface to perform geometric computations using CGAL (triangulation, alpha-shape, skin surface, etc.) in a very simple way.

Prototyping of computational structural biology and biogeometry applications is thus made very easy. ESBTL should fill in the gap for efficient development and be a valuable software tool to the structural bioinformatics community.

2 DESCRIPTION

The data model presented on Figure 1 is implemented by a hierarchical data structure in which each biological structure is represented as a `System` object. A `System` is obtained from a builder and may contain several `Model` objects. Like in other frameworks, following the PDB data structure and the remediated PDB format (Henrick *et al.*, 2008), each `Model` contains `Chain`, `Residue` and `Atom` objects that can be extracted from their parents (`Model`, `Chain` and `Residue`, respectively). Residues can be both of protein and nucleic acid types (the `Residue` name is preferred to a specific nucleotide definition). `Residue` and `Atom` objects can

*To whom correspondence should be addressed.

[1]Boost C++ Library, http://www.boost.org

[2]That is, not include-only.

[3]Computational Geometry Algorithms Library, http://www.cgal.org

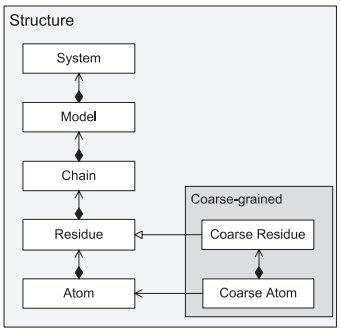

Fig. 1. UML-style schematic diagram of the ESBTL data model.

be both regular and coarse grained such as in bead-based models (Tozzini, 2005). The interface with CGAL is made by providing kernels where atom or pseudoatom can directly be used as point types. Note that CGAL is not required to use ESBTL.

Access to various objects of the hierarchy are provided through direct access and iterators (e.g. atoms of a model and residues of a chain). Native structure properties (e.g. atom types and residue names) benefit from direct access. A classifier allows easy definition and access to specific properties (e.g. atom radii and residue physico-chemical properties). Ready-to-use `System` objects are built by applying a structure selection that can be done at the parsing stage. This allows to load only the interested parts of the molecule. Structures can be read from both PDB files, compressed PDB files (`.gz` and `.bz2`) and `.xtc` trajectory files.

The template structure fulfills most of user-specific needs. This generic design allows to personalize each class used in the hierarchy, coarse-grained models, specific properties, restricted iterators, required PDB fields and so on. For most of general-use code development, shortcut types are supplied to avoid complete template declarations and to provide easy access for beginners.

3 EVALUATION

ESBTL is designed to be robust and will by default reject violations of the PDB format. Indeed during code development we had the opportunity to report a few PDB remediation file inconsistencies.

To evaluate the performance of our library relatively to existing software, we parsed the 4266 files of the PDB_SELECT dataset (Hobohm and Sander, 1994). For each PDB file we extracted the number of models, chains, residues and atoms by iteration. The tests were performed on a Intel Q9550 2.83 GHz CPU. Running times cannot be directly compared as the parsers do not perform exactly the same tasks. The following is just mentioned to provide an order of magnitude.

We performed a reading test on all the file atoms. Unsurprisingly, the Biopython PDB parser is relatively slow as it took 48 min. Using BALL, the performance is much higher as it took roughly 7 min (9 ms per file on average). Our parser needed 8 min (11 ms per file on average) when working on the PDB_SELECT chains and 14 min (18 ms per file on average) to perform the same task with no chain selection switched on.

On the same dataset, we also tried the ESBTL parser in a geometric construction setting. The Delaunay triangulation of atoms were built using CGAL for each PDB file. It took 8 min and 30 s, i.e. 11.5 ms on average per structure file.

4 CONCLUSION

The ESBTL parser and structure library provides a very easy-to-use and powerful library for large-scale structural biology applications. It is very lightweight and its template structure allows to use it as a robust base for efficient software development. On the contrary to many other projects, as it is an include-only library, it is easy to install and does not require a handful of external libraries. Interestingly, a coarse-grained model builder is supplied for multi-level application development. New features will be added to the library and contributions to the project are welcome.

Funding: Partial support from the EU Coordination Action FOCUS K3D (ICT-2007-214993).

Conflict of Interest: none declared.

REFERENCES

Chang,W. *et al*. (1994) Design and application of PDBlib, a C++ macromolecular class library. *Comput. Appl. Biosci.*, **10**, 575–586.

Hamelryck,T. and Manderick,B. (2003) PDB file parser and structure class implemented in Python. *Bioinformatics*, **19**, 2308–2310.

Henrick,K. *et al*. (2008) Remediation of the protein data bank archive. *Nucleic Acids Res.*, **36**, D426–D433.

Hinsen,K. (2000) The molecular modeling toolkit: a new approach to molecular simulations. *J. Comp. Chem.*, **21**, 79–85.

Hobohm,U. and Sander,C. (1994) Enlarged representative set of protein structures. *Protein Sci.*, **3**, 522–524.

Kohlbacher,O. and Lenhof,H.P. (2000) BALL–rapid software prototyping in computational molecular biology. Biochemicals Algorithms Library. *Bioinformatics*, **16**, 815–824.

Krissinel,E.B. *et al*. (2004) The new CCP4 Coordinate Library as a toolkit for the design of coordinate-related applications in protein crystallography. *Acta Crystallogr. D Biol. Crystallogr.*, **60**, 2250–2255.

Ohkawa,H. *et al*. (1995) MMDB: an ASN.1 specification for macromolecular structure. *Proc. Intl. Conf. Intell. Syst. Mol. Biol.*, **3**, 259–267.

Pitt,W.R. *et al*. (2001) The Bioinformatics Template Library–generic components for biocomputing. *Bioinformatics*, **17**, 729–737.

Poupon,A. (2004) Voronoi and Voronoi-related tessellations in studies of protein structure and interaction. *Curr. Opin. Struct. Biol.*, **14**, 233–241.

Tozzini,V. (2005) Coarse-grained models for proteins. *Curr. Opin. Struct. Biol.*, **15**, 144–150.

Zhang,H. (2010) A C++ library for structure bioinformatics applications. Available at http://biocpp.sourceforge.net/ (last accessed date March 5, 2010).