FA5-MS44-P07

Crystallographic Programming Using Haskell, a Functional Language. Jonathan Coome, Michael R. Probert, Andrés E. Goeta, Judith A.K. Howard.
Chemistry Department, University of Durham, UK.
E-mail: jonathan.coome@durham.ac.uk

For many years computer processors have been getting faster, which has meant that by upgrading hardware, software would run faster without any modifications. This is now beginning to change. Processor speeds are reaching a plateau, so manufacturers have instead begun to increase the computation power available by adding more parallel making processing cores. This in turn gives rise to a different problem, because writing programs that can run in parallel is a difficult challenge, especially in imperative-style languages such as Fortran or C++. In these languages, programs are specified as a sequence of instructions, which operate on an implicit shared state. This closely models the underlying hardware, and is efficient when writing single-threaded programs. The shared state however is a hindrance when attempting to adapt programs to run in parallel. Access to variables must be carefully controlled to prevent several parts of the program changing them at the same time, and if this is not programmed very carefully, it can lead to deadlocks in which several threads are each waiting for another to release a lock before they can continue. Functional programming languages offer an alternative [1]. Pure functions have no side effects, and the only way to change the output of a function is to change the function inputs. Apart from small impure parts of the program which must interact with the world, there is no shared state, potentially making parallel programming easier. One such language is Haskell, for which a sophisticated optimising compiler is available [2].
The viability of this approach is currently being tested, including the ease in which parallel programs can be created, and the speed increase that may be obtained when running programs on multi-core processors.

[1] J. Hughes, *Comput. J.*, 1989, **32**(2), 98-107. [2] S. Peyton-Jones, *Journal of Functional Programming*, 1992, **2**, 127-202

Keywords: computing methods, parallel computing, programming

FA5-MS44-P08

RAPIDO - A Web Server for the Rapid Alignment of Protein Structures in the Presence of Domain Motions. Fabio Dall'Antonia[a], Roberto Mosca[b], Thomas R. Schneider[a]. [a]*European Molecular Biology Laboratory c/o DESY, Notkestraße 85, Bldg. 25a, 22603 Hamburg, Germany.* [b]*Institute for Research in Biomedicine, C/ Baldiri Reixac 10, 08028 Barcelona, Spain.*
E-mail: fabio.dallantonia@embl-hamburg.de

The three-dimensional alignment of crystallographic models for related protein structures is the basis for the detailed comparison of such structures. The RAPIDO algorithm [1,2] addresses the task of three-dimensional alignment by a combination of a graphed-based algorithm for the detection of similar regions and a genetic algorithm to connect such regions into larger rigid bodies. In the entire process the different levels of uncertainties of atomic coordinates due to different overall quality of the structures and/or to the different variability of atomic positions within individual models is taken into account via a DPI-based [3,4] approximation.
The algorithm aligns three-dimensional structures in terms of domains in the presence of large conformational changes (e.g. domain motions) between the different molecules. It is also capable of identifying rigid bodies consisting of regions of the polypeptide chain that are distant in sequence but close in space. The algorithm is available for use at: http://webapps.embl-hamburg.de/rapido.
We will present several case studies illustrating the capabilities of the algorithm and the usage of the web-server.

[1] Mosca R., Brannetti B., Schneider, T. R. *BMC Bioinformatics*, 2008, 9, 352. [2] Mosca, R., Schneider, T. R. *Nucleic Acids Res.*, 2008, 36, W42. [3] Cruickshank D. W. *Acta Crystallogr. D*, 1999, 55 (3), 583. [4] Schneider, T. R. *Acta Crystallogr. D*, 2000, 56 (6), 714.

Keywords: Protein structures, Structural Alignment, Domain Motion

FA5-MS44-P09

Software for maintaining and expanding the Crystallography Open Database. Saulius Gražulis[a], Justas Butkus[a], Robert Downs[b], Miguel Quirós Olozábal[c], Armel Le Bail[d]. [a]*Institute of Biotechnology, Laboratory of DNA-Protein interactions, Vilnius, Lithuania.* [b]*Department of Geosciences, University of Arizona, Tucson, USA.* [c]*Departamento de Química Inorgánica, Facultad de Ciencias, Universidad de Granada, Spain.* [d]*Université du Maine, Laboratoire des Oxydes et Fluorures, Le Mans, France.*
E-mail: grazulis@ibt.lt

Increased amount of stored data and growing data rates pose challenges to all scientific databases, and (P)COD (Crystallography Open Database and Predicted Crystallography Open Database) are no exceptions. For a database to be useful, as many errors as possible must be detected in data prior to deposition in a database, and all entries must conform some well-described standard of information encoding. To meet the first goal, (P)COD deposition procedure involves numerous tests on the incoming CIFs[1]. To meet the second, data stored CIF format in (P)COD must be validated against IUCr CIF dictionaries. In this report, we describe our CIF syntax checker, validator and additional tests used to maintain COD data integrity during structure deposition and maintenance of existing entries, and demonstrate how (P)COD is used for subsequent calculations, e.g. powder diffraction patterns for search-match software or chemical substructure search. Since (P)COD pursues an open access model, an emphasis is placed on the automation of data collection and collaboration within crystallographic community using modern computer networks.

[1] Hall, S. R., Allen, F. H. & Brown, I. D., *Acta Cryst.* 1991, A47, 655-685.

Keywords: crystal structure databases, CIF, data checking