JOURNAL OF CHEMICAL INFORMATION AND MODELING

Article

pubs.acs.org/jcim

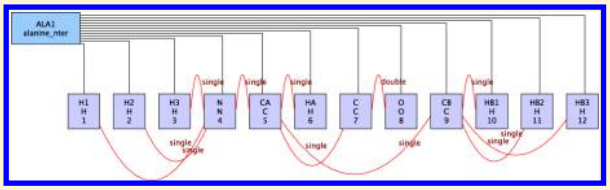
MOSAIC: A Data Model and File Formats for Molecular Simulations

Konrad Hinsen*

Centre de Biophysique Moléculaire (UPR 4301 CNRS), Rue Charles Sadron, 45071 Orléans Cedex 2, France
Synchrotron SOLEIL, Division Expériences, 91192 Gif-sur-Yvette, France

Supporting Information

ABSTRACT: The MOlecular SimulAtion Interchange Conventions (MOSAIC) consist of a data model for molecular simulations and of concrete implementations of this data model in the form of file formats. MOSAIC is designed as a modular set of specifications, of which the initial version covers molecular structure and configurations. A reference implementation in the Python language facilitates the development of simulation software based on MOSAIC.



INTRODUCTION

One of the fundamental notions of science is reproducibility. To qualify as "scientific", a study must be documented in sufficient detail that any competent scientist can reproduce it and verify the results. Reproducibility has received much attention recently because the results of a number of highly visible research studies in various domains could not be reproduced by repeating the published experimental protocols (see, for example, ref 1).

In the computational sciences, many of the uncertainties of experiments do not exist because computers are deterministic devices that produce the same results given the same programs and the same input data, at least in the absence of concurrency. If all programs and input data for a given computational study are published, anyone can *replicate* the study by rerunning the software and thus verify that the published computational protocol is complete and accurate. With the increasing use of concurrency to exploit parallelism, replication can also help to detect nondeterminism. Nondeterminism is undesirable in computational science but still very common in today's parallelized scientific software. Routine replication of computational studies would provide an incentive to software developers to make their code deterministic. Unfortunately, publications that provide programs and input data for replication remain the exception to this day. This is true for computational chemistry as well, as has recently be pointed out.[2]

While replicability is an important measure to ensure confidence in computational results, a notion of reproducibility closer to the one in the experimental sciences is scientifically more interesting. Reproducing the scientifically relevant results of a study using different methods and/or different tools makes it possible to analyze how sensitive results are to the typically large number of parameters that were chosen for convenience rather than after careful consideration of the options. For example, most computational studies on proteins arbitrarily choose one of the popular force fields on the basis of familiarity or availability in a given simulation program. It would often be interesting to try to reproduce the results with a different force field. Moreover, reproduction using different software can protect against artifacts due to mistakes or inappropriate algorithmic choices in a given program.

The technical requirements for reproducibility are stricter than for mere replicability. Reproducibility requires that peers can rerun computations with different software or with modified versions of the model. This is possible only if all the data is available in machine-readable and clearly documented formats and if the mathematical models implemented in the software are clearly documented as well. In many domains of computational science, including computational chemistry, such well-documented formats do not even exist.

The causes that led to this unsatisfying situation are numerous and complex but can ultimately be explained by an insufficiently critical attitude toward computational results in the scientific community. We do not even use software quality, as measured, for example, by the presence of tests, as a criterion for choosing the programs we use.[3] Several high-profile cases of erroneous results due to faulty software[4] have created a growing awareness of the problem and fueled a quest for reproducibility in computational science.[5]

This article addresses a specific aspect of reproducibility, the publication of data in computer-readable formats, and a specific subdomain of computational chemistry, molecular simulation, concentrating on molecular mechanics methods. In such simulations, the computational model consists of point masses representing the atomic nuclei that interact through empirical potential energy functions habitually called force fields. The input to a molecular simulation is the definition of a complete molecular system, called a simulation universe. It specifies the number and molecular structure of all the molecules in the system, its topology (periodic boundary conditions, symmetry, etc.), parameters of the force field, and other parameters

Received: October 17, 2013
Published: December 20, 2013

dx.doi.org/10.1021/ci400599y | J. Chem. Inf. Model. 2014, 54, 131−137

depending on the detailed techniques, such as temperatures or time steps.

In today's typical molecular simulation workflows, scientists prepare a molecular simulation run by writing a set of input files that are specific to a simulation program. The complete specification of the simulation consists of these input files plus the implicit assumptions made by the program, which are often only incompletely documented. Running the simulation program produces a set of output files that contain data useful for later analysis but intermediate data that would be useful for verification and validation is often inaccessible. A scientist wishing to publish a machine-readable specification of a simulation can do so only using program-specific files, which moreover are insufficiently documented.

The possibility to store a complete and unambiguous specification of a molecular simulation and its results in a program-independent format is of interest not only for improving the reproducibility and thus the quality of molecular simulation studies but is also likely to facilitate other developments increasing the reliability of molecular simulations. One example is databases and archives of molecular simulations that permit reuse of existing data, reducing the need to run new simulations for every study. Such archives can also store benchmark simulations for validating new simulation techniques and evaluating their potential advantages.

In the long run, the adoption of an expressive exchange file format is likely to improve molecular simulation techniques in all respects because software development, and thus method development, will be simplified significantly. A future molecular simulation environment could consist of many small independently written software modules that communicate with each other through files. Such an environment allows method developers to concentrate on one aspect of simulations and interoperate with other tools without having to understand and respect conventions of large and complex simulation packages that are usually undocumented.

This article describes a data model called MOSAIC (standing for MOlecular SimulAtion Interchange Conventions) with two file format implementations that cover the most basic kinds of data in molecular simulations: specification of molecules, topology of molecular systems, atomic configurations, and arbitrary quantities defined per atom, such as charges or velocities. The Mosaic specification is publicly available[6] and licensed under the CC BY 3.0 license.[7] The current version 1.0 is also included as Supporting Information to this article, together with three example files. However, the most important contribution of this article is not the final specification but the design choices that lead to this specification. Good design can only be achieved by testing it on realistic applications. A library in the Python language[8] has been developed in parallel with the design of the data model and has been used for several simulation studies of which two have been published together with MOSAIC data files provided as Supporting Information.[9,10]

APPLICATION CONTEXT

The primary application domain for MOSAIC is the simulation of biomolecular systems using a molecular mechanics approach. The only MOSAIC feature that is explicitly related to biological molecules is the possibility to label a molecule as a chain of amino acid or nucleic acid residues. MOSAIC is thus applicable to general molecular systems, including polymers other than peptide and nucleotide chains. There is no explicit limitation to molecular mechanics approaches either, but other levels of description have not been tested.

Most of the popular simulation programs for biomolecular systems, including AMBER,[11] CHARMM,[12] GROMACS,[13] and NAMD,[14] use a similar approach to specifying the system to be simulated, but each of them uses its own proprietary file formats that are documented to different degrees. Users typically start with an experimental structure for a macromolecule in the PDB format.[15] The simulation program reads in this file plus another file containing complementary information about the system, such as other molecules to be added (solvent, ions, etc.), force field choices, and the system topology (e.g., periodic boundary conditions), adds information from a program-internal database containing force field data, and produces another file containing a complete description of the simulated system, usually called a "topology file", although it has no obvious relation to the mathematical concept of topology. Advanced users can modify this file directly in order to fine-tune the system definition. It is then fed into the simulation program again in order to run a simulation, which is specified by additional parameters such as the choice of integration algorithm, step size, etc. In the following, I use the term "system definition file", reserving the term "topology" for its mathematical meaning.

Note that the above procedure is not the only approach to defining a molecular simulation. An alternative one is used, for example, by the Molecular Modeling Toolkit (MMTK,[16]), in which the system definition exists in memory and is manipulated through an application programming interface (API) defined in the Python language.[8] Writing it to a file for archiving is possible but not done routinely. From the point of view of reproducibility, forcing the user to store the system definition in a file is useful because it creates an explicit record of what has been simulated. However, the use of editable text file formats as the only way to define a molecular system leads to other problems, in particular concerning the lack of numerical precision. For example, creating a simulation of a water box following the standard GROMACS procedures leads to a system in which each water molecule has a slightly different geometry because the GROMACS system definition file stores the Cartesian positions of the atoms with insufficient precision.

The system definition files for different simulation programs vary not only in format but also in content, making an automatic translation in general impossible. The main reason for this is the complexity of the force fields, of which one part, the functional form of the interactions, is encoded in the source code of the simulation program, whereas another part, the system-specific parameters, is part of the system definition. Moreover, the frontier between those two parts differs between programs and force fields. Historically, force fields and simulation programs have evolved in parallel and no attempt was made to define a clean borderline. As a consequence, the only fully complete description of the force fields is the program source code.

A particularity of molecular mechanics approaches is their routine application to large systems that can contain millions of atoms. Large simulations lead to large files, making storage efficiency an important criterion. The exclusive use of text-based formats, in particular XML-based formats, is thus not an option. This precludes the adoption of CML,[17] in spite of the numerous advantages that the use of a format with a much wider range of applicability would offer.

dx.doi.org/10.1021/ci400599y | J. Chem. Inf. Model. 2014, 54, 131−137

DESIGN CRITERIA

The most fundamental design criterion for MOSAIC was to design not a file format but a data model with multiple implementations.[18] This decision was motivated by the fact that no single file format can satisfy all the requirements. First, there is an inherent opposition between compact and efficient binary file formats and easy-to-process text file formats. Large data sets, such as molecular dynamics trajectories, require binary storage for efficiency, but applications that require only a small number of protein structures tend to prefer text-based formats for ease of handling. Next, molecular simulation is not an isolated activity. Data needs to be exchanged with software tools from other domains, which is often easiest when the native formats of those tools can be used. Web-based software tools have good support for formats such as XML or JSON, for example, but visualization tools for large data sets prefer portable binary formats such as netCDF or HDF5 instead.

The core of MOSAIC is therefore a data model, that is, a specification of how molecular systems are described in terms of basic data types such as numbers, lists, or trees. In a separate step, concrete implementations define how these fundamental data types are represented in memory or in files. The current MOSAIC specification describes an XML representation and a HDF5 representation. Additional formats, such as JSON, may be implemented in the future. An important feature of this design is that the conversion of data between these implementations is exact as much as possible; no information is lost and no information needs to be inferred heuristically. The MOSAIC library contains a conversion tool that handles this task. Note that there is a technical limit to exact conversion of floating-point data between binary and text formats; binary formats are usually based on a binary representation, whereas text formats prefer a decimal representation. An exact conversion between binary and decimal floating-point number is, however, not possible. For this reason, decimal representations are also insufficient where exact numerical reproducibility is required because all current computing systems internally handle floating-point number in binary.

The second design criterion for MOSAIC was flexibility and possibility for extension. Computational science is a rapidly evolving field, meaning that an overly rigid specification could become obsolete in a few years. MOSAIC therefore defines generic data types rather than very specific ones. For example, instead of defining a data type for "partial charges", MOSAIC defines a data type for "array with one value per atom", with attributes for "data type of a single value", "name of the quantity", and "physical unit". This flexibility increases the risk of incompatibilities between programs, for example, when one program decides to use the name "charge" for what another one calls "partial_charge". A complete specification for storing a particular kind of data, for example, a protein structure from X-ray refinement, must therefore add a set of naming conventions to the basic MOSAIC specification.

The third design criterion was the use of a hierarchical representation of molecular structures motivated by the convenience that such a representation provides for working with complex molecules. For example, a complex of two associated proteins would be described in MOSAIC by a five-level hierarchy: at the top the complex itself, then the two proteins, each of which consists of peptide chains, which in turn consist of amino acid residues made up of atoms. A frequent argument against hierarchical representations is that they are difficult to handle in languages such as Fortran or C that lack built-in dynamic data structures. MOSAIC offers a partial solution in making it straightforward to extract a flat list of atoms without parsing the hierarchy of molecular structures. Applications that have no need to refer to the hierarchy can thus be written without using dynamic data structures.

OVERVIEW OF MOSAIC SPECIFICATION

The following description of MOSAIC aims for clarity rather than completeness. For the details, see the MOSAIC specification published online.[6]

Data Model. A MOSAIC file contains any number of MOSAIC data items. Each data item is identified by a unique name, allowing data items to refer to each other. The current MOSAIC specification defines five item types:

Universe contains the topology of a simulation universe and the structure of all the molecules it contains.

Configuration contains a position for each site in a universe plus geometric parameters of the universe itself, such as box size and shape parameters for periodic boundary conditions.

Property contains the value of a quantity for each atom or site in a universe. The values are numerical arrays (integer or real) of arbitrary shape. A property also has a label describing the values (i.e., "charge" or "velocity") and a specification of the physical units of the values.

Label contains a text string for each atom or site in a universe.

Selection contains a list of indices that define a subset of a universe. The subset can be defined in terms of atoms or sites.

The central data item is the universe. All other data items contain a reference to the universe for which they provide data. The information contained in a universe item can be grouped into two categories: topology and molecular structure. A universe does not contain any geometric data, which is stored in a separate data item, the configuration. This separation reflects the common situation in molecular simulations that a single universe can adopt a large number of different configurations.

The most general universe topology is a one-, two-, or three-dimensional crystal consisting of a unit cell that is repeated periodically in space. Nonperiodic universes can be considered a special case corresponding to an infinite unit cell. The cell shape is described in MOSAIC by a label that can take one of a list of predefined values, such as "infinite" or "cuboid". The cell shape determines the geometric parameters that a compatible configuration item must contain. For example, a cell of shape "infinite" requires no geometric parameters, whereas a cell of shape "cuboid" requires three edge lengths. The current MOSAIC specification defines the cell shapes for three-dimensional crystals: "infinite", "cube" (one parameter), "cuboid" (three parameters), and "parallelepiped" (nine parameters). More cell shapes, for example, for 2D-periodic systems, can be added in future updates to the specification.

In addition to the cell shape, the universe topology contains a list of symmetry transformations, which can of course be empty. Symmetry transformations are required to describe crystalline systems (e.g., 3D crystals used in crystallography or 2D crystals used in electron microscopy) but also artificial crystal-like topologies used in molecular simulations. For example, the "truncated octahedral" topology is described by the combination of a cubic unit cell and a symmetry transformation from a corner to the center of the unit cell. A symmetry transformation is described by a rotation matrix and a translation vector, which

dx.doi.org/10.1021/ci400599y | J. Chem. Inf. Model. 2014, 54, 131−137

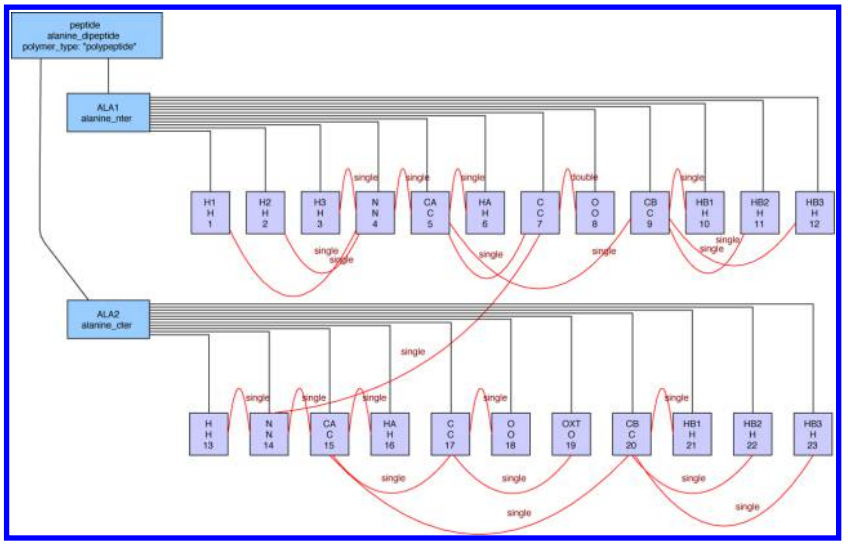


Figure 1. Hierarchical decomposition tree (black lines) and bond graph (red lines) for a two-residue peptide chain. Fragments are colored blue, and atoms are colored violet. The top line in each box is the label of the node, which identifies the node uniquely with respect to its parent node. The next line is the species name (fragments) or element symbol (atoms). The third line for the atoms is the template atom index.

are stored in fractional coordinates in order to be independent of the potentially variable cell size and shape. Symmetry transformations in nonperiodic universes would require additional geometric parameters in the configuration and are therefore not allowed in the current specification.

The contents of a universe are stored as a list of molecules. Each entry of that list has a reference to a molecule template and a repetition count. A molecule template consists of two graphs: a tree defining the hierarchical decomposition of the molecule and the bond graph of the molecule. Figure 1 provides an illustration for a small peptide chain. The term "molecule" is used in a very loose sense; a molecule is a set of atoms plus a set of bonds between these atoms. Both single atoms and supramolecular assemblies not held together by covalent bonds are thus considered special cases of molecules. Nodes in the hierarchical decomposition tree are called "fragments". A fragment specification consists of a species (e.g., "alanine residue"), a label uniquely identifying it inside its parent fragment (e.g., "ALA 11"), a list of child fragments, a list of atoms, and a list of bonds. Fragments describing a polymer also contain a field "polymer type" that can take one of a set of predefined values, such as "polypeptide".

The leaves of the tree describing the hierarchical decomposition of molecular structures are atoms. Again, this term is used in a very loose sense, standing for any point mass used in a molecular model. One of the fields of the atom descriptor is the atom type, for which the current MOSAIC specification allows three values: "element" (for real atoms having a chemical element symbol), "cgparticle" (for point masses in coarse-grained models, including united-atom models), and "dummy" for interaction points that do not represent a physical entity, as used, for example, in the TIP4P water model. The "name" field contains the chemical element symbol for real atoms and a descriptive tag for others. An atom descriptor also contains a label uniquely identifying the atom inside its parent fragment and an integer defining the number of sites required for the atom in a configuration data item. Most applications will use one site per atom, but a higher number can be useful for describing alternate positions (e.g., in crystallographic structures) or nonclassical models such as path integrals.

As mentioned above, each atom or fragment is identified by a unique label inside its parent fragment. This makes it possible to refer to any object in the decomposition tree by a path of labels separated by dots because dots are not allowed inside labels. Such path specifications are used in particular for defining covalent bonds. The bond list for each fragment has one entry for each covalent bond defined by references to the two atoms it connects and a bond order, which can be set to "unknown".

The four nonuniverse data types, configurations, properties, labels, and selections, contain a reference to the universe to which their data applies. Configurations always define three Cartesian coordinates for each site in the universe. Properties and labels offer four levels of descriptions. First, their data can be defined per atom or per site. Second, it can be defined for each atom/site in the system or for each atom/site in the molecular templates, that is, ignoring the repetition count for each kind of molecule. The latter option exists for quantities that are identical for any given kind of molecule, for example, atomic masses or charges.

File Formats. There are currently two implementations of the MOSAIC data model: one based on XML and one based on HDF5.[19] The MOSAIC Python library[20] provides a conversion tool that permits exact conversion in both directions. The choice between the two formats can thus be made according to convenience for a particular application. The XML format is easier to handle and can profit from a wide range of existing XML tools. The HDF5 is more suitable for big data sets and where I/O performance is important.

In the XML format, each data item has an id field whose value must be unique in the file. References to items (in particular universes) use the value of this id field. Because the precision of floating-point data (configurations and properties) is specified in terms of the standard IEEE formats in the data model, XML writers have to use a sufficient number of digits to ensure that any number is reproduced to the requested precision when read back. Floating-point data items can therefore take up a lot of space. The XML format is formally

dx.doi.org/10.1021/ci400599y | J. Chem. Inf. Model. 2014, 54, 131−137

defined by a RELAX NG[21] schema that is part of the MOSAIC specification.[6] This schema permits the validation of a MOSAIC XML file using standard XML validation tools. Such a validation is only partial because XML schema languages cannot express all the constraints of MOSAIC. However, it catches many common mistakes and is a valuable aid in particular for software developers implementing MOSAIC.

In the HDF5 format, universes are represented as an HDF5 group containing several HDF5 data sets. Configurations, properties, and labels are stored as single data sets with all metadata stored in attributes. The two graphs describing the molecular structure are stored in integer arrays, which ensures compact storage and permits straightforward processing in languages without built-in support for graph data structures, such as C or Fortran. All text labels are stored in a single table and are replaced by their index into that table in the graphs. References to other data items are stored as HDF5 object references in a data set's metadata.

Conventions. Neither the MOSAIC data model nor the file formats that implement it completely define how a given molecular system is represented. A MOSAIC file can contain any number of MOSAIC data items, each identified by a name. The names are unspecified, as are many aspects of the hierarchical decomposition of molecular structures. The labels that identify atoms and fragments, as well as the species names of the fragments, can be chosen arbitrarily. MOSAIC does not prescribe that fragments with identical species names should be chemically identical, although in some applications such a criterion would be useful. Finally, complex molecules do not have a single evident hierarchical decomposition. For example, the obvious decomposition of a peptide chain consists of its individual amino acid residues, but some data models further divide the residues into a backbone and a side chain part, whereas others do not.

In practice, the details left unspecified by the MOSAIC definition will be defined by additional conventions inside specific user communities. For example, the PDB converter that is part of the MOSAIC Python library makes several choices in how it stores PDB entries: (1) It maintains the PDB's residue definitions and atom naming conventions. (2) It uses four data items (one universe, one configuration, one property for the occupancy, and one property for the isotropic or anisotropic displacement parameters) with specific names to store crystallographic structures. (3) It uses a variable number of data items for NMR-derived structures: one universe plus as many configurations as there are models in the PDB entries. This convention is documented in the MOSAIC specification[6] as the "PDB convention".

Examples. Three examples of MOSAIC files using the XML representation are provided in the Supporting Information. The file ala_dipeptide.xml serves as an illustration of the hierarchical structure of molecules. It contains a peptide chain consisting of two alanine residues. The file 3ZQ8.xml is a MOSAIC representation of PDB entry 3ZQ8 (a gramicidin channel) generated by the file format converter that is part of the MOSAIC Python library. It contains a universe, a configuration, and two scalar properties for the occupancy and B-factor data. Finally, the file water.xml stores the complete simulation state of a molecular dynamics simulation of a small water box (10 molecules): a universe, a configuration, properties storing the atomic masses and charges and a set of velocities, and a label item storing the atom types for the Amber 99 force field.

IMPLEMENTATION AND APPLICATIONS

Data models and file formats cannot be designed abstractly; their design and implementation has to be validated at all stages using realistic applications. The MOSAIC specification[6] has been developed in parallel with a reference implementation in the Python language[20] and three applications using this library: (1) a converter for entries from the Protein Data Bank[22] to MOSAIC files, (2) a simple molecular viewer for MOSAIC data,[23] and (3) an interface between the Molecular Modeling Toolkit (MMTK[16]), and the MOSAIC Python library, making it possible to generate MOSAIC files from a wide variety of molecular simulations. The PDB converter is published as part of the MOSAIC Python library.[20] It reads mmCIF files from the PDB and adds information from the PDB's Chemical Component Dictionary in order to generate complete molecular structures. The converter can read the majority of existing PDB entries. It fails on a few entries whose interpretation is not obvious from the PDB documentation. It is expected that these cases will be handled in the future after a careful study of the contents of these entries.

MOSAIC has been used successfully in two recent research studies[9,10] whose supporting information contains the complete program code and all data sets in ActivePaper format.[24] One of these studies[9] used 800 protein backbone structures with multiple configurations stored together with the Python scripts working on them in a single HDF5 file.

It is important to note that the MOSAIC reference implementation is not part of the MOSAIC specification but only provided as an aid to understanding and using MOSAIC. The official interface is not the library but the file formats. Users are free to write their own software, in whatever language they prefer, to read and write MOSAIC files. Scientific data has a much higher longevity than scientific software and specific programming languages. A protein structure will be as meaningful in 100 years as it is today, whereas it can be assumed that the Python language will be obsolete by then.

RELATED WORK

The problem of exchanging and publishing computational data exists in all domains of computational science and has led to various research and development projects. In the following, I give a brief overview of projects that overlap to some extent with MOSAIC, although the main goal of all these projects is different.

The Protein Data Bank[22] stores structures of biological macromolecules, which are in fact computational models fitted to experimental data. These structures were initially stored in the PDB format[15] that was specifically developed for this purpose. Because the original format has become insufficient in many ways, the PDB has developed a new format, called PDBx/mmCIF,[25] which will definitely replace the old PDB format in 2016. PDBx/mmCIF and MOSAIC can both be used to describe computational models for molecular structures, and the MOSAIC PDB conventions are designed to make conversion between the two formats straightforward. However, MOSAIC cannot represent the information related to experimental procedures from a PDBx/mmCIF file, and PDBx/mmCIF cannot represent molecular simulation data other than conformations. PDBx/mmCIF also lacks a compact binary representation that would make it suitable for molecular dynamics trajectories.

dx.doi.org/10.1021/ci400599y | J. Chem. Inf. Model. 2014, 54, 131−137

The CCPN data model[26] was developed for handling all aspects of biomolecular NMR spectroscopy, including atomic models for molecular structures. Like MOSAIC, it is a data model definition with multiple implementations, one of which is based on XML files and is suitable for publishing and archiving data. The atomic structure subset of the CCPN data model is very similar to the PDB data model underlying both the old PDB format and the new PDBx/mmCIF format and should also be easily interconvertible with MOSAIC.

The Trajectory NG format[27] defines a compact representation of molecular dynamics trajectories that unlike files in today's popular formats include a sufficiently detailed description of the simulated universe to allow the interpretation of the trajectory without the need for additional files. However, the universe description is designed with proteins and nucleic acids in mind and is not very flexible. More importantly, it cannot be used to store simulation data unrelated to a trajectory.

The Rich Molecule Format (RMF)[28] is an exchange format under development for use by the Integrative Modeling Platform (IMP)[29] and by the visualization software Chimera.[30] It shares many design aspects with MOSAIC, in particular the use of a general hierarchical structure for defining molecules, and the possibility to store a wide range of data items related to molecular simulations. RMF aims at a somewhat different application range (exchange of single molecular structures with attached experimental data), and it is defined at the level of a library API rather than a file format, making it less suitable for publishing and archiving. However, a large common subset of both formats can probably be interconverted exactly.

All the projects summarized above have been implemented and used, at least by their authors. However, only the original PDB format has found widespread acceptance until now. Work on PDBx/mmCIF has started 20 years ago, but it will become the official format of the PDB only in 2014. This illustrates that introducing new file formats is an inherently slow process. One obvious reason is the amount of work required to implement a new format in many software packages. Another reason is that both software developers and users initially see little point in implementing and using a file format that nobody else uses. It is probable that over the last 20 years, more effort has been invested in working around the limitations of the original PDB format than what would be required to switch to PDBx/mmCIF.

OUTLOOK

While MOSAIC is already useful in the current version, more data items will likely be added in the future in order to cover a wider range of needs in molecular simulation. An obvious data item is the trajectory. From the point of view of the abstract data model, defining trajectories is rather straightforward (a sequence of configurations and/or properties), but the implementations need to take into account the large size that trajectories can have and the resulting need for high-performance I/O and data compression. This was the major motivation for choosing HDF5 for MOSAIC's binary file format, as the HDF5 library offers compression and parallel I/O. The H5MD trajectory file format[31] is also based on HDF5, and a combination of H5MD for trajectory data and MOSAIC for storing the universe and various static data items is currently under development.

In a more long-term perspective, it is desirable to have a lower-level description of molecular mechanics force fields in MOSAIC. As one of the examples in the Supporting Information illustrates, it is straightforward to encode typical force field parameters such as atom type labels or partial charges in MOSAIC. However, their interpretation requires that the corresponding force field is implemented in the software that reads the file. Most simulation programs offer only one or two force fields, making comparisons between different programs difficult. Storing a low-level description of the interactions in a MOSAIC file would allow a separation of the energy-term assignment part and the energy evaluation part in typical molecular simulation programs. The assigned energy terms, that is, an explicit list of harmonic bond terms, harmonic bond angle terms, etc., would be stored with the universe, and a different simulation program could reproduce them without containing specific support for the force field being used.

An important question about any file format meant for exchanging and archiving data is whether it will be adopted by a sufficiently large community. In the case of MOSAIC, this will depend to a large extent on the importance attributed to replicability and reproducibility by the molecular simulation community. To the best of my knowledge, there are currently no other file formats, published or under development, that could replace MOSAIC as a means of publication of molecular simulation data. I hope that scientists and software developers interested in reproducibility will contribute to the further development of MOSAIC rather than develop competing formats. The Open Source development model, the use of the Github platform, and the independence of any existing simulation software should help to encourage the formation of a community around MOSAIC.

ASSOCIATED CONTENT

Supporting Information

Version 1.0 of the Mosaic specification, example files in MOSAIC format. This material is available free of charge via the Internet at http://pubs.acs.org.

AUTHOR INFORMATION

Corresponding Author

*E-mail: konrad.hinsen@cnrs-orleans.fr.

Notes

The authors declare no competing financial interest.

ACKNOWLEDGMENTS

This work was supported by the Agence Nationale de la Recherche (Contract No. ANR-2010-COSI-001-01).

REFERENCES

(1) Harris, C.; Coburn, N.; Rohrer, D.; Pashler, H. Two failures to replicate high-performance-goal priming effects. *PLoS ONE* **2013**, *8*, e72467.

(2) Walters, W. P. Modeling, informatics, and the quest for reproducibility. *J. Chem. Inf. Model.* **2013**, *53*, 1529−1530.

(3) Joppa, L. N.; McInerny, G.; Harper, R.; Salido, L.; Takeda, K.; O'Hara, K.; Gavaghan, D.; Emmott, S. Troubling trends in scientific software use. *Science* **2013**, *340*, 814−815.

(4) Merali, Z. Computational science: ...Error. *Nature* **2010**, *467*, 775−777.

(5) Peng, R. D. Reproducible research in computational science. *Science* **2011**, *334*, 1226−1227.

(6) Hinsen, K. MOSAIC Specification. http://github.com/mosaic-data-model/mosaic-specification (source code, accessed December 10, 2013); http://mosaic-data-model.github.io/mosaic-specification/ (HTML version for reading, accessed December 10, 2013).

(7) CC BY 3.0, Creative Commons, Attribution 3.0 Unported License. http://creativecommons.org/licenses/by/3.0/ (accessed December 10, 2013).
(8) Python Software Foundation, The Python programming language. http://www.python.org/ (accessed December 10, 2013).
(9) Hinsen, K.; Hu, S.; Kneller, G.; Niemi, A. A comparison of reduced coordinate sets for describing protein structure. *J. Chem. Phys.* **2013**, *139*, 124115.
(10) Chevrot, G.; Hinsen, K.; Kneller, G. R. Model-free simulation approach to molecular diffusion tensors. *J. Chem. Phys.* **2013**, *139*, 154110.
(11) Case, D. A.; Cheatham, T. E.; Darden, T.; Gohlke, H.; Luo, R.; Merz, K. M.; Onufriev, A.; Simmerling, C.; Wang, B.; Woods, R. J. The Amber biomolecular simulation programs. *J. Comput. Chem.* **2005**, *26*, 1668−1688.
(12) Brooks, B. R.; et al. CHARMM: The biomolecular simulation program. *J. Comput. Chem.* **2009**, *30*, 1545−1614.
(13) Pronk, S.; Páll, S.; Schulz, R.; Larsson, P.; Bjelkmar, P.; Apostolov, R.; Shirts, M. R.; Smith, J. C.; Kasson, P. M.; van der Spoel, D.; Hess, B.; Lindahl, E. GROMACS 4.5: A high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics* **2013**, *29*, 845−854.
(14) Phillips, J. C.; Braun, R.; Wang, W.; Gumbart, J.; Tajkhorshid, E.; Villa, E.; Chipot, C.; Skeel, R. D.; Kalé, L.; Schulten, K. Scalable molecular dynamics with NAMD. *J. Comput. Chem.* **2005**, *26*, 1781−1802.
(15) Worldwide Protein Data Bank, Protein Data Bank Contents Guide: Atomic Coordinate Entry Format Description, Version 3.3. http://www.wwpdb.org/documentation/format33/v3.3.html (accessed December 10, 2013).
(16) Hinsen, K. The Molecular Modeling Toolkit: A new approach to molecular simulations. *J. Comput. Chem.* **2000**, *21*, 79−85.
(17) Murray-Rust, P.; Rzepa, H. S. CML: Evolution and design. *J. Cheminform.* **2011**, *3*, 44.
(18) Hinsen, K. Caring for your data. *Comput. Sci. Eng.* **2012**, *14*, 70−74.
(19) The HDF Group, Hierarchical Data Format, Version 5. http://www.hdfgroup.org/HDF5 (accessed December 10, 2013), 2000-2010.
(20) Hinsen, K. MOSAIC Python Library. http://github.com/mosaic-data-model/mosaic-python (accessed December 10, 2013).
(21) OASIS, RELAX NG Home Page. http://relaxng.org/ (accessed December 10, 2013).
(22) Berman, H.; Henrick, K.; Nakamura, H. Announcing the worldwide Protein Data Bank. *Nat. Struct. Biol.* **2003**, *10*, 980.
(23) Pellegrini, E.; Aoun, B.; Goret, G.; Hinsen, K. MOSAIC Molecular Viewer. http://github.com/mosaic-data-model/mosaic-viewer (accessed December 10, 2013).
(24) Hinsen, K. ActivePapers, Python Edition. http://www.activepapers.org/python-edition (accessed Decemebr 10, 2013).
(25) PDBx/mmCIF. http://mmcif.wwpdb.org/ (accessed December 10, 2013).
(26) Vranken, W. F.; Boucher, W.; Stevens, T. J.; Fogh, R. H.; Pajon, A.; Llinas, M.; Ulrich, E. L.; Markley, J. L.; Ionides, J.; Laue, E. D. The CCPN data model for NMR spectroscopy: Development of a software pipeline. *Proteins* **2005**, *59*, 687−696.
(27) Spångberg, D.; Larsson, D. S.; Spoel, D. Trajectory NG: Portable, compressed, general molecular dynamics trajectories. *J. Mol. Model.* **2011**, *17*, 2669−2685.
(28) Rich Molecule Format, RMF. http://salilab.github.io/rmf/index.html (accessed December 10, 2013).
(29) Integrative Modeling Platform, IMP. http://www.integrativemodeling.org/ (accessed December 10, 2013).
(30) UCSF Chimera. http://www.cgl.ucsf.edu/chimera/ (accessed December 10, 2013).
(31) de Buyl, P.; Colberg, P. H.; Höfling, F. H5MD: A structured, efficient, and portable file format for molecular data. arXiv preprint 1308.6382, 2013.