Mosaic Documentation

Release 1.0.0

Konrad Hinsen

December 09, 2013

CONTENTS

CHAPTER

ONE

THE MOSAIC DATA MODEL

1.1 Mosaic data items

Mosaic data items are the smallest units of information that can be stored in files. A Mosaic file can contain any number of data items, each of which has a unique identifier. Identifiers are text strings whose exact specification may vary between file formats. ASCII encoded identifiers are allowed in all file formats and are therefore preferred.

1.1.1 Some definitions used in the following

int8, int16, int32, int64 A signed integer occupying 8, 16, 32, or 64 bits in memory.

uint8, uint16, uint32, uint64 An unsigned integer occupying 8, 16, 32, or 64 bits in memory.

bool A truth value that is either True or False.

float32 An IEEE single-precision floating point number, occupying 32 bits in memory.

float64 An IEEE double-precision floating point number, occupying 64 bits in memory.

label A text string in ASCII encoding containing at most 32767 characters which are each an upper or lower case letter, a digit, or one of the punctuation characters in the string "!#$%&?@^_~+-*/=,()[]"'. This includes all the ASCII punctuation characters except for the dot. Spaces and control characters are not allowed.

list An ordered collection of items. Corresponding data structures in programming languages are typically called list, vector, or array.

set An unordered collection of items, in which each item can occur at most once.

array An n-dimensional ordered collection whose elements are of identical type.

atom A point-like particle in a molecular simulation. May represent a real atom, a united-atom particle in a coarse-grained model, or a dummy interaction point with no physical properties.

site A point in space related to an atom. An atom has at least one site. Atoms with multiple sites can be used for representing quantum models (path integrals, wave functions, ...), atoms with multi-modal position distributions (as used in crystallography), etc.

fragment A hierarchical structure consisting of atoms, other fragments (subfragments), and bonds. A fragment loosely corresponds to the concept of a functional group or moiety in chemistry, but its definition covers a much wider range of chemical structures: the extreme use cases for fragments in Mosaic are single atoms and whole molecules.

template The definition of the molecular structure for a species of molecules in a universe. Any fragment can be used as a molecule template. It is important to distinguish between a molecule and a molecule

template: every molecule has exactly one template, but a template can describe a large number of molecules. For example, in a box of water molecules, all molecules share a single template.

1.2 Data item type "universe"

The universe is the most central Mosaic data item because all the other ones require a reference to a universe, because their data contents can only be interpreted meaningfully in the context of their universe.

A universe describes a molecular system, defining

1. the chemical structure of the molecules it contains
2. the topology of the whole system (periodic boundary conditions etc.)
3. symmetries, if required

A Mosaic universe contains:

- a cell shape field, whose value is "infinite", "cube", "cuboid", or "parallelepiped"
- a convention field, whose value is an ASCII-encoded text string naming a convention for atom names, decomposition of standard groups (e.g. amino acid residues) into subgroups and atoms, etc.
- a (possibly empty) set of symmetry transformation. Each symmetry transformation is defined by a rotation matrix and a translation vector. The full system consists of the explicitly represented atoms and molecules plus their images obtained by applying all the symmetry transformations. Symmetry transformations are defined in fractional coordinates and therefore allowed only for periodic universes.
- a list of molecules, each molecule being defined by a (fragment, count) pair, where count is a positive integer. Fragments are defined below.

A Mosaic fragment is not a data item, because it cannot be written to a file in isolation. Fragments exist only as part of a universe definition.

A fragment contains the following information:

- a label field, whose value is a label that identifies the fragment uniquely inside its parent fragment (if any parent fragment exists)
- a species field, whose value is a label that describes the chemical entity described by the fragment
- a boolean field is_polymer, whose value is "true" if the fragment describes a polymer. A polymer fragment has an empty atom list, i.e. it contains only sub-fragments. A polymer fragment also has an additional polymer_type field, whose value is "", "polypeptide", "polyribonucleotide", "polydeoxyribonucleotide", or "polynucleotide". The empty string is used for polymers that are not of any other type, or for polymers of unknown type.
- a (possibly empty) list of sub-fragments
- a (possibly empty) list of atoms
- a (possibly empty) set of bonds

An atom is described by:

- a label field, whose value is a label that identifies the atom uniquely inside its parent fragment. Each label inside a parent fragment can name an atom //or// a sub-fragment, but not both.
- a type field, whose value is "element", "cgparticle", "dummy", or "". The empty string is used for any type other then the explicitly named ones, and for atoms of unknown type. The type "element" refers to a physical atom with a well-defined chemical element. The type "cgparticle" refers to

coarse-grained particles that represent several physical atoms. The type "dummy" refers to interaction sites that have no physical reality.

- a name field, whose value is a label that describes the chemical nature of the atom. For atoms of type "element", it must be the chemical element symbol, with the first letter upper-case and the second letter, if one exists, in lower-case.
- a number of sites field, whose value is a positive integer.

A bond is described by (1) a set of two atom references and (2) a bond order specification, whose value is "", "single", "double", "triple", "quadruple", or "aromatic". The empty string is used for bonds of any other order, or for bonds of unknown order. Bonds must be defined at the level of the smallest possible fragment that includes both atoms implied in the bond. In other words, it must be possible to check if two atoms in a fragment are linked by a bond without looking at parent fragments.

An atom reference is an ASCII-encoded text string naming an atom relative to the current fragment by the sequence of labels that define the path to the atom. The labels in the sequence are separated by a dot.

1.3 Data item type "configuration"

A configuration contains:

- a reference to a universe
- one position vector for each site in the universe
- for universes with a bounded cell, the parameters of the cell, stored as an array whose shape is determined by the universe's cell shape: an empty shape vector (i.e. the array is a scalar) for "cube", shape (3) for "cuboid", and (3,3) for "parallelepiped".

The elements of the position vectors and the cell parameters are floats of the same precision, either float32 or float64.

1.4 Data item type "property"

A property contains:

- a type field, whose value is "atom", "site", "template_atom", or "template_site"
- a reference to a universe
- one array (see details below) for each
  - atom in the universe, if the type field is "atom"
  - site in the universe, if the type field is "site"
  - atom in the molecule templates, if the type field is "template_atom"
  - site in the molecule templates, if the type field is "template_site"
- a name field, whose value is a label
- a units field, see details below

The arrays for each atom or site have identical shapes and their elements identical types. The type can be int8, int16, int32, int64, uint8, uint16, uint32, uint32, uint64, float32, float64, or bool.

Properties of type "atom" or "site" are defined for each atom or site in the universe. Properties of type "template_atom" or "template_site" are defined for each atom or site in the molecule templates of the universe. They are thus identical for the corresponding atoms or sites in each molecule sharing the same template.

The value of the units field is a text string in ASCII encoding. It contains a sequence of unit factors separated by a space. A unit factor is either a number (an integer or a decimal fraction) or a unit symbol optionally followed by a non-zero integer which indicates the power to which this factor is taken. Examples:

- "nm3" stands for cubic nanometers
- "nm ps-1" stands for nanometers per picosecond
- "60 s" stands for a minute

Each unit symbol may occur only once in the units field. There may also be at most one numeric factor, which must be the first one.

The following unit symbols may be used:

Length	pm	picometer
	Ang Ångström	
	nm nanometer	
	um micrometer	
	mm millimeter	
	m meter	
Time	fs	femtosecond
	ps picosecond	
	ns nanosecond	
	us microsecond	
	ms millisecond	
	s second	
Mass	amu	gram/mole
	g gram	
	kg kilogram	
Quantity	mol	mole
Energy	J	joule
	kJ kilojoule	
	cal calorie	
	kcal kilocalorie	
	eV electron-volt	
Temperature	K	Kelvin
Pressure	Pa	pascal
	kPa kilopascal	
	MPa megapascal	
	GPa gigapascal	
	atm atmosphere	
	bar bar	
	kbar kilobar	
Electrical units	e	proton charge
	C coulomb	
	A ampere	
	V volt	
Angles	deg	degree
Constants	c	speed of light
	h Planck constant	
	me electron mass	

Note that the only unit for angles is the degree. Contrary to SI recommendations, angles are taken to be dimensionless in Mosaic. This corresponds to how angles are treated *de facto* in computational science. The unit "deg" is thus a dimensionless conversion factor equal to $180/\pi$.

1.5 Data item type "label"

A label contains:

- a type field, whose value is "atom", "site", "template_atom", or "template_site"
- a reference to a universe
- one text string in ASCII encoding for each
  - atom in the universe, if the type field is "atom"
  - site in the universe, if the type field is "site"
  - atom in the molecule templates, if the type field is "template_atom"
  - site in the molecule templates, if the type field is "template_site"
- a name field, whose value is a label

Labels of type "atom" or "site" are defined for each atom or site in the universe. Labels of type "template_atom" or "template_site" are defined for each atom or site in the molecule templates of the universe. They are thus identical for the corresponding atoms or sites in each molecule sharing the same template.

1.6 Data item type "selection"

A selection contains:

- a type field, whose value is "atom", "site", "template_atom", or "template_site"
- a reference to a universe
- an array whose values are the indices of the atoms or sites that are part of the selection.

The index array is one-dimensional and the type of its elements is one of the unsigned integer types: uint8, uint16, uint32, uint32, uint64. The indices are stored in monotonously increasing order with no index being listed more than once.

Selections of type "atom" or "site" contain indices for atoms or sites in the universe. Selections of type "template_atom" or "template_site" contain indices for each atom or site in the molecule templates of the universe. They are interpreted as a selection of all atoms or sites that correspond to the selected template atoms or template sites.

CHAPTER

TWO

MOSAIC XML FILES

The Mosaic XML format is defined by a Relax NG schema. The following explanations document additional constraints that cannot be encoded in a schema.

2.1 Mosaic data items in an XML file

A Mosaic XML file contains a top-level element with tag `mosaic` whose children are Mosaic data items. Each of these data items has a required attribute `id` that gives it a name through which it can be identified uniquely inside the `mosaic` element. No two data items in the same top-level element may have the same `id` value.

References to a data item take the form of an empty element with the same tag as used for the definition of the data item itself. The empty element contains a single attribute `ref` whose value is the unique identifier of the data item that is referenced.

2.2 Floating-point numbers

Configurations and real-valued *properties* contain floating-point data. The Mosaic specification allows two floating-point data types, "float32" and "float64", which are based on IEEE binary floating point formats. In XML, numbers are usually stored in decimal form. Since an exact conversion between binary and decimal floating-point numbers is not possible, a compromise must be chosen. Since a main reason for using XML is its human-readable layout, Mosaic's XML format uses a standard XML decimal representation, at the cost of non-exact conversion from and to binary data layouts.

Programs writing Mosaic XML files should convert floating point values using the largest possible number of digits and then remove trailing zeros from the mantissa. They should not attempt any rounding. The symbols "NaN", "+inf" and "-inf" should be used for non-numbers, although such values rarely make sense in the context of molecular simulations.

2.3 Array data

Property data items contain one array value per atom or site. The N-dimensional array values in a property are stored as a single N+1-dimensional array, whose leading dimension is the number of atoms or sites. Likewise, the positions in a *configuration* data item are stored as an array of shape (M, 3), where M is the number of sites.

The elements of an array are stored as a single list in row-major order.

CHAPTER

THREE

MOSAIC IN HDF5 FILES

HDF5 files contain a tree structure whose leaves are datasets. Non-leaf nodes are called groups and work much like a directory in a file system. Each dataset is an array whose elements can be numbers or characters, but also compound data types (similar to record types in various programming languages) or fixed-size arrays of numbers or characters. Groups and datasets can have metadata tags called attributes. Each group or dataset can be identified by a path specifying how to reach it from the root group of a file. However, a path is not necessarily unique, because HDF5 provides links that effectively put a single node in several places in the tree. Moreover, HDF5 provides a data type "reference" that allows to refer to a node or a subset of a dataset.

The design criteria for the HDF5 representation of Mosaic data were efficiency of storage and ease of use from low-level languages such as C or Fortran. As much as possible, Mosaic data is stored as arrays of numbers.

HDF5 has two string layouts: fixed-size strings (character arrays) and variable-length strings. The two layouts are not interchangeable. In order to facilitate software development, Mosaic uses only variable-length strings.

3.1 Mosaic data items in a HDF5 file

A Mosaic data item in an HDF5 file can be a dataset or a group containing multiple datasets. It is identified by four attributes, all of which are required:

DATA_MODEL a variable-length string with the value "MOSAIC"

DATA_MODEL_MAJOR_VERSION an integer

DATA_MODEL_MINOR_VERSION an integer

MOSAIC_DATA_TYPE a variable-length string

References between data items are stored as attributes whose value is an HDF5 object reference.

3.1.1 Universes

A universe is stored as a group containing several datasets. The datasets *convention* and *cell_shape* are variable-length strings. The *symmetry transformation* list is stored in dataset `symmetry_transformations` as a one-dimensional array, possibly empty, whose elements are of a compound data type with fields

rotation A 3x3 array of float64 numbers.

translation An array of 3 float64 numbers.

The fragment tree is stored in several arrays. All string values are stored in a one-dimensional array dataset `symbols` whose elements are variable-length strings. In the fragment tree, the strings can then be replaced by integers, which are indices into this symbol list. Ideally, each string is stored only once in the symbol array, though this is not a

requirement. The tree data structure is stored in two integer arrays, `fragments` and `atoms`. Each node (fragment or atom) has one array entry, which is a compound data type whose fields are unsigned integers. Any size of unsigned integer can be used, but the same type must be used everywhere for a given universe group.

The entries of the `fragments` array have the fields

parent_index The index of the parent node in the `fragments` array. A value of 0 indicates a root node, which has no parent.

label_symbol_index The index of the *label* in the `symbols` array.

species_symbol_index The index of the *species* in the `symbols` array.

number_of_fragments The number of sub-fragments. This is redundant information, provided to facilitate reading fragment-related information without analyzing the whole fragment tree.

The first entry (index 0) of the `fragments` array is unused, in order to allow an index value of 0 to stand for "no parent". The `fragments` array thus has one more entry than the number of fragments in the universe.

The entries of the `atoms` array have the fields

parent_index The index of the parent node in the `fragments` array.

label_symbol_index The index of the *label* in the `symbols` array.

type_symbol_index The index of the *type* in the `symbols` array.

name_symbol_index The index of the *name* in the `symbols` array.

number_of_sites The *number of sites*.

The entries of the `bonds` array have the fields

atom_index_1 The index of the first atom in the `atoms` array.

atom_index_2 The index of the second atom in the `atoms` array.

bond_order_symbol_index The index of the bond-order label in the `symbols` array.

The entries of the `molecules` array have a large number of redundant fields (all but the first two) that are provided to allow atoms be attributed to molecules without analyzing the full fragment tree.

fragment_index The index of the fragment node in the `fragments` array.

number_of_copies The number of copies of the molecule in the universe.

first_atom_index The index of the first atom in the `atoms` array.

number_of_atoms The number of atoms in the molecule.

first_bond_index The index of the first bond in the `bonds` array.

number_of_bonds The number of bonds in the molecule.

first_site_index The index of the first site of the molecule.

number_of_sites The number of sites in the molecule.

Since the atoms, sites, and bonds of a molecule have consecutive indices, the redundant "first_index" and "number_of" values are sufficient to locate atoms, sites, and bonds for each molecule. For many applications this is sufficient, making it unnecessary to use the `fragments` array.

Finally, the array `polymers` has one entry for each polymer fragment in the universe. Its fields are

fragment_index The index of the fragment node in the `fragments` array.

polymer_type_symbol_index The index of the polymer-type label in the `symbols` array.

If the universe has no polymer fragments, the dataset `polymers` may be omitted.

3.1.2 Configurations

A configuration is stored as a group containing two datasets: `positions` (required) and `cell_parameters` (required if the universe's cell shape is not "infinite"). The reference to the universe is stored in the attribute `universe` of the group.

The dataset `positions` is a one-dimensional array whose length is equal to the number of sites in the universe. Its elements are one-dimensional arrays of length 3 whose elements are of type "float32" or "float64".

The dataset `cell_parameters` is an array whose elements are of type "float32" or "float64", and whose shape is defined in the *specification*.

3.1.3 Properties

A property data item is stored as a dataset that is a one-dimensional array whose length is equal to the number of atoms or sites in the universe or the universe's fragment list. Each element of this array is an array whose shape and element type is defined by the property's *data*. The *reference* to the universe is stored in the attribute `universe` of the group. The property's *name* and *units* are stored in attributes of the same name as variable-length strings. The property's *type* is stored in the attribute `property_type`, also as a variable-length string.

3.1.4 Labels

A label data item is stored as a dataset that is a one-dimensional array whose length is equal to the number of atoms or sites in the universe or the universe's fragment list. Each element of this array is a variable-length string. The *reference* to the universe is stored in the attribute `universe` of the group. The label's *name* is stored in the attribute `name` as a variable-length string. The label's *type* is stored in the attribute `label_type`, also as a variable-length string.

3.1.5 Selections

A selection is stored as a dataset that is a one-dimensional array of integers. The *reference* to the universe is stored in the attribute `universe` of the group. The selection's *type* is stored in the attribute `selection_type` as a variable-length string.

Conventions:

CHAPTER

FOUR

MOSAIC PDB CONVENTION

Mosaic can be used to store molecular models from the Protein Data Bank (PDB). The main application is to use such models as the starting point for molecular simulations. The following conventions describe how a PDB structure is stored in terms of Mosaic data items. Note that only the structure itself can be stored, but not experimental data (structure factors etc.) or metadata describing the experiment or the refinement process.

The PDB's official data format is called PDBx/mmCIF. In the conversion from PDBx/mmCIF to Mosaic, as much information as possible is transposed without modification. In particular, residue and atom names are the same.

4.1 Crystallographic structures

A crystallographic structure is represented by two required data items:

- A universe defining the molecular structures and, in the case of crystals, the symmetries. The atoms in the universe have multiple sites if the PDB structure contains alternate locations.
- A configuration providing the positions for all sites and the shape of the unit cell in the case of crystals.

Additional information from the PDB entry can be provided by optional data items:

- The occupancy of each site can be provided as a *property* of *type* "site" or "template_site" with an empty *units* string. Each value is a scalar of type "float32" or "float64" in the interval [0..1]. If no occupancy values are provided, the occupancy of all sites is assumed to be 1.
- An anisotropic displacement parameter for each site can be provided as a *property* of *type* "site" or "template_site". A valid *units* string must be provided, the preferred units are "nm2". Each value is an array of shape "6" and of type "float32" or "float64", the order of the elements is [1][1], [2][2], [3][3], [2][3], [1][3], [1][2]. For the precise definition of the anisotropic displacement parameters, see the PDB documentation for items `_atom_site.aniso_U[1][1]` to `_atom_site.aniso_U[3][3]`.
- An isotropic displacement parameter for each site can be provided as a *property* of *type* "site" or "template_site". A valid *units* string must be provided, the preferred units are "nm2". Each value is a scalar of type "float32" or "float64". An isotropic displacement parameter of value `x` is equivalent to an anisotropic displacement parameter of value `[x x x 0 0 0]`.

If anisotropic displacement parameters are provided, then no isotropic displacement parameters may be given, in order to prevent incoherencies in the data.

4.2 NMR structures

An NMR structure is represented by the following data items:

- A universe defining the molecular structures.
- One configuration per model contained in the PDB entry.
- *genindex*

INDEX