Biochemfusion

Binary molecule file format

version 1.0

2012-12-18

Table of Contents

Introduction

An industry standard way of saving a molecule graph to a file is the MDL molfile format.[1] The MDL molfile format is relatively verbose and as a result it compresses fairly well, typically to ~20% of original size when using ZLib compression.[2]

However, writing molecule data uncompressed to a binary file format can produce molecule files with a size of only ~12% of a corresponding MDL molfile (2D-only). So a binary optimized format is both smaller than a compressed molfile and has far less processing overhead.

The Biochemfusion binary molecule format is such a format and this document describes how it is structured.

1 Data format

The data format is structured in blocks:

```
molecule header
count info
atom block
bond block
data block(s)
EOF marker
```

1.1 Molecule header

The molecule header is 5 bytes consisting of 4 byte identifier string plus a version-and-format byte. The identifier string is always "BCFM".

'B'	'C'	'F'	'M'	ver | fmt

The version-and-format byte's upper 4 bits contains the version number (1 - 16) and the lower 4 bits specify the size of the *mol_integer* type in bytes (1, 2, or 4). This document describes version 1 only.

The *mol_integer* type is an unsigned integer of variable size. Its size is chosen to be the minimal size needed to represent atom/bond-indices and -counts:

- 1 byte for molecules with up to 255 atoms/bonds
- 2 bytes for molecules with up to 65535 atoms/bonds
- 4 bytes for really large molecules (up to ~4E9 atoms/bonds)

Atom- and bond-indices are always zero-based.

1.2 Count info

The count info is two consecutive *mol_integer*s: the atom count followed by the bond count.

atom count	bond count

The count info determines how many atom and bond records will be read from the atom and bond blocks.

1.3 Atom block

The atom block consists of a number of 8-byte records, each record defining the X and Y coordinate of an atom followed by its atomic number. The format does not support atom Z-coordinates, at least not in this version.(see however 2.3)

x	y		a-no
32 bits	16 bits	byte	byte

X and Y coordinates are 28-bit signed integers (3½ bytes each); the atomic number is a single-byte integer. R-group atoms will have an atomic number of 0.

The two 28-bit integers are written as a 32-bit integer followed by a 16-bit integer followed by a byte. The X-coordinate's 28 bits are located in the most significant 28 bits of the 32-bit integer. The Y-coordinate has its 4 most significant bits in the 4 least significant bits of the 32-bit integer; bits 23-8 are in the 16-bit integer and the 8 least significant bits are in the byte.

1.4 Bond block

The bond block is a number of 3-, 5-, or 9-byte records, each record defining a bond via its from- and to- indices and its bond-type.

from-index	to-index	card | stereo

The from- and to-indices are *mol_integer*s. The bond-type is a single byte with its upper 4 bits determining bond cardinality (1 = single, 2 = double, and 3 = triple) and its lower 4 bits determining stereo information (7 = down bond, 8 = no stereo, 9 = up bond).

1.5 Data blocks

The data blocks are optional blocks with information on R-group atom labels, attachment points, and atom charges.

Each data block has a two byte header: A single byte (usually an uppercase letter) that defines the block type, followed by a single-byte unsigned integer block byte count.

The block is followed by *n* identically sized records, with *n* * record size = block byte count.

Multiple blocks of the same type are allowed to support the case where there are too many records to fit into the maximum block size of 255 bytes; e.g. more than 127 charged atoms.

Data blocks may come in any order.

1.5.1 R-group atoms data block

The block type is 'R' and records consist of a *mol_integer* atom index followed by an R-group label which is a single non-zero unsigned integer byte.

'R'	block byte count	atom index	R-group label	atom index	R-group label	...

1.5.2 Attachment points data block

Attachment points and their numbers are listed in a 'A' type data blocks. Each record contains a *mol_integer* atom index, the atom's attachment point number is a single non-zero unsigned integer byte.

'A'	block byte count	atom index	attach. point no.	atom index	attach. point no.	...

1.5.3 Charged atoms data block

Charged atoms are listed in 'C' type data blocks. Records consist of a *mol_integer* atom index followed by a single byte signed integer charge value.

'C'	block byte count	atom index	charge	atom index	charge	...

1.5.4 Other data block types

Additional block types can be stored but may be skipped safely by readers by first reading the block header and then skipping *n* number of bytes according to the block byte count given in the block header.

1.6 EOF marker

Data is terminated by an EOF byte (ASCII 26).

2 Data structure details

2.1 Endianness

Since the majority of popular computer platforms today is little-endian the file format is also little-endian - least significant bytes are stored first.

2.2 A note on the chosen coordinate format

The reason for using 28-bit integers to represent atom coordinates is that this is both a more compact and more faithful representation than using 32-bit floats.

The MDL molfile format specifies coordinates as fixed point values in the format "`xxxxx.xxxx`" - 9 significant digits. One digit will be used for the sign character in case of negative values so the effective precision is 8 significant digits. Since the coordinates are fixed point values we can easily transform them into integers without information loss.

A signed 28-bit integer spans the values -134E6 .. +134E6 which is just above 8 significant digits (100E6). This means that atom coordinates of an MDL molfile multiplied by 10 000 can safely be stored in 28-bit integers without loss of precision.

In comparison, a 32-bit float only has 24 bits precision and will therefore lose information when the coordinates have more than 6-7 significant digits. This error will be insignificant in practice, but nevertheless it is both less precise and more verbose to use 32-bit floats instead of 28-bit integers.

2.3 Z-coordinate support

The majority of MDL molfiles are 2D-only. To keep the binary format simple and compact Z-coordinates can be stored in an optional data block if they must be stored.

The block type identifier for optional Z-coordinates is recommended to be 'Z'. The block record data could be a *mol_integer* atom index followed by the atom's Z-coordinate transformed to a 32-bit signed integer.

3 References

(1) The MDL molfile format was defined by the company MDL, now aquired by Accelrys. The molfile specification is public and available at http://accelrys.com/products/informatics/cheminformatics/ctfile-formats/no-fee.php.

(2) zlib is a very widely used lossless compression / decompression library. It can be downloaded from http://www.zlib.net/.