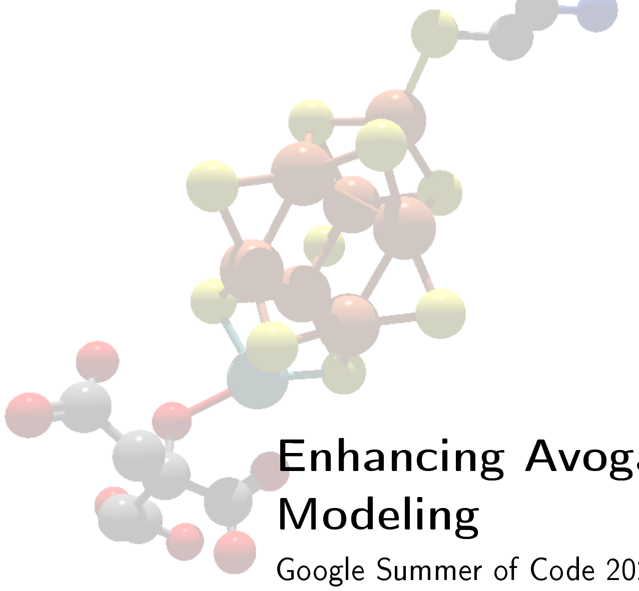


Enhancing Avogadro for Biomolecular Modeling

Google Summer of Code 2022 project proposal

Aritz Erkiaga

Personal Details

Name:	Aritz Erkiaga Fernández
Email:	aerkiaga3@gmail.com
Country & Timezone:	Spain, GMT+1
School Name & Study:	University of the Basque Country, Medicine
Instant Messaging:	@aerkiaga:matrix.org (Matrix)
Instant Messaging:	@aerkiaga, *https://t.me/aerkiaga* (Telegram)
Personal Website:	*https://wordsmith.social/aerkiaga/* (blog)
GitHub profile:	*https://github.com/aerkiaga*

Synopsis

Having casually contributed to other projects in the past, and after learning of the new, relaxed requirements for GSoC 2022, I set out to work on Avogadro, one application I frequently use, throughout the summer. My project will be improving support for macromolecules in the program, which will involve visualization improvements and work on specific molecular features.

1 Project Details

Specs & Scope

This project's goal will be enabling Avogadro as a tool for bioinformatics, specifically for biomolecule visualization and drug design. This should continue the line of work explored by recent GSoC projects. The concrete tasks I propose are:

- Implementing surface visualization and at least one surface plugin useful to molecular binding visualization, such as charge density, hydrophobicity, binding pockets...
- Giving Avogadro the ability to render at least one kind of intermolecular interaction in a predictive manner (e.g. hydrogen bonds, salt bridges, pi stacking...).
- Adding enough support for metal coordination complexes and quaternary ammonium groups so they don't cause trouble when encountered by plugins. Many external files tend to contain both of these, especially the former, as they are quite common in macromolecular structures.
- Stabilizing PDB file import. Currently, it produces malformed structures for some files. Fixing other bugs that affect workflow is also included here.
- Refactoring and performance work wherever required or appropriate.

The heaviest portion of the work is expected to revolve around surfaces and intermolecular interactions, which will require a significant amount of code to be produced. I have already been introduced to the current status of those features. I'll implement surface generation using the algorithm by Hermosilla *et al.*[10], in principle as a purely CPU-based implementation.

I will get the surfaces to display other features by color; charge density is the ultimate candidate, as it is directly applicable to the field of organic chemistry. A close second, subject to be implemented if time allows it, is hydrophobicity; while it's not difficult to tell apart more or less hydrophobic areas on sight, it's beneficial to have a quantitative representation at one's disposal.

Intermolecular interactions, like surfaces, could impose a bottleneck in productivity if not carefully implemented. My main tool to deal with algorithmic issues is binning atoms, bonds, etc. into orthohedral areas, which exploits locality and can bring time complexity down to $O(n)$ provided an upper limit on interaction distance.

Regarding the specific features to consider, as well as their implementation details, I have chosen hydrogen bonds as my target, which properties are already relatively well-known. I plan on developing a simple heuristic model based on a 1987 article[11] on hydrogen bond geometry, combining the usual approach of filtering F, O, N atoms with the hybridization feature present within Avogadro's code.

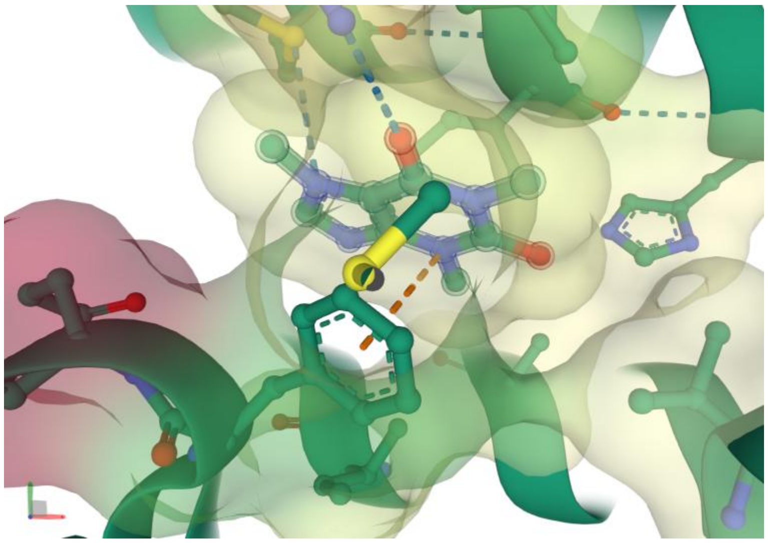


Fig. 1: caffeine bound to its target, the adenosine 2A receptor, as seen with the RCSB PDB ligand viewer[1]. A translucent surface, colored by residue hydrophobicity, has been enabled.

Metal coordination and charge support should be comparatively easy, mainly expected to involve a large number of self-contained changes. However, they are still an important milestone in the path towards a complete biomolecular modeling solution. There is a more in-depth explanation of these changes in 'Groundwork'.

Anticipated Challenges

The major concern for this problem is the likely eventual realization that one or more of the proposed tasks require substantial refactoring for adequate completion. This is particularly expected with formal charge perception support, and will be addressed as follows:

- Communication and thorough planning to figure out the optimal implementation scope; what to change and what to simply work around.
- Outlining most of the refactoring during the design phase, to avoid ineffective work.
- Following good coding practices to avoid having a negative impact on the codebase, especially regarding stability.

Groundwork

The fact Avogadro can't, in many cases, handle formal charges is cumbersome. Regarding this project, the main implication is the inability to work with qua-

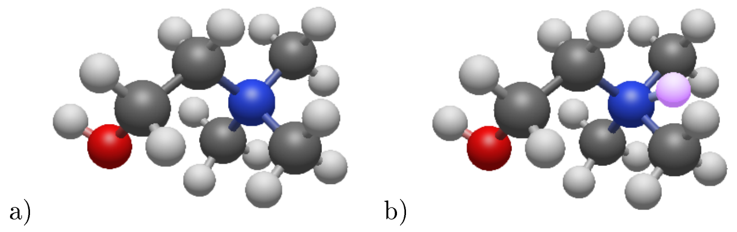


Fig. 2: a) Choline molecule on Avogadro, inserted through SMILES. b) Here, the hydrogens have been added automatically. The one extra atom is highlighted in pink.

ternary ammonium moieties, which are frequently encountered in biological systems. For instance, observe how Avogadro fails to add hydrogens properly to a choline molecule (Figure 2).

Most of the file formats that Avogadro can work with do indeed support formal charges[**?**, 3, 4, 5], as well as the Open Babel API[6] and SMILES[7]:

Open Babel	SMILES	CJSON	CML	MMTF	PDB	XYZ
Yes	Yes	No	Yes	Yes	No	No

Avogadro also has much code to support it. Unfortunately, the change would require adding support within the Chemical JSON format. Formal charges should also be inferred when loading a PDB or XYZ file.

Coordination chemistry handling is a related issue; both macromolecules and smaller biomolecules often carry these in the form of amino acid-coordinated cations or pairs of cations, various heme cofactors, chlorophylls, vitamin B12, iron-sulfur clusters of different sizes, molybdopterin-bound, or even extremely complex clusters like the FeMo cofactor in nitrogenase.

Avogadro does not currently handle coordinate bonds any different from regular covalent bonds. Since these bonds don't have the same relationship to atom valence, calculations involving the latter are affected. This is, for example, the case of trying to automatically add hydrogens to heme (Figure 3).

There are a few ways out of this problem. The first is to simply detect these bonds within plugins, and handle them in a special fashion (Figure 4). Plugins that work with atom valencies would interpret bonds between certain hypervalent atoms and metals as coordinate bonds, and ignore them when appropriate.

Not all file formats contain bond order information. It's important to verify that Avogadro can infer correct bond orders even in the presence of complexes. This is a summary of bond order support:

Open Babel	SMILES	CJSON	CML	MMTF	PDB	XYZ
Yes	Yes	Yes	Yes	Yes	No	No

An alternative is to modify the structure before performing certain operations. This would involve either dissociating metal centers from their ligands (neutral ligands only or all of them), or formalizing the partial charges on all

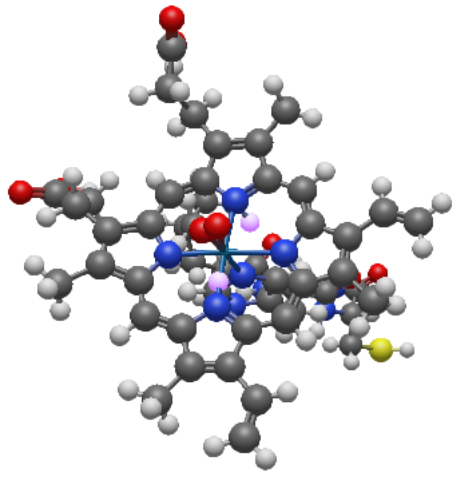


Fig. 3: An oxygen-carrying heme B[8] with hydrogens added automatically. The dioxygen ligand can be seen right above the iron(II) center, with a fragment of the carrier protein including the histidine ligand opposite to it and partially obscured by the heme. Two hydrogens on the protoporphyrin ring have been incorrectly added (highlighted in pink).

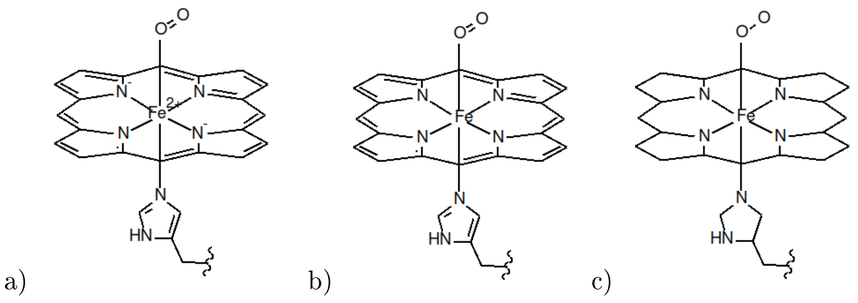


Fig. 4: a) Structure of heme with actual formal charges; bonds around the iron do not contribute to ligands' bond count. b) In this representation, charge has been transferred from the anionic ligands to the metal center, so those two bonds do contribute to valence. c) As obtained from a file with no bond order information.

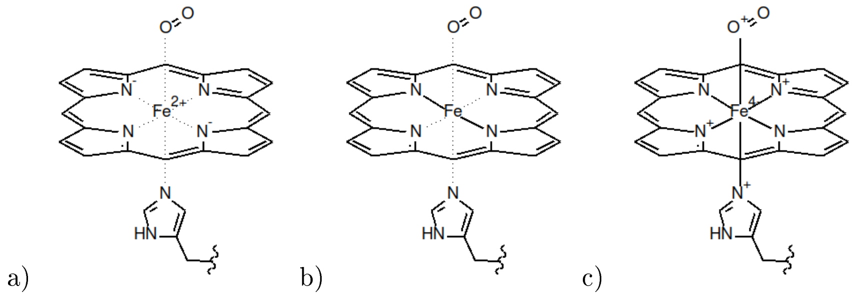


Fig. 5: a) Heme, with all coordinate bonds removed but formal charges preserved. b) Here, anionic ligands remain bonded, and their formal charge is transferred to the center. c) All electron pairs are distributed equally between the metal and its ligands, giving iron a -4 charge.

atoms in the complex (Figure 5). The first choice is quite logical, and could be converted into a user-triggerable operation (e.g. for feeding the structure into another program with worse support). The second is a relatively rough approach, but has the advantage of keeping all the atoms bonded while honoring valences.

Alternatively, coordinate bonds might be *stored*, rather than simply treated, differently. This approach would require some additional support that is beyond the scope of this project.

Open Babel	SMILES	CJSON	CML	MMTF	PDB	XYZ
No	No	No	Yes	No	No	No

As for PDB import, a few bugs were encountered during testing. These affect the Avogadro PDB importer, but not the alternative Open Babel path, while others seem to be common to both. For instance, trying to delete any single atom removes the entire molecule, regardless of the import method.

Another issue is the incorrect parsing of metal atoms, which occurs in Avogadro's PDB importer, but not in Open Babel (Figure 6). In this example, the relevant atoms are parsed as “Xx”, rather than “Fe” (read on for an update on this).

After eventually deciding to participate in GSOC 2022 (which, to my surprise, does not require a Computer Science background this year), I tried my hand at doing some small changes to the Avogadro code, just to get a glimpse of what I was about to set off for. I was bothered by the slowness of the PDB importer, so I tried to optimize it as much as I could during March 3 morning. This is the result (time to load a file, less is better):

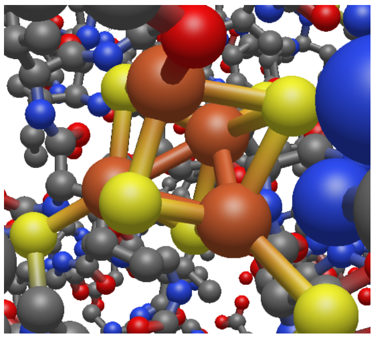
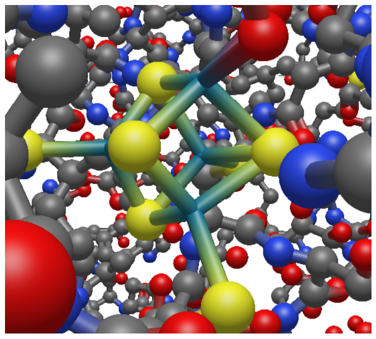
a)  b) 

Fig. 6: a) Aconitase[9] iron-sulfur cluster as imported by Open Babel. b) Same cluster, imported by Avogadro's native PDB reader.

Molecule	Atoms	Bonds	Before (s)	After (s)	Open Babel (s)
Aconitase	6138	5215	25	4	4
Nitrogenase	17025	13933	169	18	20
2x Hb-Hp	19196	17349	213	15[1]	31

I did this by rewriting `Core::Molecule::perceiveBondsSimple` to be $O(n)$, in contrast to the current $O(n^2)$ implementation. This makes the Avogadro PDB importer about an order of magnitude faster, and even faster than the Open Babel path!

The afternoon came, and I decided upon a new target. Remember the metal misparsing issue brough up a few paragraphs ago? I immediately fixed it. PDB atom symbols are always in capitals, while `Core::Elements::atomicNumberFromSymbol` expects the second letter in 2-letter symbols to be in lower case; a simple conversion was sufficient.

Regarding non-covalent interaction support, it would be beneficial to refactor the `Core::Molecule::perceiveBondsSimple` code above so it can be generalized to other cases. In particular, repurposing it as an iterator over neighborhoods, as lists of atoms. This has the potential to speed up other parts of the program, as well as make creating efficient plugins simpler.

Hydrogen bonds are the strongest intermolecular force in biological systems. They are crucial to understanding the normal structure of macromolecules, as well as how different drugs and metabolites interact with them. In an attempt to be as precise as possible in their prediction, VSEPR theory will be used, with extra vector operations to allow arbitrary deformed geometry, like that in the very common closed ring systems.

For a list of hydrogens near an electronegative acceptor atom, their bonds will be scanned to filter out those bonded to atoms that are not electronegative enough, or those that would form a highly deformed bond. If any atoms remain,

[1] While this set of hemoglobin-haptoglobin complexes has the most atoms and bonds, it also has a relatively larger surface area, so more non-neighboring atoms are excluded by the algorithm.

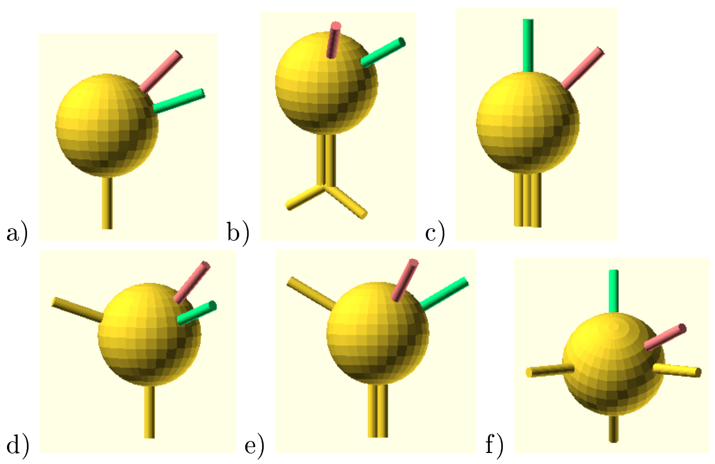


Fig. 7: a) A single bond on F, O, N marks tetrahedral hybridization with 3 lone pairs. This is a special case due to rotational freedom, which, after placing the best fitting hydrogen, would drop to the d case. b) A lone double bond requires prospecting the bonded atom to properly identify the nearest existing lone pair. c) The simplest case, sp hybridization. d) Two bonds, tetrahedral symmetry; as b, but deformed geometries must be taken into account. e) Deceivingly simple, deformed geometries make this case more complex. f) A vector sum of normalized bond vectors is enough to easily spot the lone pair. All of these images were rendered via OpenSCAD. A geometry with zero bonds is not shown, but possible ($\mathbf{F}^-$ anion).

the acceptor's bonds will be checked to decide what its hybridization is, and what set of allowed angles for lone pairs best matches each potential hydrogen bond. A difference angle will be calculated and candidates excluded accordingly.

2 Project Schedule

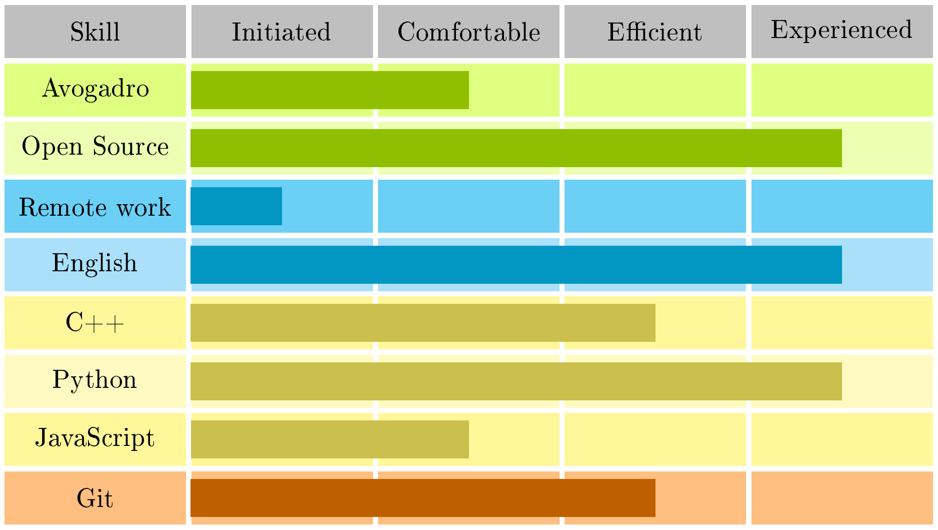
March 1	Start writing application, read and understand Avogadro code.
March 7	Send application to developers, start discussing goals.
April 4	Officially apply for GSoC 2022.
May 20	Start of the design phase. Investigate the code, document findings, outline plans. Low-hanging fruit is expected to be taken care of.
June 10	Ophtalmology final done. Now I can focus completely on coding.
June 13	Start implementing formal charge and coordination perception.
June 27	Formal charge and coordination milestone. Start working on visualization.
July 25	Basic visualization milestone. Start working on performance and small features.

August 8	Code improvements done. Remaining time for other work or unfinished details.
September 5	Project deadline.

The project schedule is centered around the visualization code. It's designed in a way that it can be completed with reasonable ease despite the occasional personal commitment. Once all milestones have been completed, as mentioned, I'll keep working (possibly at reduced productivity) to improve upon Avogadro further along the same line. I'll communicate with the mentor(s) to make sure I have the original goals in mind.

I expect to complete all tasks within the indicated schedule, and I highly doubt I might be unable to complete them within the entire summer, which also includes an extra month after the last milestone.

3 Experience



I've been using Avogadro since before Avogadro 2 was a thing. However, I've never done any computational chemistry beyond some properties prediction and the occasional tutorial to learn a molecular dynamics package. I mostly use Avogadro as an aid to understand how different reaction mechanisms work in 3D space, how substrates, ligands and drugs interact with proteins, and how macromolecules and macromolecular complexes are arranged.

Regarding other Open Source programs, I've used many of them since as long as I can recall (LibreOffice, GIMP, Firefox, GCC). For the last 4 years I've exclusively used Linux distributions as my daily driver OS, and now I use a large number of Open Source tools; Inkscape, Blender, Vim, LLVM, OpenSCAD, Git, CoqIDE, ChucK and OpenFOAM, to name but a few. I've had some contact with PyMOL, but I don't consider my experience with it worthy of highlight.

I learnt C at 13 and have been coding very frequently since then. I have learned Python, Java, C++, Lua, Dart, Bash, and recently Rust. I've written small programs in JavaScript (which I found extremely similar to Dart) and encountered no issues, but I'm far from fluent in the language. I've also occasionally used other programming languages like Scheme or Go.

I've contributed code to a few Open Source projects (both written in C++):

- LibreOffice
- Minetest

I have also had a few projects of my own:

hmars	A verified pmars-compatible simulator for playing the zero-player programming game Corewar, including a simple SDL GUI. Written in C many years ago, it used to include Python code, which I later rewrote in C. I did some heavy optimization, wrote many features and some automated tests for it, but the project's code quality was not acceptable overall.
"Comp"	This is what I called an embedded application I coded in C, for my own use, featuring chemical structure rendering (Figure 8) and optimized formula plotting in the 2D plane. I didn't release it *per se*, and don't intend to do so.
rsource	An interactive TUI Human genome visualizer, written in Python, 2 years ago. I kept the code cleaner and more modular, and used VCS. Now I consider it feature-complete.
Nodeverse	My ongoing project, a space-exploration game for the Minetest engine. Most of the code is written in Lua, but there are many components done in Python, Shell or Scheme, as well as the audio programming language ChucK and OpenSCAD's model scripting language. I've made a great effort to write high-quality, self-documenting code, and it's had a few contributors. Right now the package has over 2k downloads.
fluidics	A newer project of mine, a set of tools to aid with microfluidic device design. Written in Python and OpenSCAD.

Although my first tongue is Spanish, I speak English at near-native level. I attained 222 points in the Cambridge English Scale through official examinations (Cambridge English Proficiency, Grade A); for comparison, a CEFR C1 level is equivalent to over 180 points, while a C2 (the maximum level) requires 200.

I've attended 300 hours of Biochemistry lectures and activities as part of my medical degree. I've also done my share of personal research on chemistry, mostly organic chemistry, as well as biochemistry, genetics, etc. I have a very superficial understanding of physics, including quantum mechanics and the mathematical formalisms involved in it.

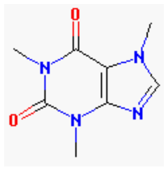
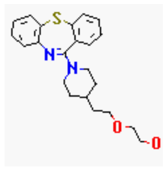
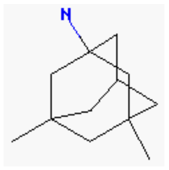
a)  b)  c) 

Fig. 8: a) Caffeine structure as rendered by “Comp” (on emulator). b) Quetiapine, a second-generation antipsychotic drug. c) Memantine, a symptomatic drug for Alzheimer’s disease.

References

[1] Doré AS, Robertson N, Errey JC, Ng I, Hollenstein K, Tehan B, Hurrell E, Bennett K, Congreve M, Magnani F, Tate CG, Weir M, Marshall FH. Structure of the adenosine A(2A) receptor in complex with ZM241385 and the xanthines XAC and caffeine. Structure. 2011 Sep 7;19(9):1283-93. doi: 10.1016/j.str.2011.06.014. PMID: 21885291; PMCID: PMC3732996.

[2] GitHub - Open Chemistry - Chemical JSON [Internet]. Hanwell MD; 2016 December 28 [updated 2019 February 15; cited 2022 March 1]. Available from: https://github.com/OpenChemistry/chemicaljson/blob/master/chemicaljson.md

[3] Chemical Markup Language - Molecular Convention [Internet]. Townsend J, Adams S; 2011 August 28 [cited 2022 March 1]. Available from: http://www.xml-cml.org/convention/molecular

[4] GitHub - RCSB - MMTF 1.0 [Internet]. Rose A; 2016 August 16 [cited 2022 March 1]. Available from: https://github.com/rcsb/mmtf/blob/v1.0/spec.md

[5] wwPDB Format version 3.3 [Internet]. Berman HM, Henrick K, Nakamura H; 2003 [updated 2012 November 21; cited 2022 March 1]. Available from: http://www.wwpdb.org/documentation/file-format-content/format33/v3.3.html

[6] Open Babel API Documentation 2.3 [Internet]. Open Babel Developers; 2012 February 28 [cited 2022 March 2]. Available from: https://openbabel.org/dev-api/

[7] James CA, Vandermeersch T, Dalke A, Apodaca R, May J. OpenSMILES specification 1.0. OpenSMILES [Internet]. 2007 November 13 [updated 2017 September 17; cited 2022 March 2]. Available from: http://opensmiles.org/opensmiles.pdf

[8] Andersen CB, Torvund-Jensen M, Nielsen MJ, de Oliveira CL, Hersleth HP, Andersen NH, Pedersen JS, Andersen GR, Moestrup SK. Structure of the haptoglobin-haemoglobin complex. Nature. 2012 Sep 20;489(7416):456-9. doi: 10.1038/nature11369. Epub 2012 Aug 26. PMID: 22922649.

[9] Lauble H, Kennedy MC, Beinert H, Stout CD. Crystal structures of aconitase with trans-aconitate and nitrocitrate bound. J Mol Biol. 1994 Apr 8;237(4):437-51. doi: 10.1006/jmbi.1994.1246. PMID: 8151704.

[10] Hermosilla P, Krone M, Guallar V, Vazquez PP, Vinacua A, Ropinski T. Interactive GPU-based Generation of Solvent Excluded Surfaces. Vis Comput. 2017;33:869–881. https://doi.org/10.1007/s00371-017-1397-2.

[11] Legon AC, Millen DJ. Angular Geometries and other Properties of Hydrogen-bonded Dimers: A Simple Electrostatic Interpretation of the Success of the Electron-pair Model. Chem Soc Rev. 1987;16:467-498. https://doi.org/10.1039/cs9871600467.