Computer Systems Project Proposal

AvaMol – Molecular Mechanics to Quantum Chemistry and Back

Masood Malekghassemi

Purpose: To implement generic computational molecular mechanics, implement more accurate ab initio computational chemistry methods, and then to create an AI that will run the ab initio methods to generate generic rules for the molecular mechanics part (thus making calculations more accurate). This is a valuable project because – and this is only as far as I know, which isn't too far – all previous chemistry programs have been written by chemists that know computer programming, not computer programmers that know chemistry. This will be the first full-featured implementation in C++ of molecular mechanics and ab initio methods that I've seen, if it comes to completion (that 'if,' by the way, actually means 'when,' because it **will** come to completion).

Scope of Study: The scope of this project is basically a tour of computational chemistry – avoiding only the absolute most annoying and rather complicated mathematical substitutions involved in semi-empirical calculations (something I've gleaned just from reading it – they may however just be deceptively complicated and quite easy once ab initio methods in general are grasped). Molecular Mechanics is just a ball-and-spring model of the molecule – simple to implement, simple to calculate energy, relatively simple to implement an iterative self-consistent field method for it. This particular section will be an exploration in generic code and code-reuse (due to the many ways that I could encounter a particular phenomena in the ab initio methods, I'd like to be able to implement new code without modifying old code too much). This involves using many different more conventional I/O functions as functors and defining numerous generic structures for 'Rules' and 'Interactions' and 'Identities;' the former defines the subsequent, and the latter defines the constituents referenced by the two preceding.

Background: I have been reading up on computational chemical methods, such as those for solving the multi-body Schrodinger equation and the simple ball and springs model. I have three books on the topics I'm interested in. One of them in fact has the title 'Computational Chemistry: Introduction to the Theory and Applications of Molecular and Quantum Mechanics.' The title in itself should give away that this is no novel topic – the idea of using numeric methods to solve chemical problems has been around for a long time, since the late 1930s (for computers specifically, the late 1950s or 60s). However, all of these programs appear to be written in cryptic Fortran code with non-descriptive variable names and eyesores based on the fact that Fortran 77, which all (save for one that I've found) of the programs are written in, is an incredibly ugly language due to its usage of character columns for semantic purposes. So, really, a re-write of these projects in a language that's high level and just as suited for numeric programming (optimizing C++ compilers makes the language a fair competitor with Fortran) could in itself be a whole new project, but I digress. There exists no 'state of the art' project comparable to the generality of what I'm doing. There are many computational chemistry programs out there – but

there certainly aren't any programs that have an AI to generate rules for molecular mechanics calculations from runs of ab initio methods.

Procedure and Methodology: My project is split into three parts. The first of them, the molecular mechanics section, I will definitely be done with before December (the I/O from files is being a pain in the neck, I haven't been able to decide very easily how I want my input files to be formatted). This first part is split into the Rules, the Identities, the I/O, and the System/ForceField calculations. I've finished the Rules and Identities, am almost done with the I/O, and due to my taking of the easy way out (using an iterative SCF method rather than a Lagrangian approach), the System/ForceField calculations will be a breeze (unless I hit a wall, like needing to refactor my code). The second part, the ab initio methods implementation, will probably end up using computational Hartree-Fock and/or computational Density Functional Theory for its operations (which won't need to be made generic, as it is purely a solver, not a rule maker). The final part will be the trickiest – an AI that will be able to read the data coming out of the ab initio approaches and transfer that to the generic rule based molecular mechanics part. This last part is the one part that I'm unsure of whether or not I'll be able to finish this year. If I have little schoolwork come Spring, it will be done – if I don't, well, you can say that my project was backlogged.

Testing: My program will be tested based on the functionality of the individual parts. The molecular mechanics part will be considered a limited success if I can take a methane molecule (a C with four H attached) with non-tetrahedral bond angles and get a tetrahedron out of it. It will be considered a general success if I can take a warped out of shape acetic acid molecule and get a dog (essentially what the acid is shaped like) out of it. Visualization will either be done imaginatively from looking at numbers or by cheating my graphics creativity out of a job by using GLUT to draw spheres and rods.

The ab initio methods will be tested much the same way. For both molecular mechanics and ab initio methods, I'm going to start with small molecules and work my way up (ab initio will start at H, MM will start at H_2).

Testing the AI will be a doozy. It's dooziness is why I'm unsure of being able to finish it – I won't really know if it's finished or not. I guess the best way to test it is to run the ab initio part, look at the output, interpret it myself, and then see if the AI's interpretation matches my own.

Expected Results: A program that will do what I've been saying over and over again throughout this paper.

Visualization: Okay, I guess I just have to write a small graphics part to be able to portray the volumetric density of electron clouds and molecular mechanics ball and spring models and potential energy surfaces... Or I could just use Ogre3D and only worry about making sphere meshes for *it* to draw (and *it* is a damn good 3D graphics library by the way. If not *it*, I'd use Irrlicht).

Value to Others: Uhhh, being able to have programs run accurate yet slow methods during idle time that will streamline and make more accurate the calculations of faster methods? I think it's a win win.