LORE: A TOOL FOR MANIPULATING KNOWN STRUCTURAL INFORMATION IN PROTEIN MODELING AND CRYSTALLOGRAPHY

Barry C. Finzel
Structural, Analytical & Medicinal Chemistry, Pharmacia & Upjohn, Kalamazoo, MI 49007
bcfinzel@am.pnu.com

1 Introduction

With the rapid increase in the rate at which protein structures are solved in recent times, we are challenged as crystallographers to try and make as much use of previous structural data as possible in the completion of our own work. To do so may be advantageous for several reasons. The tasks of density interpretation, refinement, and structure analysis could all be simplified with easy access to prior work. Making use of earlier data in a more analytical way should also help to provide more uniformity in structure determinations from laboratory to laboratory. Furthermore, if we have convenient tools for identifying and retrieving protein structural data, there is no telling what generalizations about structure may be revealed.

But *as* crystallographers, we have a number of unique needs that must be addressed in the design of software to help us utilize structural information derived by others. Simply knowing that one structure is related to another - that two proteins share the same protein fold, for example - is not enough. Secondary structure assignments, likewise, are of marginal utility. What we really need are tools for slapping a piece of structure into electron density, tools for overlaying similar structural fragments, tools for comparing segments of structure to one another. It is often the case that homology to a known structure is recognized only after portions of the molecular structure are fit into electron density. We could really use tools to help us make this initial fit, and then tools to help us recognize that the emerging structural topology is a familiar one. These are tools that manipulate *atomic coordinates*, often in very complicated ways. It is not enough just to identify another structure that *could* serve as a model for our new one. We must actually have the flexibility to *make* it a model, and modify it as appropriate.

For several years, we have been involved in the development of such tools in the body of a program called LORE. It includes a collection of features that help the crystallographer (or even an occasional modeler) identify and manipulate fragments of structure from prior art. Most descriptions of proteins in the literature generalize structural features. LORE simply provides a ready access to the atomic coordinates - a capability that we find indispensable. We use LORE for everything: for comparison of similar structures, overlay of enzyme/inhibitor complexes, the analysis of the likelihood of loop conformations, as an aid to interpreting difficult electron density, for side chain rotamer modeling.

In this presentation, we will deal primarily with LORE design and programming issues. Examples of how LORE is used are more easily demonstrated in an on-line tutorial, and I invite everyone here to see one and try it out.

2 THE PROGRAM DESIGN

2.1 Needs

Before describing what LORE *is*, let’s consider what we would like LORE to be. What should a program that makes use of data from other structures be?

1) It must be capable of finding substructures from known structures that have the attributes the user desires.

A simple statement, but it brings to mind many questions. *What are substructures?* A polypeptide chain?; 5 residues; 15 residues; 100 residues? Must it be only one polypeptide chain? Could a substructure be a beta sheet, or a bundle of helices, or components of an active site? *What are the known structures?* Anything in the Protein Data Bank? Only unique, well refined structures (does anybody really know what those are...)? Just T4 lysozyme structures? Everything except T4 lysozyme? Only the structures that I have done myself? Do known structures include enzyme co-factors? inhibitors? solvent?*What kind of attributes define the desired structures?* Are these geometrical? Conformational? Primary, secondary, or tertiary structure? What about other protein attributes like polarity, dipole moments, electrostatic potentials... The list of parameters describing protein structure is almost endless. Perhaps, a more relevant questions is *How will the desired attributes be communicated by the User? How do you find these substructures?* Perhaps the constraints of the search algorithm will define the mechanism of query.

2) Once identified, substructures must be manipulated in ways that benefit the user.

This is more vague than the first demand, but perhaps the computational implications can be best illustrated with a concrete example. Suppose we want to find all bent helices 20 residues or longer with a proline (inducing the bend) in the middle. Suppose we know how to search for such things, and we do so, and we find that there are 50 of them in the PDB. *What do we do with them?* It would be nice to look at them. But coordinates from the PDB are scattered all over 3D space. They need to be overlaid somehow, but on what frame of reference? *Just what do we mean by overlay?* Superposition? Superposition requires *two* sets a coordinates. What serves as the target? *What atoms do we overlay?* All of them? Surely not all fifty of our hits have the same amino acid sequence. They can't be expected to have the same atoms. Should we overlay just CAs, or maybe just the atoms in the proline? If we are interpreting electron density, it would be nice to have the substructures overlaid on the density. Is that even possible? When we view the substructures, do we want to see them all overlaid, or side by side? One at a time? All at once? Fifty substructures is a lot to look at. *Just how do we plan to use them?* It would be nice to pick one and include it in our model, but there are the amino acid sequence differences to contend with. *Can the software help us to decide which we should use?* What criteria could be used to evaluate them? Is that criteria always the same, or does it vary from search to search?

Our consideration of the requirements of a useful program has led to more questions than answers and raised many computational issues. But this is necessary. It is clear that one program cannot do or be everything. We must choose those things that are important to us as crystallographers, and try to find workable solutions that result in a program that is really useful.

2.2 Design Considerations

Often the design of a building is dictated more by the limits imposed by the available materials, than by the ideal first imagined by the designer. Such is the case with LORE. It was written and rewritten and rewritten again to help generalize the utility of two algorithms that form its' heart and soul: the conformational search algorithm, and the superposition routine. The superposition routine is not all that special, though the interface to it is somewhat unique, and we rely on it heavily. The search algorithm, however, is another story. You can't really understand the LORE user interface without understanding the strengths and weaknesses of the search algorithm, for the program was designed around this algorithm.

The Search Algorithm: In 1986, Alwyn Jones (Jones & Thirup, 1986) published an article describing a search algorithm based on alpha-carbon similarity. He showed that if you compute a matrix of all inter-alpha carbon distances in a protein substructure, you can use this like a molecular fingerprint to find similar substructures in a library of known proteins by looking for identical patterns of interatomic distances in library structures. (The alpha-carbon fingerprint was not really a new idea, but using these as the basis of a search algorithm was). Once these interatomic distance matrices are computed, this algorithm is remarkably fast, and very reliable. (One could argue that a left and right handed helix, for example, has the same pattern of inter-alpha carbon distances, but in practice this ambiguity is rarely a problem. A superposition of coordinates quickly provides a means of discriminating between misfits, in any event).

The other nice thing about this algorithm is that, with the proper user interface, it can be very flexible. If we don't know an alpha carbon position, we can just ignore all the matrix elements involving that CA, and chances are that the substructures we find are going to look like what we wanted anyway. You can also easily imagine a search for multiple disconnected segments of chain, such as the strands of a beta sheet, by just looking for the fingerprints of the individual strands, and then requiring that the off-diagonal matrix elements that define the relationship between CA's in different segments match as well.

There are really only two things that you need to use such an algorithm: a database of known structures, including pre-computed inter-alpha-carbon distances, and a target alpha carbon geometry.

The Database: The database of known protein structures that we construct for LORE is layered to enhance efficiency in searching. The lowest layer contains atomic coordinates of Protein Data Bank (PDB) entries. These are reformatted into residue-indexed direct access files to simplify extraction of selected residue ranges. Layered over this is a database of "chain information" containing amino acid sequence and Ca geometry data for individual polypeptide chains in the PDB, and pointers to the complete atomic coordinate data in the lower layer. The chain information is the primary data used in searching. Atomic coordinate data in the lower layer is referenced only after homology has been confirmed through examination of sequence and geometry data contained in the chain information. The highest layer is a text-based index of chains. This index points to chain information and includes chain rankings based on properties such as uniqueness, resolution of the structure determination, R-value, etc. The rank is helpful in selecting the specific subset of chains most appropriate to a particular application. For example, the user can restrict a search to only proteins from a given structural family simply by modifying the rankings in the index file and then selecting only the chains that meet a minimum rank requirement at run time. In our laboratory, structures are most often ranked by quality (resolution, R-value) and only the best chains are selected unless a larger database subset is necessary.

The Target: Since the search algorithm requires an alpha-carbon geometry, we must declare a ’target’ as a residue range (or ranges) from the developing molecular model to key the search. The target specification establishes the length of fragments to be considered and defines (through predetermined alpha-carbon positions of these residues) an approximate geometry of acceptable fragments. It is recognized that many structural units of protein structure may not be represented by a single residue range. The adjacent strands of a twisted b-sheet, for example, have a well defined structure independent of the length of the intervening loops or the relative position of the segments in the overall amino acid sequence. The target specification must be flexible enough to allow for this degree of complexity. Also, since an important application of fragment fitting is the modeling of incomplete structures, we must not require that the target be entirely defined beforehand. Any number of missing atoms or residues should be tolerated.

Superposition: When creating a fragment, LORE always anticipates that it will be positioned to overlay the current target, and when possible a residue-to-residue correspondence is established between the fragment residues and target residues. This correspondence simplifies subsequent fragment manipulations such as superposition and coordinate substitution. The target thereby provides a real set of coordinates that are necessary for common superposition of identified fragments. Very often, an RMS fit to the target coordinates is the most reliable measure of the quality of fragments from a search. But the superposition engine must be flexible. It must tolerate missing residues or atoms, and deal with amino acid type mismatches. It may not always be in the users best interest to superimpose just CAs, so the user should be able to say what atoms will be superimposed, with reasonable defaults. It would also be convenient to be able to repeat a superposition after a few fragments have been examined and the user has a better feeling for which atoms of the target are most trustworthy. This is easily done.

The Residue Mask: To provide additional flexibility to all search and superposition operations in LORE, we have created a user interface feature called the target "residue mask". With the mask, the user can simply exercise residue by residue control over different LORE operations. It is through the mask that users specify particular sequence requirements in substructure searching (e.g., a glycine at the fourth residue position), or allow for conformational uncertainty when Ca positions are not known or trusted. Because it influences so many different LORE functions, the status of the mask is always on display in the LORE terminal window. Algorithmically, the mask consists of two logical arrays. The first array defines a logical state for each of the twenty amino acids, and specifies allowed sequences. This construct is used primarily in searching. The second array defines an *on/off* status for each residue in the target and, consequently, corresponding residues in a fragment. This *on/off* status has some impact on most LORE operations involving residues. It can be set to disable superposition at certain residues, for example, or flag CA positions that should be ignored during a search.

The Fragments: The result of a search is really just a pointer to a molecular fragment in a library structure. Fragments are loaded into memory as a completely different step. This is a design feature to give the user more flexibility in dealing with the results of a search. When a fragment is loaded and superimposed onto the target, the user has the option of loading more than just the identified substructure. A fragment may contain prosthetic groups or solvent molecules, or any number of atoms in the neighborhood of atoms tagged to the target. This makes LORE a very powerful general superposition tool.

Rotamers: The idea of "rotamers", common conformations of amino acid side chains that represent geometries of particular stability, is now widely accepted. To speed modeling, many graphics programs include push button or menu driven interfaces that generate a set of possible amino acid side chain conformations from which the correct conformation may be chosen. The addition of a rotamer look-up capability to other LORE algorithms is very beneficial, because the user may then assess the frequency of occurrence of different side chain conformations in given main chain conformational contexts. This functionality permits the interactive extension of generalizations about the relationship of side chain conformations and secondary structure to very specific substructural motifs. Frequency analyses of this type can be quite predictive.

Options for Atom Manipulation: It is the presumption in LORE that any residue in any fragment may replace a residue in the developing molecular model at any time. It is important, therefore, to be able to manipulate coordinates as precisely as possible. It is generally the case that precision follows complexity, but there are a couple of simple operational options that can provide a lot of flexibility without introducing excessively complicated concepts. A simple list of atom names (the Superposition Atom List), for example, defines atoms from corresponding residues that will be overlaid during superposition. This can be set to Ca, all main chain atoms (N Ca C O), or any arbitrary list. Since the r.m.s fit of a fragment to the target is often taken as an indicator of a good fit, it is very useful to be able to limit the superposition to only trusted atom positions. This list, when coupled with the residue alignment control available through the residue mask, offers good flexibility without significantly complicating the user interface. A similar list of atom names specifies which atoms are replaced when target residues are replaced by their counterparts from a given fragment. In some applications, such as structure refinement, it is as important to leave atom parameters undisturbed during rebuilding as it is to make necessary corrections. If only certain side chain atom positions are wanted from a given fragment, this can be easily arranged. Finally, since LORE does most everything in memory, it is desirable to be able to optimize memory utilization. One can specify a list of names of atoms that are to be loaded into memory from the disk when fragments are created. Side chain or solvent atoms can be easily ignored when these are not needed.

Implementation: LORE is implemented as a non-graphical subprogram of the model building program CHAIN. Since LORE does no molecular graphics, it really relies upon this host program to display fragments; a necessary step in the process of selection of desired fragments. Also, because CHAIN is a map display and map-fitting program, it provides a convenient platform for LORE. CA positions are easily built into skeletonized electron density with CHAIN, and these can then be expanded to a complete molecular model with tools in LORE.

3 EXAMPLE

If time permits, examples showing how LORE can be used build a complete molecular model from an alpha carbon backbone, or to locate and overlay a homologous protein structure will be shown. These examples have already been well documented (Finzel, 1995).

References

[1] Jones, T.A. & Thirup, S. Using known substructures in protein model building and crystallography. *EMBO J.*, 1986, **5**, 819-822.

[2] B.C. Finzel, B.C. (1995) "Mastering the LORE of Protein Structure". *Acta Crystallogr. Sect D,* **51**, 450-457.