Automated site-directed drug design using molecular lattices

Richard A. Lewis, Diana C. Roe, Conrad Huang, Thomas E. Ferrin, Robert Langridge, and Irwin D. Kuntz

Department of Pharmaceutical Chemistry and the Computer Graphics Laboratory, University of California, San Francisco, USA

Receptor-based drug design is predicated on the knowledge of the structure of a target receptor and the principles of molecule recognition. The objective is to produce a wide diversity of structures that are sterically and electrostatically complementary to a specified receptor site. Many drug–receptor interactions are controlled by a few key receptor groups. This observation leads to a design approach in which one focuses on chemical fragments that putatively interact with the key receptor groups. There then remains the difficult task of joining the fragments into molecular structures that match the spatial patterns of recognition forces in the receptor site. In this paper, we describe a new modeling program, BUILDER, that combines database searching techniques and structure generation algorithms within an interactive graphics modeling environment (MidasPlus). A novel tool for process communication (delegate) *is introduced and examples of its use are given. To demonstrate the functionality of the package and its ability to produce novel structures, we examine the active site of HIV-1 protease.*

Keywords: drug design, molecular graphics, MidasPlus, database searching, structure generation, HIV-1 protease

INTRODUCTION

Receptor-based drug design requires knowledge of the structure of a target receptor and must incorporate principles of molecule recognition. The structural information is available through experiments such as X-ray crystallography, or, indirectly, through a pharmacophore model. Many drug–receptor interactions are controlled by a few key receptor groups. The receptor binding site can be surveyed to locate these *hot spots* of interaction.[1,2,3] This is an important first step in the design of novel compounds, but there remains the very difficult task of joining these interaction regimes together to generate conformationally sensible, synthetically accessible target molecules.[4] In this paper, we focus on the joining operation with the goal of producing chemical structures that match the spatial patterns of recognition forces in the receptor site.

There are three basic methods for developing molecular structures using site-directed drug design: database searching, automated structure generation and manual model building. All three methods have their strengths and weaknesses. In this paper, we borrow from each of these methods in an attempt to combine the strengths while avoiding the weaknesses. An example of the database searching paradigm is the DOCK package.[5,6] This program searches a coordinate database for structures that show a high degree of shape complementarity to the site. The strength of the method is that it can quickly scan many different structures to give a list of compounds that, in specific conformations, fit the site well. The usefulness of this approach for finding leads has recently been demonstrated for the active site of the HIV-1 protease.[7] One weakness of DOCK is the simplicity of the scoring function employed[6]—a structure might be sterically reasonable but electrostatically nonsensical. It is also possible that small complementary structures might not score highly enough on steric grounds alone to appear on the final list of compounds. DOCK also only considers a single conformation of the active site and individual conformers of the putative ligands. While there is some allowance for geometric variation, the conformational search problem is not addressed directly.

Structure-generation programs[8,9] connect receptor site points with idealized but automatically generated molecular graphs in an exhaustive and unbiased way. The principles of molecular recognition are built into the generation algorithms so that only molecular graphs with reasonable conformations and electrostatic interactions are produced. However, structure generation algorithms are very prone to combinatorial explosion, so that many simplifying heuristics have to be employed. This limits the number of bonding geometries that can be examined, and ultimately the diversity of molecular graphs that can be produced. Furthermore, the structure generated by connecting the site points may have good chemical complementarity but may not be synthetically accessible. Finally, a molecular graph with acceptable conformational properties may still not fit the active site with high accuracy.

Receptor-based drug design can also be performed manually using molecular modeling programs. Each structure

Color Plates for this article are on page 106.

Address reprint requests to Dr. Kuntz at the Dept. of Pharmaceutical Chemistry, University of California, San Francisco, CA 94143–0446, USA.

Received 20 February 1991; accepted 9 July 1991

can have ideal bonding geometries and as much complementarity as the designer can incorporate. The main weakness is that this approach is very dependent on the creativity of the user and the way in which the site is perceived. Despite the advances that have been made in the field of molecular graphics,[4,10] it is still a very difficult task for a human operator to build complex ligands for a three-dimensional (3D) site, particularly when the site is large and several bond lengths are to be spanned. The main strength of a model-building approach is that it allows a synthetic chemist to become involved at an early stage during the design process, so that the proposed ligands are at least synthetically reasonable.

The approach described in this paper uses supramolecular lattices and graph-searching techniques and makes use of a novel tool for interactive process communication, the *delegate* program. We combine the capacity of database searches to find real structures that fit into the site with the exhaustive and unbiased algorithms from structure generation and the ability of molecular graphics to allow the interactive incorporation of synthetic knowledge into the design process. The goal is to produce a diversity of structures that are conformationally, sterically and electrostatically reasonable.

METHODS

Overview of design strategy

An overview of the general design strategy is shown in Figure 1. The starting point is a 3D atomic model of the receptor site. Next a DOCK search of a 3D database is performed to find a collection of molecular structures that match the site sterically. Finally, one or more large irregular lattices are created by joining the molecular structures according to chemically acceptable virtual bonds. These steps comprise the initialization phase, which is accomplished in batch mode, off-line calculations. The design phase begins with a user working at an interactive graphics station running the enhanced MidasPlus graphics program.[11] The user selects a region of interest by placing a small sphere anywhere in the receptor site. Portions of the molecular lattice are displayed on command. These portions can be as restricted as single atoms or can be the full set of molecules that place an atom within the probe sphere. Other commands allow the user to inspect atoms or molecular fragments that are joined by virtual bonds to the molecular fragment containing the initial atom. Unwanted atoms or fragments can be removed from the display. Any part of this process can be repeated at secondary locations in the receptor site, providing a set of fragmentary structures which, in general, are not connected to each other. The user can attempt to join the fragments by continued interactive searching through the database. We also provide several automatic search routines to join parts of *target* atoms. The whole process, or any stage of the process, can be repeated as desired to explore or edit the molecular lattice in chemically reasonable steps. In general, the structures thus generated are not ends in themselves; rather they should be seen as a series of sensible suggestions, serving as steric, chemical, and synthetic templates. By providing methods for the task of how to connect ligand fragments into viable structures, we hope to free the user to think about other synthetic and design issues.

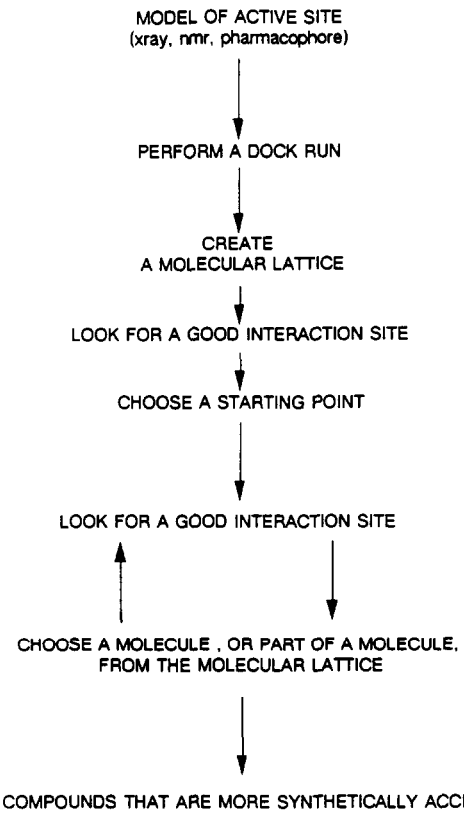

Figure 1. Overview of the general strategy used for lattice-based drug design

Chemical graphs and molecular lattices

The fundamental data structure used in this work is the chemical graph, whose nodes are the atoms of a chemical structure and whose edges are the chemical bonds. Each atom, or node, has coordinate and element information associated with it. The chemical graphs in this paper are formed from a large number of molecular structures (see below) and could contain several thousands of atoms and bonds. We call these large graphs molecular lattices. A subset of the lattice might form a molecule, a ring or just a pair of bonded atoms. Previous efforts at building chemical graphs within active sites have considered regular lattices, such as the diamond lattice.[12] Here, we describe a method for creating irregular, site-specific lattices that have some promising characteristics.

Creation of molecular lattices. Our concept of an irregular lattice is very general. The lattice can be created from any collection of molecules or molecular fragments placed by any procedure into the active site of interest. The only requirement we imposed is that none of the molecules or fragments in the collection has bad van der Waals contacts with site atoms. However, it is not necessary, or even desirable, for all molecules to fit the site closely. Loosely fitting structures, so long as they avoid the walls, can help to populate the site. We have used the shape-fitting DOCK

program[5,6] to search a subset (Seibel, unpublished) of the Cambridge Structural Database.[13] The top candidates from this search (see below) are used to create our prototype databases. The DOCK procedure automatically avoids steric clashes and guarantees good steric fit. The collection of molecules, with their original bonds are the *0-level* lattice. At this level, the graph will be disconnected.

We note in passing that any source of molecular data would be potentially useful. For example, the geometries of amino acid side-chain interactions[14] could serve as a collection of side-chain fragments. Furthermore, other database-searching methods[15,16] could be used to identify and position molecular structures in the site.

The next step in the procedure is to fuse the structures in the database into a large composite molecular lattice (Figure 2). The rules for fusing the database[17] include geometric and chemical considerations. A bond or edge between two atoms i and j in the lattice is created if i and j were linked originally by an intramolecular bond, or if the distance D_{ij} falls with the statistically determined limits for a bond formed between the element types of i and j.[18] Limits could also be placed on the angles and dihedrals allowed in the lattice and on the chemical bonding combinations. Such limits would reduce the interconnectedness of the lattice, but at the cost of greater processing time of the initial data set. At this point, we do not examine bond and dihedral angle restrictions at the lattice level. We take bond and dihedral considerations into account during the graphical search procedures described below. The result of these operations is to fuse the database into a level-1 lattice with the original molecular structures connected by acceptable intermolecular edges.

We can also create a lattice at a higher level of abstraction in which the original hybridization of the atoms in the database is deliberately ignored. The lattice is then simply a collection of atoms joined by real or virtual chemical bonds. Since the original molecules at this level are not given special weight, some of the real and virtual bonds will be mutually exclusive. We call this type of representation a level-2 lattice. Each of the lattice representations has a useful role to play in the design phase of the project, as explained below.

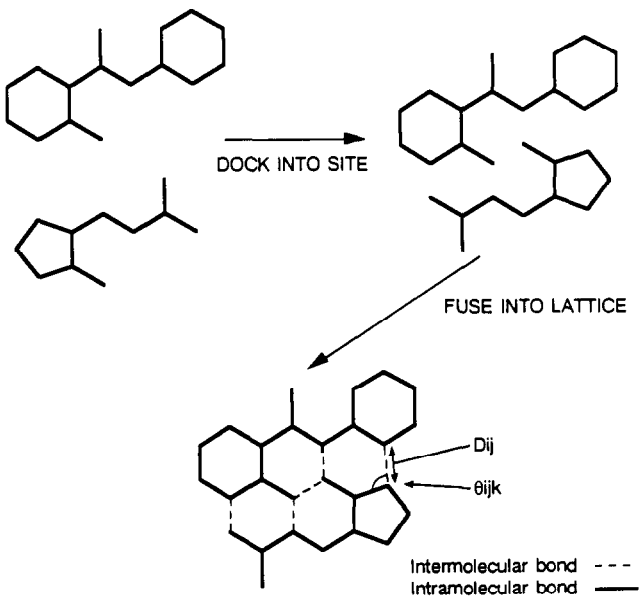

Figure 2. Molecular lattice formed from molecules positioned from a DOCK run. Molecules are fused into a lattice by adding allowed intermolecular bonds. Each atom may have hundreds of new neighbors

Lattice size. An irregular molecular lattice can contain an immense amount of information. The complexity of the final lattice depends critically on the number of molecular structures to be fused. For this initial work 200 molecular structures were selected to create the level-0 lattice. For this collection there are on average 4 intermolecular bridges between any pair of original molecules based on bond-length considerations alone, giving a totally interconnected level-1 lattice. Using just bond-length constraints, the prototype lattices have a mean of 100 edges (i.e., possible bonds) for each atom for the level-2 lattice. The question of what constitutes minimum acceptable statistics for these lattices will be explored more rigorously elsewhere (manuscript in preparation). For present purposes we are only concerned that the interconnections are appropriate for the automatic search procedures described later.

Data storage. It is not desirable to impose *a priori* limits on how large the lattice may be, or how many nodes it may contain. For this reason, we have chosen to store key pieces of information about the lattice in dynamically sized arrays. A level-0 lattice, containing atom-type, coordinate and parent-molecule data for several thousand molecules can easily be kept in physical memory using this data structure. The edge lists needed to represent level-1 and level-2 lattices are significantly larger and, under some circumstances can be too large to keep in physical memory. Optional storage of these arrays on disk for query by random access is provided. Theoretically, the number and density of structures one could use as input to create an irregular lattice is bounded only by the amount of main memory and disk space available. Practically, we have found collections of 150–300 structures suitable for prototype purposes. About 1 Mb of main memory and 3 Mb of disk storage are required to hold lattices of this size without compression efforts.

Interactive aspects

Probing the site. To create a putative ligand, the initial action is to locate the important interactions within the site. The site could be explored by a variety of methods such as AMBER potentials[19,20], GRID,[21] HSITE[3] or by visual interpretation of electrostatic or accessible surface models. No single method has proven clearly superior, so our program provides for compatibility with each of these approaches.

Choosing a zone of interest. An interesting region of space, called a *zone*, can be selected and visualized on the graphics display by establishing a sphere of user-specified radius and position. This sphere can be moved within the site interactively with standard MidasPlus functionality. Once this region has been chosen, a ZONE command can be issued to the delegate program. This command causes the program to find the atoms in the lattice that fall within the zone sphere and to respond by displaying the parent molecules for each atom within the zone. The lattice is inter-

preted here at level-0, that is, as disconnected constituent molecules. The molecule containing the atom closest to the center of the zone sphere is displayed first. The other molecules can be paged through by the user, to prompt ideas as to what chemical groups might be best to occupy that zone. Since there may be many molecules from the lattice with atoms inside the sphere, the user also can inspect them in groups, rank ordered as desired.

It is anticipated that the ZONE command will be used frequently to create multiple zones within the site and to redefine old zones if nothing interesting is found on the first search. A brute force search for atoms within certain regions of space would require an $O(n)$ algorithm where n is the number of atoms in the database. For large lattices, this search could lead to unacceptably long delays. We have used a cubing algorithm to perform a once-only partition of the lattice atoms into boxes during the initialization phase of the program. The dimensions of the boxes are determined by the number of atoms in the lattice so that there is an average of 5 atoms per box. To find the atoms within a zone, we find boxes which impinge upon the zone, and check for inclusion only those atoms lying within these boxes. In the current implementation, the minimum dimension a box may have is 1.5 Å; the maximum radius of a spherical zone region is 1.5 Å. Hence only the box that contains the center of the zone and the other 26 adjacent boxes need be considered. In the worst case, the number of distance checks that must be made is $O(27\ M)$ where M is the maximum number of atoms per box. In trial experiments performed on a lattice with the average number of atoms per box set to be 5, the box dimensions varied between 1.5 and 1.9 Å and the number of atoms per box varied between 0 for a box at the boundary of the site and 149 for a box in a central, well-populated part of the site. Using this strategy, the ZONE command responds very quickly.

As one builds up a composite structure, it is often visually confusing to have several molecules displayed simultaneously in the same region of space. There are commonly steric conflicts between molecules in the current zone and those that have been chosen to fill other zones. These clashes need to be resolved in some hierarchical fashion. In our current implementation, we have chosen a simple strategy for resolving such conflicts. We color all atoms that clash unacceptably and let the user edit out the offending parts, yielding a collection of (disconnected) molecular fragments. Other strategies are under consideration. It may prove feasible to have the molecule at highest priority throw a shadow over the other fragments. This shadow could be formed by a van der Waals-type surface. Shadowing would eliminate the visual paradox while still allowing the user to see what other molecules lie within a bonding distance of the atoms in the current molecule. It should also be possible for users to create their own display hierarchies that emphasize chemical information. The combination of MidasPlus and the delegate procedures (below) allows such display features to be coded rapidly.

Connecting fragments

General methods—depth-first and breadth-first searches. We next turn to the problem of joining two or more molecular fragments. The goal is to connect these fragments into a single fragment or a complete molecule. As noted earlier, the fused molecular lattice can be interpreted as a connected chemical graph (level 2). Chemical graphs are readily traversed by either depth-first or breadth-first algorithms[22] (Figure 3). Both strategies are variations of the priority search algorithm (Figure 4). Depth-first searching is achieved with a stack data structure. For breadth-first searching, this task is done with a queue. The search tree is maintained as a linked list, and each item on the list contains information on the atom it represents, its parent node, and the next node to be visited.

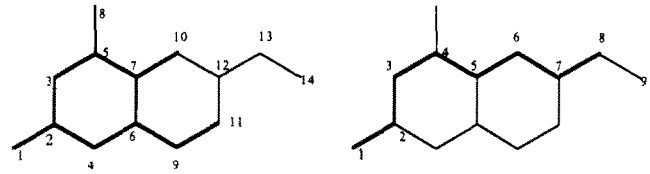

Figure 3. Two different graph-searching strategies visit the nodes on a graph in a different order as indicated by the number at each node

These two approaches have well-known advantages and disadvantages. A depth-first search will normally find a path using fewer computer resources, but it may not be efficient as there is no guarantee that it will move towards the node of interest. A breadth-first search strategy will ensure that the shortest path between two nodes is found. The disadvantage of breadth-first searching is that the search tree grows exponentially. If each node has M neighbors (ignoring the ones that are parents), then there will be $O(M^n)$ nodes to be visited at the nth generation. The amount of memory required to hold the growing tree could be as much as $\Sigma\ kM^i$, where k is the size of memory required by each node. In our trial databases, with $M \approx 100$ and $k \approx 12$ bytes, one would need over 12 Mb by the third generation.

Prioritizing and pruning searches. Both search strategies require modification for finding paths within a chemical graph as large as the irregular lattices used here. First, the order in which nodes are visited has been changed so that

```
initialize
start main list
do
        get next item from main list
        get item neighbours (possibly prioritized)
        while there are still neighbours to be visited
                get next neighbour
                if the neighbour has not been visited
                        and the pathway passes the pruning tests
                                add the neighbour to the main list
                                check to see if terminating criteria have been met
until the main list is empty or the search is terminated
```

Figure 4. Pseudocode program for searching a graph using a generalized priority queue. The code has been modified to allow for pruning and prioritizing as indicated. A depth-first search is achieved by using a stack as a data structure and stack-based operations to provide the next item on the list. A breadth-first search requires the use of a queue and queue operations to access the data

the most promising paths are visited first. This is done using a prioritizing function that sorts the neighbor list of a particular node to find the best neighbors. The prioritizing function directs the depth-first search toward the target.[12] Both distance- and angle-based prioritizing functions have been implemented and tested. For the distance function, the best score is the shortest Euclidean distance to the target. The node that is closest to the target is looked at in preference to the other nodes. The angle function has a score related to the angle between a target vector and the parent–child vector. A node that directs the search more parallel to the line segment between the origin atom and the target atom is favored. The depth-first search also has a limitation on total path length: The search is terminated whenever the current path length is greater than twice the estimated minimum path length between the origin and the target atoms.

So far, the lattice has been discussed as if it were conservative, that is, as if all edges and all nodes are allowed to occur on any pathway. A search should be *pruned* by requiring that all the growing paths have a geometric or chemical sense. From the definitions used to create the irregular lattice, it is probable that all edges (that is, bonds) are chemically reasonable and hence allowed. For a level-2 lattice, there are no rules that restrict bond angles, so whenever a new atom is added to a growing path, its geometric relationship with its 1, 3 and 1, 4 neighbors must be tested. A list of the angle constraints and tolerances used is given in Table 1. Of course, a pathway must be self-avoiding, so a new atom is checked for steric clashes with the other nodes on the path.

The search can also be pruned according to chemical element. Certain acyclic chemical fragments can be removed because they are intrinsically unstable. For example, a path of three oxygen atoms in a row might be rejected, as might a path containing a monovalent element. The DENDRAL badlist[23] has been used as a basis for rejection (Table 2).

There are other ways in which a search can be terminated. One might want to link two specified atoms from two fragments, or to trace a path to an atom of a certain type with a zone, for instance a heteroatom next to a hydrogen-bonding site point.

The directed, constrained, pruned searches are more efficient and provide more chemically reasonable paths than the standard implementations of the search algorithms. The process of connecting up zones or fragments is fast enough to be considered for interactive use. Under trial conditions, depth-first searches had a response time of about 2 minutes real time on a Sun Microsystems SparcStation I, whereas breadth-first searches tended to have response times of about 5–10 minutes real time.

Table 1. Bond angle constraints

Bond type		Ideal angle (degrees)	+Tolerance	−Tolerance
Bond	1	180.0	5.0	5.0
Torsion	1	0.0	180.0	180.0
Bond	2	120.0	5.0	5.0
Torsion	2	0.0	5.0	5.0
Torsion	2	180.0	5.0	5.0
Bond	3	109.5	5.0	5.0
Torsion	3	0.0	180.0	180.0

Table 2. Disallowed atom sequence combinations for path generation

O	O	O
S	S	S
N	N	N
?	Cl	?
?	Br	?
?	F	?
?	I	?
N	S	S
S	S	N

Note: ? = Any atom type

User interface

Any molecular modeling program that requires direct human intervention should be firmly based on an interactive molecular graphics system. Typically, the user will need to run complex programs such as the *zone* and *search* routines described above and to have the output displayed visually. The design of a lead compound is a particularly complicated task, requiring that the user be able to interact with the design program and that the design program be able to modify the graphical display. For the work described here, we relied on the MidasPlus graphics system running on a Silicon Graphics IRIS 4D/80GT workstation. MidasPlus provided both the necessary functionality to display and manipulate molecular representations and a command-line oriented user interface. The latter feature permits both interaction with a human user and image manipulation by external computer programs. For example, a user running MidasPlus can invoke another program that causes coordinate data to be passed directly from the currently displayed image to that program and the output from the program to be interpreted as MidasPlus commands that manipulate the displayed image. Although this mechanism is general, it is not necessarily efficient when information is required frequently during a graphics session.

A novel solution to this difficulty has been the creation of the MidasPlus delegate feature. A delegate is a program initiated by MidasPlus that can execute in parallel with MidasPlus. A two-way interprocess communication channel is maintained that uses the standard MidasPlus command language. These delegate programs can pause and report directly back to the user, even during the middle of a computation, to allow the user to monitor the progress of a calculation and to decide what further action, if any, is necessary. This is a valuable addition to our interactive environment.

The delegate mechanism provides an effective way to add functionality to MidasPlus and provides a valuable addition to our interactive environment. Users who are willing to program in any language supported by the system can easily develop their own delegates and need not wait for MidasPlus developers to implement desired features. In addition, del-

egates can use information sources other than the structural and chemical knowledge available in the MidasPlus data structures. Thus users can incorporate chemical information such as electrostatic forces, hydrophobicity, conformational restrictions, and structure prediction into MidasPlus via the delegate feature. Delegate commands we have developed specifically for BUILDER are:

- BALLMOVE adjusts location of the probe sphere (see ZONE).
- CLASH checks for and highlights steric clashes between displayed molecules.
- COMBINE combines fragments of two or more molecules into a single molecule and saves it to a file.
- LINK identifies a chemically sensible path between two atoms in the lattice.
- REMOVE deletes from the display a specified set of atoms.
- SAVE writes to a file the coordinates of the specified molecule.
- SHOW displays molecules identified by the LINK or ZONE commands.
- STOP stops the delegate communication package with proper terminations.
- ZONE finds all molecular fragments in the database containing an atom within a user-defined distance of the center of the probe sphere.

More details about delegate from both user and implementator perspectives are given in Appendix A. An example of a delegate program is given in Appendix B.

The entire BUILDER program has been implemented in C and has been used on Silicon Graphics workstations. For availability of this code and MidasPlus, contact the authors.

RESULTS

Two molecular lattices were created from a subset of 10 000 organic molecules from the Cambridge Structural Database[13] selected by the DOCK package to fit inside the active sites of *L. casei* dihydrofolate reductase[24] (Brookhaven Protein DataBank (PDB) code 4dfr) and HIV-1 protease[25] (PDB code 4hvp). Both DOCK runs were performed with standard parameter settings and the top-scoring structures (200 for HIV protease and 147 for dihydrofolate reductase) were retained for the two fused lattices. The creation of each lattice took under 30 CPU minutes on a Sun SparcStation I. Some statistics for the lattices are given in Table 3.

Some of the molecules used to create the 4hvp lattice are shown in Color Plate 1. The density of atoms drops off near the mouth of the site tunnel, where the solvent boundary lies. The several pockets within the site are all filled. The site is much larger than most of the molecules in the database that we searched, and each pocket within the site can be filled separately with small molecules that will have similar DOCK scores. A molecule that fits tightly into just one pocket will probably score well enough to be included on the list of the top 200 structures.

Table 3. Supramolecular lattice statistics for dihydrofoliate reductase (4dfr) and HIV-1 protease (4hvp)

Active Site	4dfr	4hvp
Number of structures	147	200
Number of atoms	5176	6852
Number of links	437648	526140
Mean number of links/atom	84	77

Performance of search strategies

We turn first to a consideration of search strategies. For these purposes we used the 4dfr lattice, since it describes a relatively small and geometrically constrained site. We tested the effect of different constraints upon the performance of the depth-first and breadth-first search algorithms. The performance of a search strategy can be assessed in terms of the running time of the program, the number of atoms that were visited, and how often a path between two randomly chosen atoms could be found. The use of pruning heuristics on a graph search problem will often shorten the search, but at the possible price of removing all solutions.

Breadth-first search. Two randomly selected atoms from the 4dfr lattice were used as start and target atoms. Table 4 shows the effects on breadth-first searching of different levels of pruning. As expected, prioritizing strategies have little effect in the breadth-first searches, since all nodes of a current generation are considered. Comparison of runs 4, 5, and 6 and of runs 7, 8, and 9 show that prioritizing does not direct the breadth-first search, it just adds an extra computational load. Application of pruning with angle constraints improved performance by a factor of 10, where improvement is defined as the ratio of edges followed per node. The use of chemical heuristics improved the quality of the path without much change in performance.

Another 10 runs were performed with different random pairs of atoms as the start and target nodes, tracing paths forward and then backward. A graph of running time against the number of nodes visited shows that the time dependence of the breadth-first search is approximately linear (Figure 5). The worst cases occur when no allowed path can be found and the search tree expands until memory is exhausted. The length of the search in this small sample does not seem to be influenced by the length of the path.

Comparison of the results from the forward and reverse searches indicated that our pruning rules had introduced a path dependency into the algorithm. In an ideal topologically conservative graph, a path from A to B will always be a path from B to A. In the lattice, there are often several routes between A and B. However, when nodes are pruned from the graph, an intermediate atom C between A and B might be disallowed for one path but not for another. The atom C will be available for selection only if it has not been previously part of a disallowed path. For example, if C were visited and pruned in the third generation, it could not then be used, even on a valid path, in the fourth generation. Nodes that are pruned before addition to a path are returned unseen. A node that loses all its children could be returned unseen, but that is not done in the present code. Remedial action on nodes that are more deeply embedded in the tree is also feasible but at the cost of increased computing and memory costs. It would only be justifiable if the user does not like any of the connections derived by other methods and is prepared to wait longer for a response.

Table 4. Breadth-first search results for 4dfr molecular lattice

Run number	Priority function	Prune by angle	Prune by chemistry	Number of paths	Path length	Number of nodes	CPU time(s)
1	Null	X	X		?	>8E5	—
2	Distance	X	X		?	>8E5	—
3	Angle	X	X		?	>8E5	—
4	Null	√	X	17	8	68591	150
5	Distance	√	X	17	8	68591	561
6	Angle	√	X	17	8	68591	762
7	Null	√	√	17	8	68314	157
8	Distance	√	√	17	8	68314	579
9	Angle	√	√	17	8	68314	758

Note: X = test not performed; √ = test performed; ? = no solution found

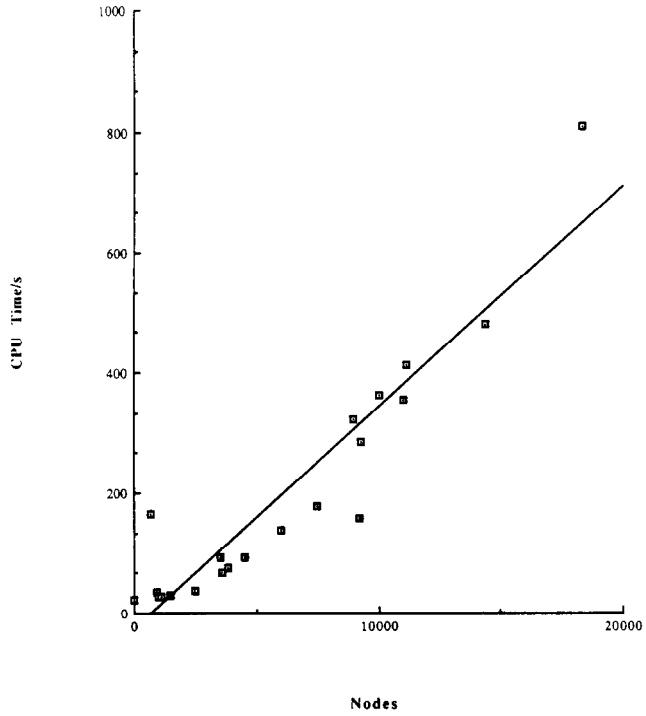

Figure 5. Run time vs. the number of nodes visited during a breadth-first search. The linear regression coefficient is 0.94

Depth-first search. The properties of the depth-first search algorithm were investigated using the same pairs of nodes, performing the searches in both forward and reverse directions. The effect of the constraint strategies is shown in Table 5. In unpruned searches (runs 1, 2, and 3), prioritizing by distance improves the search greatly, and will generally result in a shorter, more direct path. Prioritization by angle seems to be poorer as the search does not converge well, circling the target. Comparison of runs 2 and 5 show that addition of pruning strongly alters the character of the search. The most direct path is probably disallowed, so a decrease in performance is seen, but the new path found will at least be chemically reasonable. The CPU times are generally higher, reflecting the extra computations performed in pruning and prioritizing. The conditions of run 8 have been used as the standard conditions for subsequent depth-first searches.

A graph of CPU time against the number of nodes visited shows a fairly linear correlation (Figure 6); there appears to be an upper limit to the size of the tree that is produced. Even for failing runs, the tree only contained ~ 4200 nodes. The response time for depth-first search was under 3 minutes, even for the worst cases.

In comparison with the results of the breadth-first searches, a depth-first search strategy was much faster but produced paths containing one or two more atoms, and it failed more frequently. As noted above, the breadth-first algorithm is path-dependent with respect to the pruning of nodes deeply embedded in the search tree. Preliminary tests of a modified

Table 5. Depth-first search results for 4dfr molecular lattice

Run number	Priority function	Prune by angle	Prune by chemistry	Path length	Number of nodes	CPU time(s)
1	Null	X	X	13	1145	31
2	Distance	X	X	8	1717	37
3	Angle	X	X	9	969	33
4	Null	√	X	10	1388	37
5	Distance	√	X	9	1281	41
6	Angle	√	X	10	6859	37
7	Null	√	√	10	1384	36
8	Distance	√	√	9	1280	41
9	Angle	√	√	10	847	38

Note: X = test not performed; √ = test performed

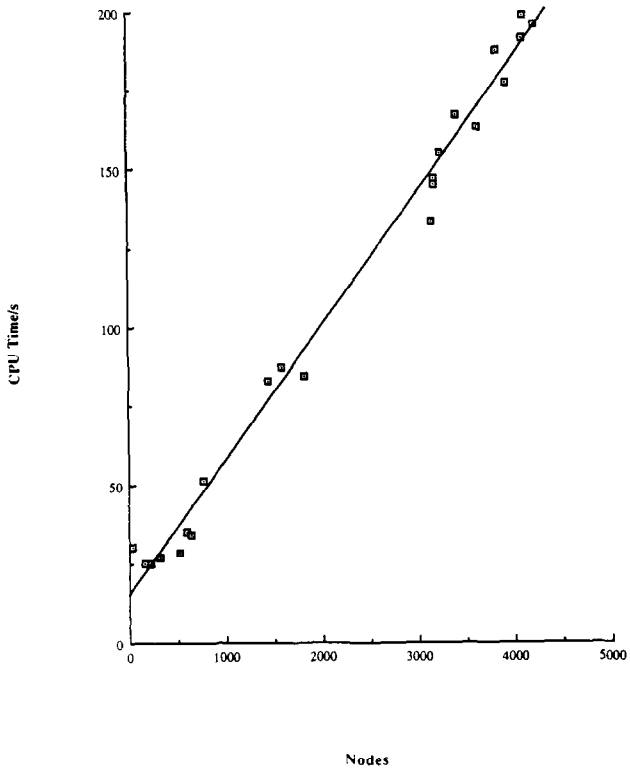

Figure 6. Run time vs. the number of nodes visited during a depth-first search. The linear regression coefficient is 0.99

strategy with decreased path dependency indicated that some of the failures could be converted into successful paths, but the number of nodes visited would often go beyond the dynamic memory limitations.

Fragment joining

A prototype version of the program was tested out on the active site of 4hvp. The lattice was created from a collection of 200 molecules derived from a standard DOCK run (Color Plate 1). A model of the active site was displayed graphically using MidasPlus and the site was evaluated by comparison with the inhibitor MVT101 present in the complex structure (Color Plate 2). The task chosen was the design of a composite structure that occupied at least two of the deep pockets in the site and that placed an atom near enough to the active site aspartyl groups that hydrogen bonds could potentially be formed. Two zones were created, one in the S_2 pocket, the second deep in the S_3 site. Ten molecules were found within zone 1. Most placed a ring within the pocket but the bulk of the structure was not located near the active site aspartyls. One compound did fill the zone with a cyclohexyl ring; it also filled the S_1 site with a tricyclic fragment and contained atoms close to the aspartyl groups (Color Plate 3). A similar situation existed for zone 2. There were a number of molecules that filled the deep tunnel. One was selected that had atoms near the center of the site (Color Plate 4).

The next stage was to join the two fragments together. Visual inspection showed that there were two atoms, one from each structure, that were 1.9 Å apart. We first joined the two molecules manually via these atoms to create a composite molecule (Color Plate 5). This structure was clearly distorted so it was subjected to a brief energy minimization within the SYBYL package (Tripos Associates, St. Louis, MO, USA). The internal energy of the connecting chain dropped from 100 kcal/mol to 5 kcal/mol. The root-mean-square difference after minimization was only 0.13 Å. The minimized composite still clearly occupied all the sites, and in addition contained an atom with the ability to bind to the active site aspartyl groups.

The two parent molecules also were joined together using the connecting algorithms described above. A subset of each molecule, defined by the aggregate used in the minimization, was taken and an anchor atom was set up for each fragment. The anchor atoms were used to seed a breadth-first search and a depth-first search through the molecular lattice. In addition, the anchor atoms were used to initiate a search through a regular diamond lattice using the beeline algorithm.[12] The distance to be spanned was 5 Å. The results are shown in Table 6 and Color Plate 6. The chains derived from the diamond lattice do not match the anchor atoms well, missing by 1 Å. This is not surprising as the gap between the anchor points is too small for this algorithm to work effectively. The chains produced by the lattice search routines join the anchor atoms exactly by several different pathways, thus offering the user a good choice of linkages. A composite molecule made from the aggregates and one of the chains was also minimized. The internal energy was reduced from 75 kcal/mol to 4 kcal/mol. These decreases in internal energies are to be expected; a linear chain has enough degrees of freedom to be able to accommodate cleanly the changes in geometry brought about by the minimization process.

The results that we have obtained with the prototype system indicate that the molecular lattice paradigm is sufficiently advanced to be tested in more sophisticated drug design projects. The lattices seem to have sufficient richness to guide the thoughts of a trained chemist. We note that the resolution of the site description may be increased by including more molecules in the original database. The program is designed to accommodate larger lattices, if necessary.

It should be stressed that this test was performed as an illustrative example only, to test the capabilities of the program. A detailed study of the site was not performed, nor were the fragments selected for their synthetic accessibility. The structures that we have suggested would surely be demanding synthetic targets, and we have made no effort to predict their binding abilities.

Table 6. Results for HIV-1 protease (4hvp)

Search method	Search direction	Number of paths	Path length	Number of nodes
BFS	Forward	5	5	975
BFS	Backward	0	10	6583
DFS	Forward	1	5	3656
DFS	Backward	1	6	1319
BEELINE*		3	6,6,7	

*Diamond lattice search[12]

DISCUSSION

The goal of this research project is to develop software tools that combine interactive design and molecular complementarity in an integrated manner. A crucial issue is how to explore and visualize the chemically plausible ways of filling a receptor site. We propose a solution that has three components. First, we describe a method to create an irregular supramolecular lattice that has several interesting properties: It fills all the pockets and invaginations of a complex site; it generally makes good van der Waals contacts with the surface of the site; the edges of the lattice are, by construction, chemically acceptable bonds. Second, we examine the problem of abstracting molecular structures from this lattice using path-finding algorithms of standard forms, modified to permit efficient and chemically reasonable searches. In keeping with our earlier work, we emphasize that the goal is to provide many molecular structures that meet user-generated constraints. This approach also permits synthetic issues to be addressed from an early phase of the design process. The third component is the novel communication scheme that permits database searches and, in fact, arbitrary calculations to proceed asynchronously but in full communication with the interactive molecular graphics program. Taken together, these tools offer an intriguing new way of developing and refining lead compounds.

An important aspect of our prototype effort was the development of heuristic rules for the generation of the molecular lattice and the lattice-search routines. While our ultimate goal would be to incorporate these rules into an expert system, at this point we have deliberately separated the individual rules into independent modules so that their performance can be evaluated under user control. For example, the search and zoning strategies have not been optimized; they should be viewed instead as guidelines that will be adjusted based on extensive testing and experience. For example, it is possible to tighten the angle constraints to any desired tolerance, but the number of paths between two nodes is a steep function of tolerance and the density of the lattice so that tight tolerances might yield no acceptable paths. Conversely, looser constraints would provide more but possibly poorer paths. We expect that the latter situation is preferable—the chemist user should have a large choice of chemical linkages—but more experience is needed to develop the proper balance of constraints. To facilitate such adjustments, all the rule bases are maintained as separate data files.

The molecular structures used for the construction of the lattice can come from a variety of sources. While we have only explored the use of the Cambridge Structural Database and DOCK, any other source of 3D data and any method of placing the molecules within the receptor site can provide nodes for the molecular lattice. We also point out that these nodes can be added after the initial lattice generation. For instance, a lattice developed from DOCK can be augmented with nodes provided from a systematic search procedure.[26]

The user also should be free to select the path-finding strategies. While our initial efforts include two of the most obvious algorithms, the modular construction of the program will permit other search methods to be readily incorporated. For depth-first and breadth-first algorithms, we are developing a function to estimate response time. We note that there is a fundamental difference in the types of structures seen when using the ZONE and LINK commands. The former will normally display molecules or rings, whereas the LINK command proposes linear chains. This reflects the use of different lattice modes by the two commands. We recognize that the presence of long chains in a potential ligand is often not desirable. It is possible to develop algorithms to brace chains with externally created rings or to select preferentially paths of high rigidity. An alternative approach would be to search the lattice at level 1, that is, as a collection of molecules linked by interatomic bridges. In this connection, the delegate program mechanism is ideal for creating menu-driven interactive software; the graphical interface allows the user to quickly rerun searches, using the information displayed as a result of previous commands.

The quality of the molecular structures developed depends on the range of structures in the database that forms the molecular lattice, as well as on the steric and chemical complementarity that the lattice achieves with the site. It also depends on the user's interpretation of the site. If the DOCK algorithm is used in the construction of molecular lattices, no molecule or atom of the lattice will clash sterically with the site atoms. Thus, shape complementarity is guaranteed. Hydrophobic, electrostatic, and hydrogen-bonding interactions must be assessed in the context of a specific receptor site and template. A hydrophobic pocket might be filled best by a branched alkyl chain; a phenyl group might have a poorer fit but may be more synthetically accessible. Our intent is that this essential tension be resolved by the chemist.

Each structure designed from the lattice will have some internal distortion, since it is formed from some combination of intramolecular fragments and intermolecular linkages. At the end of the design stage, these chimeric structures should be energetically minimized to remove the internal distortions. This step will highlight conflicts between geometry and chemistry very quickly. A program, IDATM, has been written to derive atom hybridization states from low-resolution, distorted structures.[27] It will give us the essential information needed for full minimizations. Other utilities to aid the chemist to criticize the suggestions from the program are under consideration. In the prototype system, we rely on visual assessment and energy minimization. An extension to permit molecular dynamics calculations is planned.

At present, it is still a very difficult task to predict the affinity of an untested structure or even the binding mode of a known high-affinity ligand. In practice, one would hope that a novel designed lead compound might show micromolar affinity for the receptor site. It is important then that the lead compound be readily accessible so that experimental information on the affinity of the ligand and the structure of the complex with the receptor can be quickly obtained. This information can then be used to formulate further hypotheses about how the ligand binds and to guide the design of new leads.

CONCLUSION

A novel approach to receptor-based design using irregular lattices has been described. It is based on the use of a combination of techniques from database searching and

structure generation. An essential component of the system is the graphical interface, which has been implemented efficiently using a new delegate program mechanism. This interface will allow a user to incorporate synthetic knowledge interactively into the design process. This approach will enhance user creativity and will stimulate the user to design a wide diversity of structures that are conformationally, sterically, and electrostatically reasonable.

ACKNOWLEDGEMENTS

Lewis would like to thank the Royal Commission of 1851 for a Research Fellowship and the Fulbright Commission for a Senior Scholarship. This work was partially supported by NIH grants GM-31497 (Kuntz), GM-39552 (George Kenyon, Principal Investigator), and RR-01081 (Langridge). An academic license to use SYBIL from Tripos Associates is gratefully acknowledged.

REFERENCES

1 Boobbyer, D.N., Goodford, P.J., McWhinnie, P.M., and Wade, R.C. New hydrogen-bond potentials for use in determining energetically favorable binding sites on molecules of known structure. *J. Med. Chem.* 1989, **32**, 1083–1094

2 Ferrin, T.E., Huang, C.C., Jarvis, L.E., and Langridge, R. *UCSF MidasPlus User's Manual* University of California, 1989

3 Danziger, D.J. and Dean, P.M. Automated site-directed drug design: the prediction and observation of ligand point positions at hydrogen-bonding regions on protein surfaces. *Proc. Roy. Soc. London Ser B* 1989, **236**, 115–124

4 Cohen, N.C., Blaney, J.M., Humblet, C., Gund, P., and Barry, D.C. Molecular Software and Methods for Medicinal Chemistry. *J. Med. Chem.* 1990, **33**, 883–894

5 Kuntz, I.D., Blaney, J.M., Oatley, S.J., Langridge, R., and Ferrin, T.E. A Geometric Approach to Macromolecule–Ligand Interactions. *J. Mol. Biol.* 1982, **161**, 269–288

6 DesJarlais, R., Sheridan, R.P., Seibel, G.L., Dixon, J.S., Kuntz, I.D., and Venkataraghavan, R. Using Shape Complementarity as an Initial Screen in Designing Ligands for a Receptor Binding Site of Known Three-Dimensional Structure. *J. Med. Chem.* 1988, **31**, 722–729

7 DesJarlais, R.L., Seibel, G.L., Kuntz, I.D., Montellano, P.O.D., Furth, P.S., Alvarez, J.C., DeCamp, D.L., Babé, L.M., and Craik, C.S. Structure-based design of nonpeptide inhibitors specific for the human immunodeficiency virus 1 protease. *Proc. Natl. Acad. Sci. USA* 1990, **87**, 6644–6648

8 Lewis, R.A., and Dean, P.M. Automated site-directed drug design: the concept of spacer skeletons for primary structure generation. *Proc. Roy. Soc. London Ser B* 1989, **236**, 125–140

9 Lewis, R.A., and Dean, P.M. Automated site-directed drug design: the formation of molecular templates in primary structure generation. *Proc. Roy. Soc. London Ser B* 1989, **236**, 141–162

10 Langridge, R., Ferrin, T.E., Kuntz, I.D., and Connolly, M.L. Real-Time Color Graphics in Studies of Molecular Interactions. *Science* 1981, **211**, 661–666

11 Ferrin, T.E., Huang, C.C., Jarvis, L.E., and Langridge, R. The MIDAS display system. *J. Mol. Graph.* 1988, **6**, 13–27

12 Lewis, R.A. Automated site-directed drug design: approaches to the formation of 3D molecular graphs. *J. Comp.-Aided Drug Design* 1990, **4**, 205–210

13 Allen, F.H., Bellard, S., Brice, M.D., Cartwright, B.A., Doubleday, A., Higgs, H., Hummelink, T., Hummelink-Peters, B.G., Kennard, O., Motherwell, W.D.S., Rodgers, J.R., and Watson, D.G. The Cambridge Crystallographic Data Centre: Computer-Based Search Retrieval, Analysis, and Display of Information. *Acta Crystallogr. Sect. B* 1979, **B35**, 2331–2339

14 Singh, J. and Thornton, J.M. SIRIUS: An automated method for the analysis of the preferred packing arrangements between protein groups. *J. Mol. Biol.* 1990, **211**, 595–615

15 Van Drie, J.H., Weininger, D., and Martin, Y.C. ALADDIN: an integrated tool for computer-assisted molecular design and pharmacophore recognition from geometric, steric and substructure search of three-dimensional molecular structures. *J. Comp. Aided Mol. Des.* 1989, **3**, 225–51

16 Jakes, S.E., Watts, N., Willett, P.J., Bawden, D., and Fisher, J.D. Database searching. *J. Mol. Graph.* 1987, **5**, 41–48

17 Lewis, R.A.. and Kuntz, I.D. Site-directed computer-aided drug design: progress towards the design of novel lead compounds using molecular lattice. In *Scientific Computering and Automation* (E.J. Karalainen, ed.) Elsevier, Amsterdam, 1991, p 117–132

18 Allen, F.H., Kennard, O., Watson, D.G., Brammer, L., Orpen, A.G., and Taylor, R. Chemical bonds. *Chem. Soc. Perkins Trans. II* **1988**, *S*1–S19

19 Weiner, P.K. and Kollman, P.A. AMBER: Assisted Model Building with Energy Refinement. A General Program for Modeling Molecules and Their Interactions. *J. Comp. Chem.* 1981, **2**, 287

20 Weiner, S.J., Kollman, P.A., Case, D.A., Singh, U.C., Ghio, C., Alagona, G. Profeta, S., and Weiner, P. A New Force Field for Molecular Simulation of Nucleic Acids and Proteins. *J. Am. Chem. Soc.* 1984, **106**, 765–784

21 Goodford, P.J. A Computational Procedure for Determining Energetically Favored Binding Sites on Biologically Important Macromolecules. *J. Med. Chem.* 1985, **28**, 849–857

22 Sedgewick, R. *Algorithms* Addison-Wesley, London, 1984

23 Lindsay, R.K., Buchanan, B.G., Feigenbaum, E.A., and Lederberg, J. *Applications of Artificial Intelligence for Organic Chemistry: the DENDRAL project* McGraw-Hill, New York, 1980

24 Bolin, J.T., Filman, D.J., Matthews, D.A., Hamlin, R.C., and Kraut, J. Crystal Structures of *E. Coli* and *L. casei* DHFR refined to 1.7-Angstrom resolution. 1. General features and binding of Methotrexate. *J. Biol. Chem.* 1982, **257**, 13650–13662

25 Miller, M., Schneider, J., Sathyanarayana, B.K., Toth, M.V., Marshall, G.R., Clawson, L., Selk, L. Kent,

S.B.H., and Wlodawer, A. Structure of Complex of Synthetic HIV-1 Protease with a Substrate-Based Inhibitor at 2.3-Å Resolution. *Science* 1989, **246**, 1149–1152
26 Humblet, C. and Marshall, G.R. Three-Dimensional Computer Modeling as an Aid to Drug Design. *Drug Development Res.* 1981, **1**, 409–434
27 Meng, E.C. and Lewis, R.A. Determination of Molecular Topology and Atomic Hybridization States from Heavy Atom Coordinates. *J. Comp. Chem.* 1991, **12**, 891–898.

APPENDIX A. DELEGATE PROGRAMMING WITH MIDASPLUS

User's perspective

Delegates are controlled from MidasPlus using the **delegate** command, which supports three operations: **start**, **stop**, and **list**. New delegates are invoked with the command:
delegate start *delegate_name command arguments*
where *delegate_name* is the name that MidasPlus will use to refer to the delegate process, and *command* and *arguments* are the Unix command to execute the delegate program. Once a delegate is running, the user can send commands to it by prefixing the command with the name of the delegate, that is,
delegate_name delegate_command arguments

To terminate an active delegate, the user can issue the command
delegate stop *delegate_name*

Finally, the command
delegate list
lists the names of all active delegates.

Implementor perspective

When a delegate is executed, a bidirectional communications channel is established. The standard input (C-language standard I/O library *stdin* or Unix file descriptor 0) and standard output (*stdout* or descriptor 1) are connected to MidasPlus. Data sent to *stdout* will be interpreted by MidasPlus. Data received on *stdin* will be either commands from the user (via the *delegate_name* mechanism), or replies from MidasPlus commands normally displayed to the user.

The communications protocol between the delegate and MidasPlus is very simple. An event diagram of the protocol is shown in Figure 7. On start-up, the delegate sends lines of commands to MidasPlus for execution. For each line of command, MidasPlus sends back to the delegate all the replies, followed by a line containing only the word **SYNC**. When the delegate is finished with its initialization, it informs MidasPlus by sending a line containing only the word **SYNC**. At this point, the delegate waits for user commands to arrive on *stdin*.

When the delegate receives a user command, it sends lines of commands to MidasPlus, using exactly the same synchronization as on start-up. MidasPlus will not accept any input from the user between the time it forwards a user command to the delegate and the time it receives the terminating **SYNC** message from the delegate. Thus, the state of the graphics program will not change spontaneously while the delegate program is processing user commands.

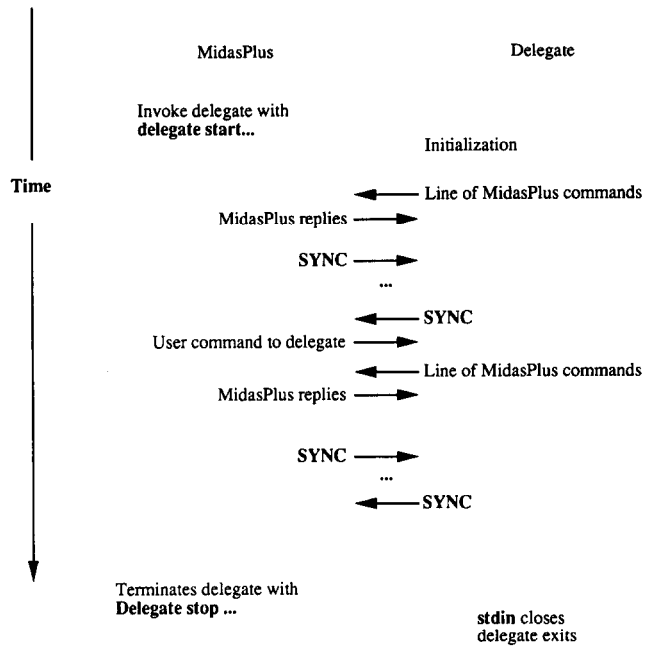

Figure 7. Event diagram of the delegate program communication protocol

Normally, a delegate is active only after receiving a command. However, while waiting for a user command, a delegate also may generate MidasPlus commands spontaneously. This is used for programs that need a substantial amount of computation time before yielding results and that run independently of user input. Under these circumstances, the user can manipulate the graphics image while the computation occurs. These delegates can send a **SYNC** in response to a command, enter background computation mode, and then spontaneously generate MidasPlus commands once the calculation completes. The only potential drawback to using background computation in this manner is that it could result in a race condition where the user sends additional commands to the delegate that may be interpreted as a reply to a MidasPlus command. Implementors of delegates should clearly document which user commands employ background computation modes and warn the user at run time that further input to the delegate program is not advisable.

As an illustration of the type of commands that have been developed, there is a rotation delegate procedure that supports two commands: SNAPSHOT and INTERPOLATE. Users can use the SNAPSHOT command to save molecule positions at selected points during a session and the INTERPOLATE command to smoothly interpolate between two saved positions. While it would be possible to add commands such as SNAPSHOT and INTERPOLATE directly to MidasPlus, that would require the programmer to have both an understanding of its internal program structure and permission to change source files. Writing the rotation delegate needed no modifications to MidasPlus, and, once the matrix arithmetic was solved, took less than a day to implement. Less than 300 lines of new code (including comments) were required.

Source code for a simple delegate program is given in Appendix B.

APPENDIX B. A SAMPLE DELEGATE PROGRAM

An example of a delegate that simply echoes user input follows.

```
#include <stdio.h>
#include <stdarg.h>

#ifndefTRUE
#define     TRUE      1
#define     FALSE     0
#endif

#define     BUFSIZ 256

FILE    *record_fp;

/*
 *Sample MidasPlus delegate
 */
main(int ac, char **av)
{
        register int    c;
        int             verbose = FALSE;
        char            buf[BUFSIZ];
        extern int      optind;
        extern char     *optarg;
        static int      send_command(char *, ...);


        /*
         * Process command line arguments
         */
        while ((c = getopt(ac, av, ''vr:''))!=EOF)
                switch (c) {
                 case 'v':
                        verbose = TRUE;
                        break;
                 case 'r':
                        record_fp = fopen(optarg, ''w'');
                        if (record_fp != NULL)
                                setbuf(record_fp, (char *) NULL);
                        break;
                }

        /*
         * First we sync with MidasPlus, which is expecting us to notify
         * it of proper start-up
         */
        (void) printf(''SYNC\n'');
        (void) fflush(stdout);

        /*
         * We simply echo back any commands to MidasPlus. If we are
         * in verbose mode, we notify the user before and after
         * each command
         */
        while (fgets(buf, sizeof buf, stdin) != NULL) {
                if (verbose)
                        send_command(''echo Delegate executing %s'', buf);
                send_command(buf);
                if (verbose)
```

```
                send_command(''echo Delegate done\n'');
            (void) printf(''SYNC\n'');
            (void) fflush(stdout);
        }

        exit(0);
}

/*
 * send_command:
 *      Send a command to MidasPlus and wait until the
 *      synchronizing string comes back
 */
static
int
send_command(char *fmt, ...)
{
        va_list args;
        char    buf[BUFSIZ];

        va_start(args, fmt);
        (void) vfprintf(stdout, fmt, args);
        if (record_fp != NULL) {
                fputs(''>'', record_fp);
                vfprintf(record_fp, fmt, args);
        }
        va_end(args);
        (void) fflush(stdout);          /* Make sure buffered data get sent */

        /*
         * Keep reading until we see a line containing only SYNC,
         * which denotes end of MidasPlus replies
         */
        while (fgets(buf, sizeof buf, stdin) != NULL) {
                if (record_fp != NULL)
                        fprintf(record_fp, ''<%s'', buf);
                if (strcmp(buf, ''SYNC\n'') == 0)
                        break;
        }

        return;
}
```