Conformational analysis using a truth maintenance system

Timothy Koschmann,* James P. Snyder,† Peter Johnson, Thom Grace‡ and Martha W. Evens‡**

Xerox AI Systems, 3000 Des Plaines Ave., Des Plaines, IL 60018, USA*; Drug Design, Searle Research and Development, 4901 Searle Parkway, Skokie, IL 60077, USA†; Department of Chemistry, Illinois Institute of Technology, Chicago, IL 60616, USA**; Computer Science Department, Illinois Institute of Technology, Chicago, IL 60616, USA‡

This paper describes Rotamer, a program we wrote to assist chemists in identifying the likely three-dimensional (3D) shapes (conformations) that a flexible molecule might assume. The search of the space of all possible conformations is approached as a constraint-satisfaction problem. The structure generator in Rotamer uses a simplified truth maintenance system (TMS) to cache information acquired in the course of exploring the conformation space. The computational benefits of using this TMS increase with the size of the problem.

Keywords: conformational analysis, truth maintenance systems, exhaustivite search, backtracking, thrashing, rediscovery

Received 6 November 1987

Three types of procedures are commonly used to explore the space of possible geometries that a molecule might assume. The simplest procedure involves performing an exhaustive search of the conformation space. Programs that contain a so-called dihedral driver represent examples of this type of approach.[1] Unfortunately, the number of possible geometries increases exponentially with the number of rotatable bonds in the molecule. If there are n rotatable bonds in the molecule and for each bond b_i there are m_i torsion angles, then the number of candidate geometries would be equal to

$$\prod_{i=1}^{n} m_i$$

which reduces to m^n if all m_i are equal to a constant. For larger molecules, chemists sometimes resort to using statistical search techniques, such as the Monte Carlo method, to explore the conformation space.[2] A third approach, known as "distance geometry,"[3] performs a conformational analysis using numeric optimization techniques. This approach starts by specifying upper and lower bounds for all interatomic distances within the molecule and then computes a (possibly empty) set of solutions that satisfy the problem.

Many problems in artificial intelligence (AI) can be construed as large search problems.[4] As a consequence, much work in AI has been devoted to producing efficient heuristic search techniques. Conformational analysis can be viewed as an example of a class of problems in AI research known as "constraint-satisfaction problems."

This paper describes an effort to develop a new algorithm for performing conformational analyses. Our approach differs from traditional generate-and-test algorithmns in that the program we used can preserve, for later reference, information acquired in the course of performing an analysis. The program can accumulate information about a specific problem as it solves the problem — in effect, learning to search "smarter."

We developed the program, known as Rotamer, using techniques borrowed from the area of research in AI known as truth maintenance systems (TMSs).[5] TMSs were originally developed to provide a convenient way of maintaining a consistent set of beliefs, given that all observations are subject to change and might be faulty.[6,7] TMSs allow a reasoning system to maintain more than one point of view of a problem. They have been used in expert system construction, constraint-satisfaction problems, and in qualitative reasoning about physical systems.

We will first define constraint-satisfaction problems and show how conformational analysis can be viewed as a type of constraint-satisfaction problem. We then describe how TMSs (one technique used in AI research for solving constraint-satisfaction problems) can be applied to conformational analysis. Finally, we argue that TMSs can be an important tool for solving large search problems in chemistry.

CONSTRAINT-SATISFACTION PROBLEMS

Constraint-satisfaction problems are characterized by a set (often quite large) of possible solutions referred to as "the solution space" and a list of requirements of constraints by which solutions are selected. In a constraint-satisfaction problem, there are usually several variables, each of which can assume a number of

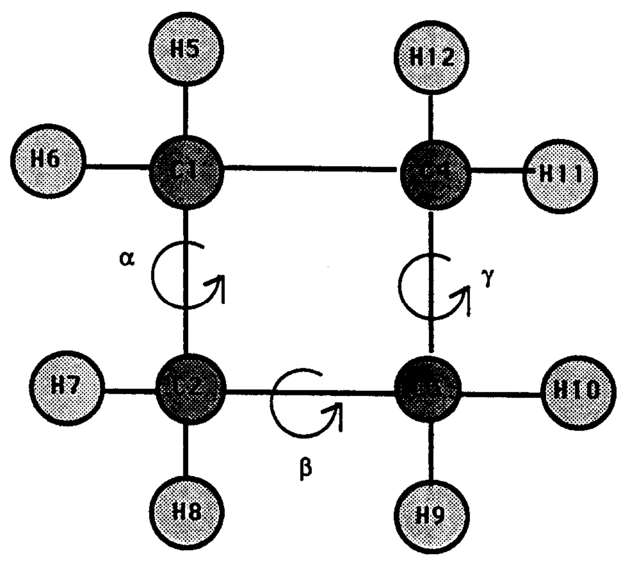

Figure 1. A sample problem based on cyclobutane

different values. The set of values that a variable can assume is referred to as the "domain" of the variable. Each potential solution in the solution space represents a unique set of assignments for all of the variables in the problem. Some constraints dictate acceptable values for a single variable (often termed a "unary"constraint), while other constraints might affect assignments to several variables (e.g., "binary," "ternary," or other higher-order constraints).

In the case of a conformational analysis problem, the variables represent the torsion angle assignments associated with the rotatable bonds. The domain of these variables is the set of possible molecular geometries (both realizable and imaginary) produced by assigning the various torsion angles to the rotatable bonds of the molecule. The constraints are a set of geometric and chemical rules that describe a stable conformation.

As a very simple example of a conformational analysis problem, consider the exploration of the possible orientations of the molecule cyclobutane. Label the atoms of the molecule as shown in Figure 1. For the purposes of this analysis, we will perform rotations around bonds C1-C2, C2-C3, and C3-C4, allowing the bond between C4 and C1 to vary in length as a function of the rotations around the other three bonds.

In this analysis, we will rotate each of the three bonds C1-C2, C2-C3, and C3-C4 in increments of 30 degrees. We will label the torsion angle associated with the C1-C2 bond α. Similarly, we will label the torsion angles associated with bonds C2-C3 and C3-C4 β and γ. Since each of these bonds can then assume one of twelve possible torsion angles, there are 12^3, or 1 728, combinations of torsion angles to be considered. Most of the molecular geometries produced by these combinations of torsion angles are not physically realizable for one of several reasons: (1) one or more atoms would be superimposed, (2) C1 and C4 would be separated too far in space to remain bonded or (3) the angle formed by atoms C3, C4 and C1 would fall outside of the acceptable range. The distances between pairs of atoms form the basis for most of the rules that will be used to reject candidate conformations.

The simplest approach to solving constraint-satisfaction problems is referred to as "generate and test." This approach involves combinatorially generating all possible solutions and then testing each to see if it satisfies the set of constraints. This procedure has the property of "exhaustivity." It is guaranteed to find all solutions. The standard algorithms for performing the generate and test procedure are based on the ideas of "backtracking."

In backtracking algorithms for constraint resolution, the constraint variables and the values of their respective domains are assumed to be represented in an ordered fashion. The algorithm proceeds by sequentially assigning values to each of the constraint variables until all variables have been instantiated. The constraints are then applied to see if this set of assignments qualifies as a solution. If not, the algorithm "backs up" to the last instantiated variable and tries the next value from the ordered domain set. If the domain set for this variable has been exhausted, then a domain value from the next-to-the-last variable is tested and the algorithm recycles through all of the domain values of the last variable. The algorithm terminates when it has repeated this cycle back up through all of the variables to the domain of the first variable. Backtracking algorithms are simple to implement, and they are guaranteed to find all possible solutions to a constraint-satisfaction problem.

If constraint satisfaction is deferred until all variables have been instantiated, backtracking is merely a depth-first traversal of the solution tree. Many times it is wasteful to completely generate a full solution before testing because it could be seen *a priori* to violate some simple unary or binary constraint. If the solution space is viewed as a tree, whole subtrees can often be removed from consideration without sacrificing exhaustivity.

However, even with early testing, backtracking algorithms do not represent the optimal approach to constraint resolution. The algorithm presented above is sometimes referred to as "chronological backtracking" to distinguish it from other forms of backtracking. Chronological backtracking algorithms are subject to two problems known as "thrashing"[4] and "rediscovery."[6]

Thrashing occurs when a partial solution fails: The algorithm always tracks back to the most recently instantiated variable (hence the name "chronological" backtracking). Often, however, the cause of the failure is several branches higher in the search tree. Chronological backtracking algorithms must cycle through all of the domain values of all the constraint variables that lie between the point where the failure first occurred and the assignment that led to the failure. None of the combinations that are generated in the process of discovering where this failure occurred will be valid.

Rediscovery occurs when combinations of assignments continue to be generated even though a subset of those assignments has been found to lead to a constraint violation. In chronological backtracking, assignments to the last variables considered that lead to constraint violations will be "rediscovered" time and again. Suppose one of the assignments to the variable γ results in atoms C2 and H12 violating a minimum interatomic distance constraint. If the variables were evaluated in the order α, β, γ, then this violation would be detected only after completely generating and testing all of the molecule's subassemblies. This computationally expensive process would be repeated 62 times when it should have to be done only once!

Historically, two different techniques have been proposed to deal with the above-mentioned weaknesses of chronological backtracking. So-called relaxation techniques apply some or all constraints as filters to reduce the sizes of problem domains.[8] Relaxation techniques can be used prior to a backtrack search or they can be extended to produce a solution set without ever resorting to a backtracking search. The second group of techniques, which we will refer to as "caching techniques," retain information learned in the process of solving the problem in order to make the search more efficient.

Caching methods avoid thrashing by storing information obtained in the course of solving a constraint problem. The technique known as "dependency-directed backtracking" is based on the idea of retaining information about variable assignments that have previously failed in order to avoid retesting of these assignments while backtracking.[4] This requires some more bookkeeping, yet for large problems, it can greatly improve the efficiency of a search. Unfortunately, dependency-directed backtracking is still subject to certain inefficiencies. Rediscovery of combinations of variable assignments that lead to failure can still occur.[6]

TRUTH MAINTENANCE SYSTEMS

TMSs represent a more elaborate means of caching information obtained in the course of performing a search.[6,7] In a TMS, each assignment of a domain value to a constraint variable is treated as an "assumption." Other observations that can be made given the assumptions are referred to as "derived data." An "environment" is defined by a set of assumptions that are held to be true within the TMS. If a contradiction can be derived from the assumptions in an environment, the environment is referred to as a "no-good environment," or simply as a "no-good." The TMS maintains a database of assumptions and combinations of assumptions that lead to contradictions.

The environments of a given problem can be organized into a lattice that shows the relationships between environments (see Figure 4). The "empty environment" appears at the top of the lattice: It is the environment in which no assumptions are made. If a contradiction can be derived within the empty environment, then all other environments are said to be "subsumed" by the empty environment. The second row of the lattice is made up of all "singleton" environments, that is, environments with only one assumption. If there are n variables with a fixed domain size of m, then the second row will always have nm environments. Continuing down the lattice, each subsequent row will have environments with one more assumption.

The arcs connecting environments indicate that an "is subsumed by" relationship exists between the lower environment and the upper environment. This relationship implies that anything that can be derived in the upper environment can also be derived in the lower environment, but not the reverse. The environment defined by the assumption set $\{a,b\}$ is subsumed both by the environment that has the assumption set $\{a\}$ and by the environment that has the assumption set $\{b\}$. Anything that can be derived given only assumption a can also be derived in an environment that has both a and b. However, clauses that can only be derived given the conjunction of a and b will exist only in the $\{a,b\}$ environment. An environment is considered to be the maximal environment (also called the "greatest lower bound") for a specific datum if it is the environment with the fewest assumptions in which the datum can still be derived.[6]

The last row of the lattice represents environments that have an assumption corresponding to each of the constraint variables. In a constraint resolution problem, these variables represent solutions. In a constraint problem there will be m^n of these environments, where n is the number of constraint variables and m is the domain size. We refer to these as "terminal environments." We will call all other nonterminal environments "subterminal environments."

Use of a TMS solves the problems identified earlier for backtracking search algorithms. The order in which the variables are processed is immaterial as long as the environments are processed starting from the empty environment and working up to the terminal environments. Traversing the environments from the top down ensures that they will always be maximal for any observation that is derived. This makes it possible to discover contradictions at the earliest possible time. Rediscovery will not occur under these circumstances. The thrashing behavior seen in a backtracking search will also be eliminated. A backtracking search can be viewed as performing a depth-first traversal of the environment lattice; the TMS-based search proposed here would be more akin to a breadth-first traversal.

To understand how to apply a TMS to a conformational analysis problem, it is important to understand a geometric feature of the molecule structures called "local data dependencies." Assigning a new value to α will change the position in space of all the atoms in the molecule except for C1, H5 and H6, which will remain in fixed positions. The distance between C1 and C2 will not change, however. Indeed, the distances between all bonded atoms are invariant, except for the distance between C4 and C1, which we are treating as a special case. The distance between H5 and C2 will also remain constant through all rotations around the C1-C2 bond. We term these "1-3" distances because H5 and C2 are in the first and third positions of a three-atom chain with C1 in the middle. The distance between H5 and H7 will change as a function of α,

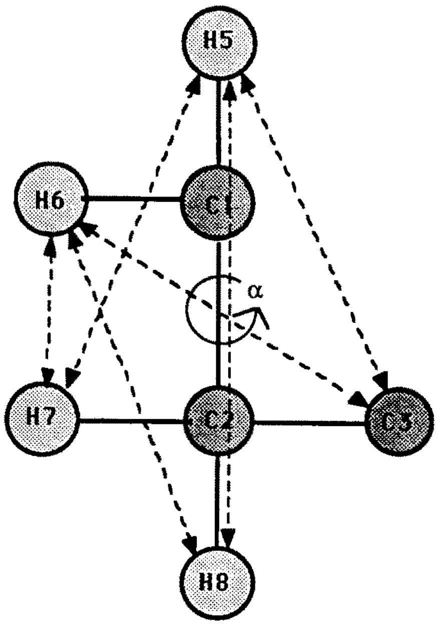

Figure 2. Interatomic distances determined by the value of α

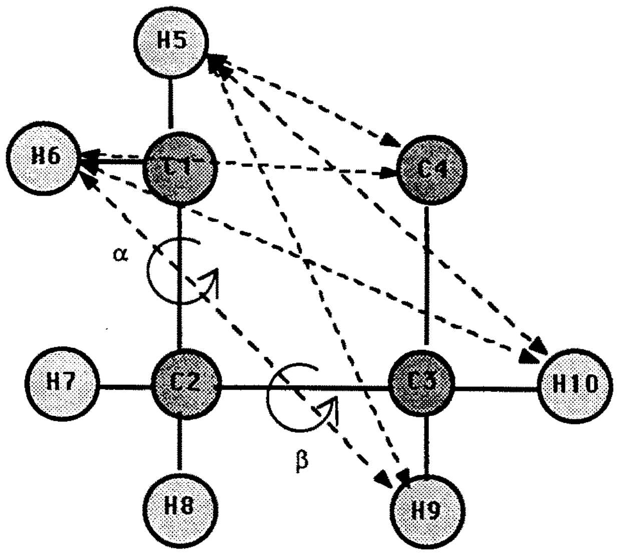

Figure 3. Interatomic distances determined by the combined values of α *and* β

as will the distances between H5 and H8, H5 and C3, H6 and H7 and so on. We refer to these as "1-4 distances," because H5 and H7 are in the terminal positions of a four-member chain. Figure 2 shows the set of interatomic distances that are determined by angle of rotation around the C1-C2 bond. These distances are therefore locally dependent on the value assigned to α. As Figure 3 shows, the interatomic distance between H6 and H9 is determined by the angles assigned to both α and β. By analogy to our previous terminology, we could call this a "1-5 distance." Bond distances (i.e., "1-2 distances") and 1-3 distances are invariant under torsion angle changes. What we have termed 1-4 distances are a function of rotation around a single bond. Pairs of adjacent rotated bonds will determine 1-5 distances. This analysis can be extended to sets of three or more contiguous rotatable bonds. The notion of local data dependencies is fundamental to understanding how Rotamer uses a TMS to solve a conformational analysis problem.

In conformational analysis, the topological structure of the molecule under consideration, the types of each of the atoms of the molecule, the list of rotatable bonds, the starting geometry of the molecule and the set of constraints that define the problem are treated as premises. Each possible assignment to a constraint variable is viewed as an assumption in the TMS. Everything else, including computed atomic positions, interatomic distances, estimated conformational energies and bond angles, is treated as derived data.

Rotamer uses geometric constraints, energetic constraints, and symmetry-related constraints. Geometric constraints are based on relative positions of atoms. The simplest geometrical constraints are based on minimum and maximum interatomic distances, similar to those used for distance geometry calculations.[3] Constraints can also be placed on more complex geometric features, such as bond angles and torsion angles. Energetic constraints can be used to eliminate unstable geometries. The procedures that Rotamer uses to estimate the potential energies of candidate solutions are described elsewhere.[5] A constraint can be applied so that no geometries will be considered whose potential energies exceed the lowest potential energy determined by some fixed threshold value. Finally, symmetry constraints are used as a means of eliminating solutions that are geometrically equivalent but differ only in the labeling of the atomic centers.

Figure 4 shows the environment lattice for the cyclobutane problem. The distances between bonded atoms and 1-3 interatomic distances represent data that can be derived in a TMS empty environment. If a contradiction is derived here, then no solutions will be found that can satisfy the problem constraints. The second row of the lattice represents all singleton environments. For example, in the singleton environment in which α is assumed to be 60°, we can also compute 1-4 interatomic distances for the atoms attached to C1 and C2. If any of these distances violate minimum distance constraints, we can remove from further consideration all environments that are subsumed by this environment (i.e., {α = 60, β = 0}, {α = 60, β = 0, γ = 0}, etc.).

Rotamer uses the following algorithm to perform a conformational analysis using a TMS. Starting from the empty environment and working down the environment lattice, Rotamer tests to see if each environment is subsumed by a no-good environment. If not, it tests to see if a contradiction can be derived in the environment (i.e., it tests to see if the assumptions held in this environment lead to violations of any of the problem constraints). If a contradiction is found, the terminal environments that are not subsumed by a no-good environment and that do not violate any of the problem constraints are considered candidate solutions. The advantage of using a TMS to solve the problem is that

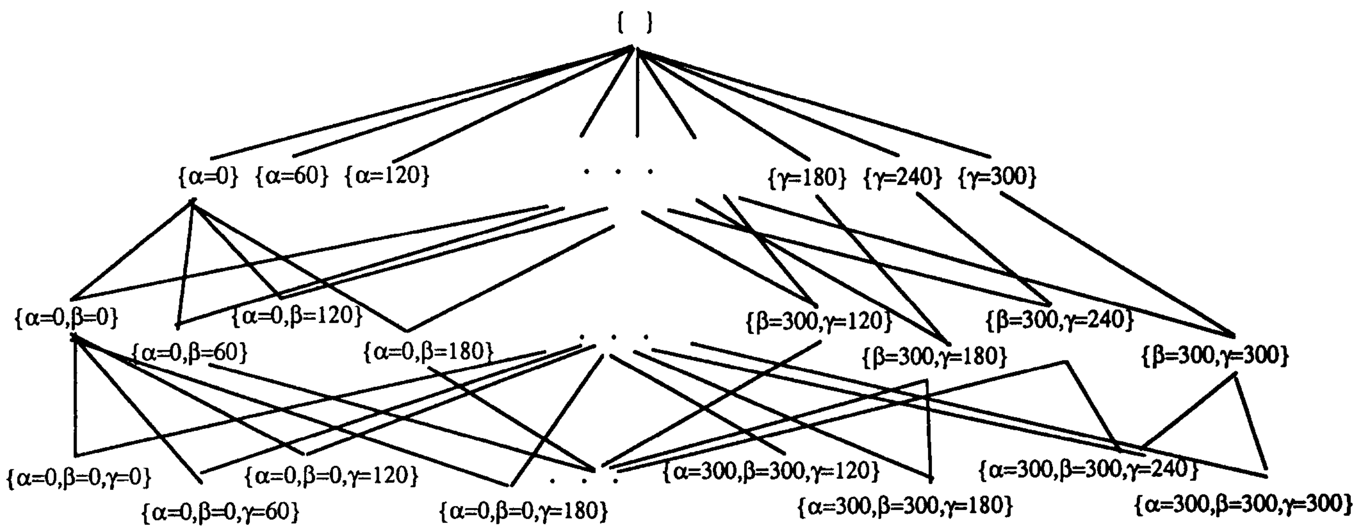

Figure 4. A TMS environment lattice for the cyclobutane problem

the vast majority of the possible molecular geometries represented as terminal environments are subsumed by no-good environments and hence need never be created or tested. An easy parallel implementation of this algorithm is possible, since all environments in any given row of the environment lattice can be processed simultaneously.

ANALYZING PERFORMANCE

The Rotamer program was developed on a Xerox 1186 Lisp Machine in both Lisp and Prolog. The system was implemented using an object-oriented programming extension to Lisp known as LOOPS.™ Objects were used to represent molecules and conformations, to create the chemist's interface to the system and to implement the TMS.[5] We used Prolog rules to represent problem constraints and data derived within the TMS environments.

We used Rotamer to perform a conformational analysis on cyclobutane. Each of the rotatable bonds was rotated in increments of 30° to produce a set of 12 possible torsion angle assignments for each bond, actually generating 1 728 possible geometries. The total number of environments — including the empty environment, the subterminal environments and the terminal environments — came to 2 053.

Of the environments, 84 were eliminated due to minimum/maximum interatomic distance constraints, and no environments were eliminated due to closure angle constraints. Symmetry constraints were not applied to this problem. Of the remaining environments, 1 800 were eliminated by subsumption by no-good environments. The solution set consisted of 15 terminal environments.

Most solutions were puckered rings. Relative potential energies were estimated, using the methods described elsewhere,[5] for the remaining terminal conformations. The published minimum energy conformation for cyclobutane is a puckered ring with a pucker angle of approximately 30°.[1] We assume that this conformation would be produced if the low-energy geometries computed by Rotamer were minimized using a properly parameterized molecular mechanics program.

As mentioned earlier, the number of possible geometries that need to be considered in a conformational analysis problem is equal to

$$\prod_{i=1}^{n} m_i$$

where n is the number of rotatable bonds and m_i is the number of possible torsion angle assignments for bond b_i. In the most common case, where the number of possible torsion angle assignments is a constant for all rotatable bonds, the number of possible solutions is m^n.

Unfortunately, Rotamer must deal not only with the terminal environments but with all the subterminal environments as well. The environment lattice always has $n + 1$ rows. The first row, of course, has only the empty environment. The second row will have nm singleton environments corresponding to the set of all possible assignments taken one at a time. In general, the number of environments in a row will be equal to

$$\binom{a}{nm}$$

where a is the number of sets of assumptions for all environments in the given row. Rotamer does not need to consider all of these possible environments. First, only environments with no more than one assignment to a variable class are allowed. Second, in Rotamer, meaningful observations can be derived only in environments that have adjacent bonds. Consequently, the number of subterminal environments investigated is substantially smaller. The formula for the number of environments actually considered by Rotamer within a given row is

$$[(n - a) + 1]m^a$$

Applying this formula, the total number of subterminal environments for a problem with the same domain for all constraint variables is

$$1 + \sum_{a=1}^{n-1}(n - a) + 1]m^a$$

where n, m and a are defined as before. The ratio of the number of subterminal environments to the number of terminal environments can be considered to be a measure of the amount of overhead that is added to a conformational analysis problem when it is processed using the TMS in Rotamer. For very small problems, this ratio can exceed one, indicating that the use of a TMS is actually less efficient than solving the problem using exhaustive search. However, this ratio decreases as n or m increases. The ratio for the cyclobutane problem above, for instance, was 0.19. Therefore, the advantage of using a TMS, at least theoretically, increases with the size of the problem. Unfortunately, as currently implemented, Rotamer can solve only relatively small problems.

DISCUSSION

Viewing conformational search as a constraint resolution problem is not an idea that is unique to Rotamer. The upper and lower bounds used in the distance geometry approach[3] represent a set of constraints. The difference lies in how the constraints are used. In the distance goemetry method, a numerical optimization procedure is used to iteratively find solutions that satisfy the constraints. Typically, weights and penalty functions are used to enhance or reduce the impact of specific constraints during the course of optimization. In Rotamer, constraints are expressed symbolically as Prolog rules. For this reason, a greater variety of types of constraints can be used in Rotamer than in distance geometry programs. New constraints can be added and old constraints can be modified or eliminated to solve specific problems. Since problems are solved symbolically, program maintenance is simplified and the chemist is allowed much more flexibility in solving problems.

Recent articles[9,10] describe Wizard, another symbolic reasoning program for conformational analysis. Wizard, like Rotamer, uses a rule-based approach to propose and test possible conformations of small organic molecules. The algorithm used in Wizard differs in one significant respect from the search algorithm in Rotamer. Wizard works with molecular substructures, referred to as "recognize units," rather than operating on the atomic level. In effect, this provides a hierarchical reorganizaton of the search space. Hierarchical reorganization of the search space is a well-known technique for reducing the complexity of large search problems.[3] It offers clear computational benefits in performing a conformational analysis. An interesting direction for future research might be to explore the combination of Wizard's hierarchical search with the truth maintenance techniques employed in Rotamer.

The Wizard program appears to have reached a higher level of sophistication with respect to its knowledge of chemistry than the Rotamer program. A recent article[9] provides structures and benchmark data for a variety of mid-sized, cyclic molecules that have been studied using the Wizard program. To date, Rotamer has been applied only to some very small cyclic alkane rings.

The Rotamer project offers a new way of looking at conformational analysis. It has demonstrated that the problem of conformational analysis can be transformed into a constraint resolution problem that can be solved with truth maintenance techniques. Work remains to be done before programs like Rotamer will become truly useful tools for performing conformational analysis, but the results so far suggest that TMSs may be worthy of further investigation as tools for computational chemistry.

ACKNOWLEDGEMENTS

Johan de Kleer offered helpful comments concerning the design of the Rotamer system. Clark Still provided several Fortran programs that were used to construct test cases for Rotamer. The good folks in the Computer Center at Chicago Medical School gave us needed assistance in the production of this manuscript. Tuition support for this project came from Xerox Corporation.

LOOPS and 1186 are trademarks of Xerox Corporation.

REFERENCES

1 Burkert, U., and Allinger, N. Molecular Mechanics (ACS Monograph #177) Amer. Chem. Soc., Washington, D.C., 1982
2 Saunders, M. *J. Am. Chem. Soc.*, 1987, **109**, 3150–3152
3 Crippen, G. *Distance Geometry and Conformational Calculations. New York, Research Studies Press*, 1981
4 Stefik, M. *Introduction to Knowledge Systems* (in preparation)
5 Koschmann, T. Conformational analysis using a simplified truth maintenance system implemented with object-oriented and logic programming. Ph.D. Dissertation, Illinois Institute of Technology, 1987
6 de Kleer, J. *Artificial Intelligence*, 1986, **28**, 127–162
7 Doyle, J. *Artificial Intelligence*, 1979, **12**, 231–272
8 Mackworth, A. *Artificial Intelligence*, 1977, **8**, 99–118
9 Dolata, D., Leach, A., and Prout, K. *J. Computer-Aided Molecular Design, 1987*, **1**, 73–85
10 Dolata, D., and Carter, R. *J. Chem. Infor. and Comp. Sci.*, 1987, **27**, 36–47