An Algorithm for Finding the Intersection of Molecular Structures

By **Malcolm Bersohn**, University of Toronto, Toronto, Canada M5S 1A1

A computer can be used to find the largest common fragment of two molecular structures, by a method which initially constructs a list of those atoms in one molecule which have a matching atom in the other molecule and whose neighbours match the corresponding neighbours in the other molecule. Forming the set of all atoms which are on this list and whose neighbours are on the list produces a new list. Next we form the set of all atoms which are on the new list and whose neighbours are also on the new list. After a very few interations of this process we are left with a short list, usually a single atom, which is the centre of the intersection. The atom central to an intersection having been determined, finding the intersection specifically is then a brief process.

If we consider a molecular structure as a graph, *i.e.* as a set of connections, devoid of stereochemistry, then we can define the greatest common subgraph of two molecules as follows. It is a graph such that if any two vertices (atoms) i and j have an edge joining them in the subgraph, then there is an edge joining the corresponding vertices i', j' in one molecule and an edge joining the corresponding vertices i'', j'' in the other. Further, there exists no subgraph with this property that has more vertices. The definition of the greatest common fragment, or intersection, of two molecules is the same as that of the greatest common subgraph except that stereochemistry is normally included. If i and j are *cis* in the subgraph then the corresponding atoms i' and j' must be *cis* in one molecule and the corresponding atoms i'' and j'' must be *cis* in the other molecule. Further, if atom i of the intersection has ligand atoms a, b, c, and d, with a, b, and c arranged in clockwise order when viewed along the vector from i to d, then the corresponding ligand atoms of atom i', *i.e.* a', b', and c', will be arranged in clockwise order when viewed along the vector from i' to d'. The ligand atoms of atom i'' in the other molecule will be arranged correspondingly.

Finding the intersection of molecular structures is of course trivially easy for a human brain. The difficulty in this area only arises when we need to put a method into a computer program. This requires an exact prescription, certain to work, *i.e.* an algorithm. Furthermore, the algorithm ideally should be efficient. It can be shown that the time required to find the intersection of two molecules need not increase exponentially or as the factorial of the number of atoms involved. It need only depend on a small power of this number.

There are at least five situations in which it is useful to use a computer to find the intersection of two molecules, as follows.

(1) We are given a long list of molecules, all of which have some interesting common property, such as biological activity, and we would like to find the intersection substructure. The intersection of the first two molecules is used as a substructure and mapped onto subsequent molecules to find the next intersection. With each successive mapping, the common fragment can only remain constant or decrease in size.

(2) We are given a molecule to be synthesized and a list of molecules which, *inter alia*, may serve as starting points for a synthesis. A reasonable starting material would be that substance which has the most important intersection with the product. 'Most important' may simply mean having the greatest number of atoms, or it may mean having the greatest number of functional groups or steric relationships.

(3) We are given a particular molecule as starting substance for the synthesis of another substance and wish to find those atoms which are superfluous and should be removed in the course of the synthesis. These atoms are simply all those which are present in the starting material but not contained in the intersection of the starting material and the substance to be synthesized.

(4) We are given the product of a reaction in which two sites are required to be equivalent in the reactant, *e.g.* the α-carbon atoms of a ketone. Since we are examining the product, the two α-carbon atoms are no longer equivalent. We then exclude the entering atom that formed a bond to one of the carbon atoms α to the ketone carbonyl carbon. Excluding this entering atom, we can now find out if the resulting fragment can be mapped onto itself in such a way that the two α-sites are equivalent (within one hydrogen atom). In order words we find the intersection between one half of the fragment and the other. This is by far more rapid than actually forming the connection table of the reactant ketone, canonicalizing it and seeing whether the two α-carbon atoms are equivalent.

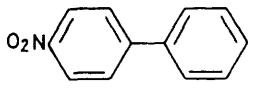
(5) We are given product(s) and reactant(s) and wish to find the unchanged moiety, so as to deduce the reaction centre. The unchanged moiety is precisely the intersection of the product and the reactant.[1] The remaining atoms are involved in the reaction.

Preliminary Definitions.—In almost all this discussion, the word atom means an atom other than hydrogen.

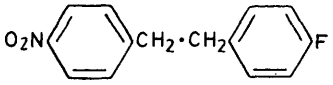
By terminal atoms we mean atoms which have only one non-hydrogen neighbour, such as methyl carbon, hydroxylic oxygen, carbonyl oxygen, amino nitrogen, halogens in their usual monovalent state, terminal methylene, *etc.*

Intersections are of two kinds, connected and unconnected. For connected intersections, a path can be found between any two atoms. If the intersections are unconnected there are at least two distinct pieces. Con-

sider, for example, 4-nitro-4'-fluorobiphenyl (1). This has a significant intersection with 1-(4-nitrophenyl)-2-(4-fluorophenyl)ethane (2). If we consider only the connected intersection then we decide that the intersection is the 4-nitrophenyl group. On the other hand the unconnected intersection consists of the two benzene rings and their heteroatom substituents. This raises the



(1)



(2)

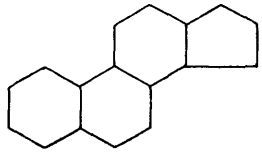
question of the minimum size of each piece. If we place no restriction on the minimum size of each matching piece then in the extreme any pair of isomers, containing n atoms, has a trivial unconnected intersection consisting of n separate atoms. It is precise and convenient therefore to speak of an unconnected intersection of minimum order k, where k is the smallest number of atoms acceptable in any one of the pieces. In the above example, the intersection consisting of the two separate benzene rings and their heteroatom substituents is an intersection of minimum order 7; the 4-fluorophenyl part contains 7 atoms.

In the above we have used the idea of the correspondence of atom i in one molecule and atom j in another without stating how this correspondence is determined. In the strictest and most usual sense, *i.e.* the exact match, atoms of different molecules ' match ' only if they have the same number of attached hydrogen atoms, the same degree of unsaturation, the same atomic number and the same chirality. Furthermore, if i is *cis* or *trans* to an atom j then i' must be *cis* or *trans*, respectively, to an atom j' which has the same atomic number, number of attached hydrogen atoms, degree of unsaturation, and chirality as atom j. The degree of unsaturation is specified as saturated, aromatic, doubly bonded, twice doubly bonded, as in an allene or ketene, or triply bonded.

It is also possible to use a skeletal match, *i.e.* to define as matching any two atoms which have the same atomic number and the same number of cyclic neighbours. Using the skeletal match we obtained the familiar tetracyclic skeleton (3) as the intersection of all steroids.

We can also use a functional match, *i.e.* we can define as matching any two atoms which have the same atomic number and which are either both functionalized or both non-functionalized. A functionalized atom is unsaturated and/or has a heteroatom neighbour. Using the functional match, the three carbon atoms of 2-chloropropane correspond to the three carbon atoms of acetone. A host of other matches can be defined. For synthesis purposes we may want to permit functionalized atoms to match non-functionalized atoms if the functionality is a carbon–carbon multiple bond or is allylic or benzylic.

Some atoms are peripheral to a molecular intersection. If an atom a in A is not part of the intersection of A and B, but its neighbour c is part of this intersection, then, for some purposes, we may wish to include atom a in the intersection, after the true intersection has finally



(3)

been determined. A requirement for this is that there must be an atom b which partly matches a and which is a neighbour of the atom d in B which matches atom c. As an example, consider the intersection of toluene and ethylbenzene. The benzylic carbons do not agree in the number of hydrogen atoms hence the exact intersection consists of the two phenyl groups. However, if we wish to modify the definition of intersection slightly, we can include, in the final stage, such peripheral atoms, which agree in atomic number but not in all other respects, *e.g.* the degree of unsaturation, the chirality, and the number of attached hydrogen atoms. Thus, using this new definition, the intersection now consists of the group Ph–C. Similarly the intersection of acetaldehyde and acetone can be either an acetyl group or a methyl group depending on whether or not we include the peripheral atom. We will refer to peripheral atoms which match with their corresponding peripheral atoms in the other molecule in their atomic number but not in all other respects, as semi-matching peripheral atoms.

We will assume in all this discussion that the atoms of the molecules concerned have been numbered by a canonical procedure and that in the process of canonicalizing the structure representation the atoms have been partitioned into equivalent classes. Unique atoms are in equivalence classes consisting of only one member. If there is for example one t-butyl group in the molecule and the methyl carbon atoms are numbered 26, 27, and 28 then atom 26 is the ' first ' member of its equivalence class. In mapping problems we will often be concerned only with the first member of an equivalence class.

Finally, we define what is meant by an atom central to an intersection. It is the atom of the intersection which is furthest from any non-matching atoms. By furthest, we mean separated by the greatest number of bonds, not separated by the longest distance in space. Consider for example the intersection of cyclohexyl bromide and cyclohexanone. This intersection consists of a string of five methylene carbon atoms. The centre of the intersection is C(4), since the non-matching atoms are three

atoms away, whereas there are no other atoms in the intersection which are three atoms away from the non-matching atoms, *i.e.* C(1).

The Direct Algorithm.—Since there are many conceivable atomic matches, there are in principle many different kinds of intersections and therefore many different algorithms for finding them. However, we can simplify matters. Let us first suppose that we are seeking a connected intersection, that is, the result will be only one fragment. Further, let us ignore the detailed content of the atomic match, *i.e.* whether it is exact, skeletal, functional, *etc.* Then from this viewpoint we can distinguish two approaches, the direct method and a method to be described herein which requires a preliminary search for the centre of the intersection.

The direct method begins by considering each of the *m* atoms of molecule A in turn and determining with which of the *n* atoms of molecule B there is a match. In this way we obtain a list of ordered pairs (a, b) in which a is an atom of A and b is an atom of molecule B. The correspondence can be many-to-many. For example atom 1 of A can match atoms 2, 6, and 8 of B and atom 2 of B can match atoms 1 and 5 of A.

For each of the ordered pairs (a, b) on this list we try to obtain an ordered quartet (a, b, c, d) such that a and c are neighbouring atoms of A, b, and d are neighbouring atoms of B, a matches b, and c matches d. If there is any ordered pair (a, b) for which we cannot find such a quartet then we strike it from our list of potential intersections. For example the amino-group of a primary amide would match that of a primary amine but a matching two-atom fragment starting with the amino nitrogen cannot be found.

In any practical implementation of this algorithm we will want to eliminate duplicates, such as (a, b, c, d) and (c, d, a, b). Also if the molecule has any symmetry, *e.g.* a and e are equivalent, then we should eliminate, at the beginning, all ordered pairs whose left member is e. Further, if b and x are equivalent atoms in B then we will want to rule out the inclusion of both (a, b) and (a, x) in the list of matching pairs.

Again for each of the ordered quartets (a, b, c, d) we try to obtain an ordered sextet (a, b, c, d, e, f) such that a and c are neighbours in A, c and e are neighbours in A, b and d are neighbours in B, d and f are neighbours in C, a matches b, c matches d, and e matches f. We continue in this way, finding ever larger multiplets until we can no longer do so. The longest multiplets thus obtained are the desired intersections.

The direct method of finding molecular intersections is quite expensive with regard to computer time. Much effort is spent in exploring match possibilities which subsequently fail, there is no looking ahead, and we are constantly required to check whether an atom being considered is or is not already in our multiplet. The direct method has been implemented in various programs, for example in that of Varkony *et al.*[2]

The Algorithm with a Preliminary Search for the Centre of the Intersection.—The method with a preliminary search attains the look-ahead capability by seeking, at first, not the details of which atoms in a fragment of molecule A match which atoms of a fragment of molecule B. Instead, what is sought is the identitity of an atom in A which is the centre of an intersection, *i.e.* for which we are assured that the neighbours have a corresponding match in B and the neighbours of these neighbours have a corresponding match in B, and so on to the greatest extent possible. When this central atom is found, finding the details of the intersection is simple, without any backtracking being necessary.

The method with a preliminary search begins by building a list of ordered pairs (a, b) for all non-terminal atoms a in A such that there is a matching atom b in B and there is at least one one-to-one matching correspondence between the neighbours of a and the neighbours of b. This means that if the neighbours of atom a are i, j, k, and l and the neighbours of b are w, x, y, and z then i matches w, j matches x, k matches y, and l matches z. It may also be true that i matches x and j matches w but this is of no consequence. We are assuming, of course, that the connection tables representing the structures of A and B have been canonicalized according to the same algorithm, so that i is the neighbour of atom a which ranks highest according to the canonicalizing algorithm, and w is the neighbour of b which ranks highest according to the same algorithm and so forth for the remaining neighbours. We also take advantage of any symmetry, *i.e.* if there is a set of equivalent atoms in A then only one of the set may appear as a left member of any ordered pair on the list. Similarly only one of a set of equivalent atoms in B may appear as a right member of any order pair on the list. The question of equivalence of atoms must be decided by a previous subprogram which has canonicalized the connection tables and decided which atoms of each molecule are equivalent to each other.

If every atom in both molecules appears on the list then we must use an alternative method for finding the centre of the intersection. This rare situation will be discussed later.

Let us call this initial list List 1. We next build List 2. List 2 consists of all those ordered pairs (a, b) which are on List 1 but which have the additional property that the neighbours of atom a also appear as left members of ordered pairs on List 1 and the neighbours of atoms b also appear as right members of ordered pairs on List 1. If an atom has two or three equivalent neighbours then it is sufficient that one of these neighbours appears on the list. By the restriction in the previous paragraph, it is impossible for two equivalent neighbours both to be on the list.

Each ordered pair in List 1 actually signifies matching fragments of at least three atoms. Since a is non-terminal it has two neighbours and each of these has a separate match among the neighbours of b. Each ordered pair in List 2 signifies matching fragments of at least five atoms. At the middle of these five or more atoms in A is the left member of the ordered pair (a, b). The atom b

is at the middle of a matching fragment of five or more atoms in molecule B.

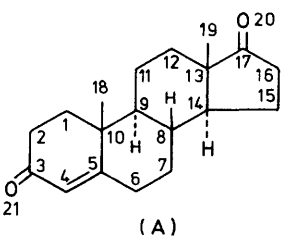
If List 2 has more than one ordered pair then we build List 3 as follows. An ordered pair (a, b) can be on List 3 if it is on List 2 and if the neighbours of atom a are also left members of ordered pairs on List 2 and the neighbours of atom b are also right members of ordered pairs on List 2. If List 3 has more than one ordered pair we build List 4 from List 1 in the same fashion. Each new list signifies matching fragments which are larger by at least two atoms than the previously detected matching fragments. Also each new list has many fewer members than the preceding list. This contrasts with the direct method, in which the space of partially matching fragments explored tends to expand during the first one or two iterations. Soon we arrive at a list which is null or has one ordered pair. The last list before the null list contains ordered pairs each of which is the centre of a separate but equal-sized molecular intersection. If the final list has just one pair, (a, b), then atom a (and atom b in its molecule) is the centre of a single intersection substructure.

Taking the example of finding the intersection of toluene and ethylbenzene, we see that the first list consists of the five ring carbon atoms each of which bears a hydrogen atom. If we use the exclusions of symmetry in molecule B, the first list consists of nine pairs, namely (2, 2′), (2, 3′), (2, 4′), (3,2′), (3, 3′), (3,4′), (4,2′), (4, 3′), and (4, 4′). The numbers for the atoms are the standard IUPAC numbers for the ring atoms. The neighbours of atoms 2 in toluene do not all appears as left members of ordered pairs on this list. Atom 1 is absent. Hence, when we build the next list, all ordered pairs beginning with 2 are deleted. Similarly, atom 2 in ethylbenzene has a neighbour, atom 1, which does not appear as a right member on the list; hence all ordered pairs ending with 2′ are deleted in building the second list. As a result the second list consists only of the *meta* and *para* carbon atoms, *i.e.* the four pairs (3, 3′), (3, 4′), (4, 3′), and (4, 4′). Evidently atom 3 has neighbour 2 which does not appear on this list and atom 3′ has neighbour 2′ which does not appear on this list. Therefore in building the next list, we must delete pairs which have left member 3 or right member 3′. This leaves only the pair (4, 4′). Atom 4 has a neighbour 3 which is on the list and another neighbour 5 which is equivalent to 3; hence atom 4 is acceptable for inclusion on the next list, and so, similarly, is 4′. We conclude therefore with the short list (4, 4′) of the *para*-atoms. When a list contains only one ordered pair then we do not have to attempt to build the next list since the latter must be null.

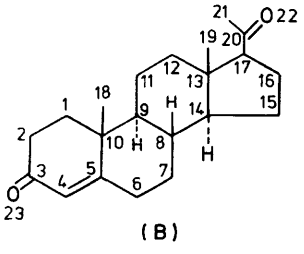
Having determined one or more atoms each of which is the centre of a molecular intersection, we proceed to generate the list of the corresponding atoms in the intersection. We begin this list with the centre and add neighbours of atoms of A on the list together with their matching neighbours of the corresponding atoms in B. Thus, from (a, b) we proceed to (a, b), (i, w), (j, x), (k, y), (l, z). We then examine i, j, k, and l in order to find new neighbours of theirs which have matching atoms among the neighbours of atoms w, x, y, and z in B. We continue in this way until all the atoms of A in the intersection have had their neighbours inspected for matches with the neighbours of the corresponding atoms in B. When this is so we are finished and the intersection is found. There may be more than one intersection found, but the intersections will be of the same size. Since each atom of A can appear no more than once as a left member of an ordered pair in the list, and each atom of B may also appear no more than once as a right member of an ordered pair on the list, in programming the algorithm a bit array or its equivalent must be established which tells whether or not an atom has been placed on the list previously.

In the example of the intersection of toluene and ethylbenzene, we begin with the *para* carbon atoms, then add the *meta* carbon atoms, then the *ortho* carbon atoms, and finally the remaining aromatic carbon atom. Note that the aromatic atoms C(1) match and are included in the intersection, despite the fact that they were not put on the first list, since their saturated carbon neighbours do not match.

Let us now consider a more lengthy problem, that of finding the intersection of the two steroids, (A) and (B).



(A)



(B)

In this case, using the numbers shown for the acyclic atoms and the standard IUPAC steroid numbering for the cyclic atoms, we find List 1 to be: 1 : 1, 2 : 2, 3 : 3, 4 : 4, 5 : 5, 6 : 6, 10 : 10, 11 : 11, 11 : 15, 12 : 12, 15 : 11, 15 : 15, 16 : 2. We use a colon in each pair to mean ' corresponds to '. The left member of each pair is the number of an atom in molecule A and the right member is the number of a corresponding atom in molecule B. Note that the pair 7 : 7 is not on the list because atom 7 in A is adjacent to an atom of *R* chirality and atom 7 in B is adjacent to an atom of *S* chirality.

List 2 in this example is 1 : 1, 2 : 2, 4 : 4, 5 : 5, 6 : 6, 7 : 7. List 3 is null.

The last non-null list is thus List 2. We begin by selecting the first pair on this list, 1 : 1, as the centre of the intersection, the beginning point for building up the list of corresponding atoms. Our survey of the neighbours of atom 1 in A and the corresponding atom 1 in B gives us 1 : 1, 2 : 2, 10 : 10. Collecting the matching neighbours of these atoms gives 1 : 1, 2 : 2, 3 : 3, 5 : 5, 9 : 9, 10 : 10, 18 : 18. One more iteration provides 1 : 1, 2 : 2, 3 : 3, 4 : 4, 5 : 5, 6 : 6, 9 : 9, 10 : 10, 11 : 11, 18 : 18. We continue to expand in this fashion until we reach the list 1 : 1, 2 : 2, 3 : 3, 4 : 4, 5 : 5, 6 : 6, 7 : 7, 9 : 9, 10 : 10,

11 : 11, 12 : 12, 13 : 13, 14 : 14, 15 : 15, 16 : 16, 18 : 18, 19 : 19. No atom on this list has any neighbour not on the list which will match the corresponding neighbour of the corresponding atom in the other molecule. Hence we have come to the end. If we add peripheral atoms which have the same atomic number then we introduce the pairs 8 : 8 and 17 : 17 before exiting. The other pairs of List 2, *i.e.* 2 : 2, 4 : 4, 6 : 6, and 7 : 7, are all present in this intersection, hence there is no need to use them as starting points. There is no other equal intersection.

There is a resemblance between this method of finding the intersection of molecules and the Morgan method of numbering the atoms of molecules.[3] Morgan's method and all methods which use extended properties essentially avoid a detailed examination of remote situations.[4-6] It is enough to know that properties of remote atoms are reflected in certain sums which are accumulated for each atom *via* the previous sums for its neighbours. Suppose, for example, after $k - 1$ iterations of a sum algorithm, atoms X and Y of a molecule still have identical sum values. On the kth iteration the sum for X becomes greater than that for atom Y. It is enough to know that somewhere k atoms away from atom X the atoms have higher ranking in atomic properties than the set of atoms which are k atoms away from atom Y. It is unnecessary to trace the exact path. We are assured that then X outranks Y and should precede it in the connection table. Similarly, in the present case it is enough to know that if we have ordered pairs (a, b) and (m, n), and after k iterations we can include (a, b) on a list but must exclude (m, n) because not all the neighbours of m are on the previous list, then at a distance of k atoms out on the tree emanating from a the atoms all match corresponding atoms on a tree emanating from b in B, whereas somehow not all the atoms on the same size tree emanating from m in A match a corresponding atom on a tree emanating from n in B. This decides that (a, b) is included in the new list and (m, n) is not. Again, it is unnecessary to know where in the environment of m this mismatch occurs.

This algorithm assumes two connected fragments. The extension to non-connected fragments is straightforward. We examine lists of equivalent pairs, both the final list and the one prior to it, seeking atoms which are not part of the first intersection. As soon as we have found one, we know that we have the centre of a separate matching fragment. This separate fragment is assembled as before, growing out from the centre, and if it has the minimum requisite number of atoms the fragment is added to the original intersection as part of the larger, unconnected intersection.

We note that by implicitly demanding at the beginning of the preliminary search that the atoms of List 1 must be the centres of matching fragments at least three atoms in size, we preclude the possibility of discovering any two-atom intersection. This is a small loss as not many applications would require intersections as small as two atoms. If we were interested in terminal two-atom intersections, such as FC–, then we could relax the restriction that the first atoms to be put on List 1 must be terminal. In such a case, since the carbon atom adjacent to the fluorine atom has a match in B the fluorine atom can be considered as a potential centre of an intersection. If we were finding the intersection of ethyl fluoride and fluoroacetone, List 1 would consist only of the fluorine atoms. The matching neighbour would be the methylene carbon atoms. Similarly, if we were looking for the intersection of styrene and propene, List 1 would consist only of the terminal methylene, and the intersection is CH_2=CH–. Even with this change in the algorithm we still could not find two-atom intersections in which both atoms are non-terminal, such as –CH=CH– or –Si–C–. We can find all two-atom intersections if we change the requirement for inclusion on the initial List 1 further and admit any matching pairs, regardless of the matching of the neighbours. This would be a slower algorithm, not necessary for most applications, but still much faster than the direct method.

In the application (3) (above), where the synthetic significance of the intersection is more important than sheer size, we must modify the basic algorithm described here. The simplest way to do this is to exclude from the goal molecule structure those parts which lack stereochemical or functionality features. Then we are considering the intersection of various possible starting materials with the crucial parts of the goal structure.

Implementation of the Algorithm.—The algorithm has been programmed in PL/I. It consists of 220 or so PL/I statements. The input is two canonicalized structure representations and the output is one or more lists of ordered pairs of atom numbers. The basic match is exact. Any semi-matching peripheral atoms are included after the exact intersection has been found. As a programming detail we note that only two lists are kept since the contents of List 3 are stored in the place reserved for List 1, the contents of List 4, if any, are stored in the place in memory formerly occupied by the contents of List 2, and so forth.

The time required for finding the intersection of toluene and ethylbenzene on an IBM 3033 computer was 2.9 ms. On the same computer the two intersections of cyclopentanone and progesterone were found in 4.0 ms. For the direct match, using a previous program in assembly language, the time required to find such intersections was of the order of several hundred ms. A copy of the PL/I program is available from the author.

Handling the Exceptional Case.—In a few cases every cluster of an atom and its neighbours in either molecule corresponds exactly to at least one atom and its neighbours in the other molecule. Consequently every atom in both molecules appears on the initial version of List 1. This happens with molecules of high symmetry that are homologously related and with certain isomers. An example is the pair cyclopentyl bromide and cyclohexyl bromide. Another such example is the intersection of 1,4-dimethylnaphthalene and 1,5-dimethylnaphthalene. Since in this situation an attempt to construct the next

list would result in a copy of the first list, at this point a computer program must check that not all atoms of both molecules appear on the list. If all the atoms of both molecules appear on the list then an exceptional method must be called in to find the centre of the intersection.

The exceptional method is based on a sum algorithm for canonically numbering the rows of a molecular connection table, *e.g.* that of the present author. The sum algorithms are used among other things for determining which atoms are equivalent to one another inside a molecule. (Uchino [7] has pointed out examples for which Morgan's sum algorithm [3] does not show which atoms are equivalent. For all his cited examples a sum algorithm which includes ring sizes as part of the atomic properties always partitions the atoms correctly into equivalence classes. Morgan's atomic property consisted solely of the number of nearest neighbours.) The sum algorithm terminates when the number of atoms which have a corresponding equivalent atom in the molecule is zero or does not decrease between the nth and the $(n + 1)$th iteration of the summing process. The nth sum for an atom includes information about atoms which are n bonds away; the $(n + 1)$th sum includes information about atoms which are $n + 1$ bonds away, *etc.* Now if we compare the sums for corresponding atoms (a, b) of two different molecules, then if the sums become different only on the $(n + 1)$th iteration this implies that the environments n atoms away for atom a and for atom b are indistinguishable. If after the $(n + 1)$th iteration of summing there are no more pairs of atoms (a, b) which gave the same sum values and which have consistently had the same sum values after each iteration, then the pair (a, b) which were indistinguishable after the nth iteration are precisely the centres of the intersection of the two molecules. We may add parenthetically that the intersection will consist of at least $2n$ atoms and the longest path in it will consist of no more than $2n + 1$ atoms.

Appendix

A Detailed Description of the Basic Algorithm.—The program receives as input the canonicalized structure representations of molecules A and B. The output of the program is a correspondence list of the form a : a′, b : b′, c : c′, *etc.* where the unprimed symbols are atom numbers of molecule A and the primed symbols those of molecule B. The algol symbol := means ' receives the value of '. Thus $I := 0$ means I gets the value zero. $I := I + 1$ means I gets the value of the previous value of I plus one, in other words I is incremented by one. A step-by-step description of the basic algorithm follows.

(A) Check for identity of molecules A and B. If A and B are the same then exit, reporting the correspondence list $1 : 1, 2 : 2, \ldots, n : n$.

(B) Build List 1.

(B1) $I := 0$

(B2) $I := I + 1$

(B3) If the value of I now exceeds the number of atoms in molecule A then exit from part B and proceed to step (C1).

(B4) If atom I is not the first member of its equivalence class in molecule A then go back to step (B2).

(B5) $J := 0$

(B6) $J := J + 1$

(B7) If the value of J exceeds the number of atoms in molecule B then go back to (B2).

(B8) If atom J is not the first member of its equivalence class in molecule B then go back to (B6).

(B9) If atom I does not match atom J then go back to (B6).

(B10) if the first neighbour of atom I does not match the first neighbour of atom J then go back to (B6).

(B11) If the second neighbour of atom I does not match the second neighbour of atom J then go back to (B6).

(B12) If the third neighbour of atom I exists and does not match the third neighbour of atom J then go back to (B6).

(B13) If the fourth neighbour of atom I exists and does not match the fourth neighbour of atom J then go back to step (B6).

(B14) Add the pair $I : J$ to List 1 and then go back to (B6).

(C) Prune the previous list, producing the current list.

(C1) If List 1 is null, then exit from the program, reporting a null correspondence list, *i.e.* no intersection.

(C2) $Q := 0$

(C3) $Q := Q + 1$

(C4) If the value of Q exceeds the number of pairs on the previous list then advance to step (C9).

(C5) Let $I : J$ be the qth pair of the previous list. If any neighbour of I does not appear as a left member of a pair on the previous list then go back to (C2).

(C6) If any neighbour of J does not appear as a right member of a pair on the previous list then go back to (C2).

(C7) Add the pair $I : J$ to the current list.

(C8) Go back to (C2).

(C9) If the current list is not null then give the previous list the value of the current list, then make the current list null, and then return to step (C1).

(C10) Since the current list is null, use the previous list as the current list. With this done, any pair on the current list is the centre of an intersection.

(D) Construct the final correspondence list(s).

(D1) $L := 0$

(D2) $L := L + 1$

(D3) If the value of L exceeds the number of centres of the intersection of the current list produced by part B then exit from the program. We have finished. The possible intersections are all listed.

(D4) Start a new list called List n, the first pair of which is the Lth pair on the ' current list ' produced by part (C).

(D5) Let this Lth pair be $I : J$. If the pair $I : J$ appears on a previous intersection then go back to (D2).

(D6) Let $I : J$ be the first pair on List n. Further suppose V is the first neighbour of I and V' is the first neighbour of J. If V matches V' and V has never been on List n then add the pair $V : V'$ to the end of List n.

(D7), (D8), (D9). Do the same as (D11) for the second, third, and fourth neighbours of I.

(D10) Delete $I : J$ from List n and add it to the correspondence list of the current intersection.

(D11) If List n is not null then go back to (D8).

(D12) List n is null and we have finished the correspondence list of an intersection. Go back to (D2).

This research was supported by a grant from the National Sciences and Engineering Council of Canada.

[1/828 *Received*, 27*th May*, 1981]

REFERENCES

[1] M. Bersohn and K. MacKay, *J. Chem. Inf. Comput. Sci.*, 1979, **19**, 137.
[2] T. H. Varkony, Y. Shiloach, and D. H. Smith, *J. Chem. Inf. Comput. Sci.*, 1979, **19**, 104.
[3] H. L. Morgan, *J. Chem. Doc.*, 1965, **5**, 107.
[4] M. Bersohn, *Comput. Chem.*, 1978, **2**, 113.
[5] W. T. Wipke and T. M. Dyott, *J. Am. Chem. Soc.*, 1974, **96**, 4834.
[6] P. Willett, *J. Chem. Inf. Comput. Sci.*, 1979, **19**, 159.
[7] M. Uchino, *J. Chem. Inf. Comput. Sci.*, 1980, **20**, 116.