Converting Chemical Formulas to Names: An Expert Strategy

ARTHUR A. EGGERT,* ANTHONY T. JACOB, and CATHERINE H. MIDDLECAMP
Chemistry Learning Center, University of Wisconsin—Madison, Madison, Wisconsin 53706

Received January 7, 1992

The ability to name inorganic chemicals, given their chemical formulas, is essential not only to chemists and students of chemistry but also to computers if they are going to assume the role of intelligent assistants in the chemical laboratory. To study the problems in this formula-to-name conversion, we have developed a model in the form of a PC-based expert system that maps formulas into chemical names. The model strives to precisely define and circumscribe the domain of commonly encountered inorganic chemicals and to handle both systematic and common names. This rule-based approach also provides a trace of its actions in arriving at a name to permit the expert to be used in computer-aided instruction.

INTRODUCTION

As computers are increasingly used by chemists, computer software must become progressively more sophisticated in handling chemical nomenclature and terminology. Programs must move beyond the capabilities of spreadsheets and databases which merely regurgitate what is put into them with specified modifications, and they must be more versatile than the pieces of code that do specific calculations. In short, the software must "think" in chemical terms.

A logical point in which to increase the chemical intelligence of computer programs is with their ability to handle the chemicals themselves, i.e., to work with chemical names and formulas in an intelligent manner. Computers can obviously store both names and formulas, but our requirements must be more stringent than this. Given a chemical formula (or name), the computer system should automatically be able to determine the chemical name (or formula) without having been told it. It should be able to break a chemical down into its parts, knowing, for instance, that $NaNO_3$ contains both a species called "sodium" and a species called "nitrate". Furthermore, it should differentiate between improbable and probable compounds, such as recognizing that potassium diiodide is not chemically feasible, but that potassium iodide and triiodides are. Such an ability to work intelligently with chemical language is essential if we are to proceed with more complex and challenging chemical problems. Otherwise, we will be faced with the situation where computers will continue to program us (i.e., regiment the ways in which we input chemical species) as much as we program them.

Our interest in the problem of chemical nomenclature and formula writing stems from our work in chemical education. During the last 5 years we have been developing an intelligent tutor for general chemistry problem solving called CHEMPROF.[1-3] Much of the project involves various facets of the problem of adapting computer technology to tutoring and adapting chemistry teaching theory to be used by a computer. A major task that must be accomplished, therefore, is the codification of chemical knowledge in a manner applicable to automated problem solving. Such problem solving is done by programs called "topic experts" (or just "experts") which supply to the teaching logic (software generally knowledgeable about teaching but not about the particular topic) the information necessary to allow it to guide the student in solving the specific problem given. Certainly, converting a chemical formula to a chemical name is an example of a problem that a student must solve; in fact, it is a representational problem that must frequently be solved before work on the problem of chemical interest can be attempted.

The formula–name interconversion is, of course, a bidirectional process. We have initially chosen to look at the formula-to-name problem, i.e., given the chemical formula of an inorganic compound, to generate its chemical name. This problem is composed of two parts. First, it must be determined whether the formula supplied by the user is chemically feasible. If so, the formula still may not be the chemical the user intended, but at least the expert should not try to name gibberish. Second, the formula must be decomposed into nameable subparts, and the name fragments recomposed into a chemically correct name. Each feasible chemical formula has one or more names to which it can be converted (the formula-to-name mapping is not unique in common usage, despite standards by the International Union of Pure and Applied Chemistry, IUPAC[4]), and an expert program must be able to produce at least the preferred IUPAC name.

Fundamentally the twin problems of recognizing whether a certain formula is chemically feasible and of then determining its correct chemical name require a mixture of chemical knowledge and artificial intelligence technology. It is a subset of the general problem of human language understanding, a problem that has received a lot of computer science investigation.[5-7] It is unique in that it requires "real world" knowledge that exists only in the field of chemistry and which is more complex than most informational domains in which language understanding has been attempted. The underlying periodic properties of the elements, coupled with human inconsistencies in developing naming conventions, complicate the task of successfully parsing and interpreting the input.

SCOPE OF THE PROBLEM

Though naming inorganic formulas is a straightforward task for the experienced chemist, it is far from easy to develop a comprehensive system to do it automatically.[8] Difficulties arise because there are different naming rules for different classes of compounds, and because each different set of naming rules has its own intricacies which must be understood.[9,10] It is further complicated by the facts that different texts use different naming strategies and that there are some inconsistencies even within the standard IUPAC rules. To reduce the scope of the formula-to-name problem to a manageable level, the set of inorganic chemicals was confined to those that were likely to be encountered in general chemistry courses: elements, ions, and ionic, molecular, and coordination compounds. Even in this domain, naming chemical formulas is a difficult task as illustrated in the following set of examples.

Consider the following two-element chemicals, each of the form AB_3: NH_3, NCl_3, NI_3, AlI_3, FeI_3, and NaI_3. Each of these is named using a different chain of reasoning. The first, NH_3, is known only by its "common name", ammonia. Common names do not lend themselves to a systematic set of rules and must simply be memorized. The second compound, NCl_3, must first be recognized as a molecular compound and then be named with the appropriate rules. In binary molecular compounds, the less electronegative element is written first in formulas and names, and it is followed by the second ele-

0095-2338/92/1632-0227$03.00/0

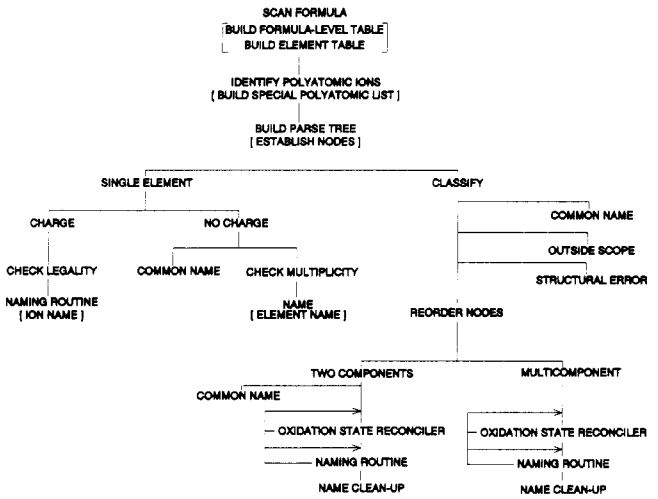

Figure 1. This is the top-level flow chart of the formula-to-name expert, showing the steps it must traverse to successfully assign a name. Steps where recursion may occur are indicated by arrowheads.

ment named with the suffix "ide". The number of atoms present is identified with the prefixes mono-, di-, tri-, etc., except when the first element has only one atom present. From these rules, NCl_3 is named nitrogen trichloride. The next compound, NI_3, has a similar name even though the naming scheme is incorrectly used. Since nitrogen is more electronegative than iodine, the name and formula should be triiodine nitride and I_3N, respectively. Because it is treated as an analogue of NCl_3, however, it is instead written in the reverse order as nitrogen triiodide. The system described thus far can be employed only for inorganic molecular compounds; a new set of naming rules is required for ionic compounds.

Since AlI_3, FeI_3 and NaI_3 are recognized as ionic compounds, they are named according to a different set of rules. AlI_3 is not named aluminum triiodide, but rather aluminum iodide. This occurs because aluminum can have only one oxidation state, +3, and can therefore form only one compound with iodine. A "tri" prefix on iodide would be redundant and is therefore neither needed nor permitted. When naming FeI_3, however, the name "iron iodide" is inadequate because iron can exist in two common oxidation states: +2 or +3. Further, the name iron triiodide is also incorrect since prefixes are applied only to molecular, not ionic, compounds. The preferred name includes the oxidation state of the metal (also called the Stock number), followed by the second element with an "ide" suffix; that is, iron(III) iodide.[11] Prior to this naming scheme, chemists called this compound ferric iodide, with the "ic" suffix on the metal indicating the higher oxidation state of iron. Any computer that understands chemical representation must therefore also accept this older name, since it is still frequently used, but must show a preference for the IUPAC name. Finally, NaI_3 is named sodium triiodide even though it is an ionic compound, not a molecular one. This apparent anomaly occurs because the three iodine atoms are treated as a polyatomic ionic group, I_3^-, named "triiodide".

Clearly, when more than two elements are present, the number of organizational degrees of freedom among the atoms in the compound increases, making the naming task even harder. To correctly name all inorganic compounds is therefore an open-ended problem. The domain of chemicals we have selected, however, will be adequate for most chemists and undergraduate students under most circumstances and has reduced the complexity and minimized the errors that could occur upon its application.

FORMULA-TO-NAME EXPERT

To systematize the process of converting a formula to a name, the formula must be separated into component parts that make chemical sense. When one looks at the variety of possible initial operations in crafting a name from a formula, ranging from directly assigning a common name to drawing the structural arrangement to find the relationship between various components, such a decomposition may appear unlikely. We have attempted this procedure using a structure called a parse tree. After the parse tree is built, it is evaluated as a whole, its nodes are reordered, the oxidation states are recursively assigned to the nodes, and finally, after returning to the root node, names are assigned to the individual nodes (Figure 1). We now examine these steps in more detail.

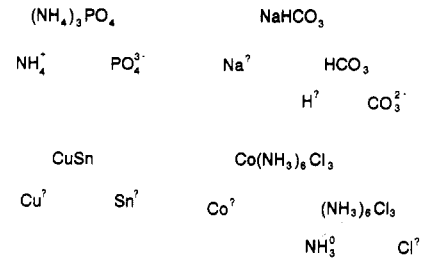

Figure 2. Four parse trees shown here are typical of those generated by the formula-to-name expert. Note that some components have their charges assigned when the tree is built, but those which can have more than one oxidation state do not.

Table I. Attributes of Chemicals Needed for Classification

no. of elements
no. of polyatomic groups
no. of invariant metals
no. of nonmetals
no. of inert elements
no. of carbon atoms
no. of hydrogen atoms
no. of species with more than 1 atom present
no. of total atoms
charge
no. of variant metals
no. of metalloids
no. of halides
no. of oxygen atoms
binary hydrogen flag

Table II. Information Stored for Each Node

formula fragment	assignment priority
name of element/group	no. of element/group present
oxidation number	pointers to group list
type (metal, metalloid, etc.)	pointers to daughter nodes

Parse Tree. A parse tree is a way of organizing information in which each item is a component of the item above it. The original item to be parsed is placed at the top of the inverted tree, which is called the "root" node, and each fragment broken from it is placed at a "daughter" node. Each daughter node can also have daughters. If it does not, it is called a "leaf" node. Several examples of parse trees are shown in Figure 2.

Information associated with the parse tree is usefully stored in three different ways. Information abstracted from the formula that relates to the tree as a whole is stored in a formula-level table (see Table I). It contains attributes which are primarily about the amounts and types of elements present and other features that cause changes in the naming conventions. (For example, the presence of a metal with multiple oxidation states causes the inclusion of the oxidation state of that metal in the final answer.) The second way to store information is at the node level (Table II). This is where most of the manipulation in the naming process is undertaken, and where the information is assembled before the final determination of the name. The final way of storing information is in a table of elements arranged in the order in which they

Table III. Attributes Stored for Each Occurrence of Each Element

symbol	valence
atomic number	type (metal, metalloid, etc.)
no. present	no. of times seen (frequency)

are encountered in a left-to-right scan of the chemical (Table III). The chemical symbols and the number of atoms present are later used by the naming subroutine and in determining whether the chemical has been properly entered.

To start the construction of the parse tree, any charge present is stripped off the chemical formula, and the latter is examined for gross errors (e.g., unmatched parentheses). After scanning the formula, a table of element-level characteristics is built (Table III), and at the same time, a count is made to establish the various formula-level attributes (Table I). The whole formula is placed at the root node of the parse tree with the hypothesis that it is a polyatomic group, i.e., a polyatomic ion or neutral molecule that is manipulated as a chemical entity. This group is then checked against a master list of polyatomic groups known to CHEMPROF. If the process is unsuccessful, it is repeated with a new hypothesis: all but the last element or group marked by parentheses is now considered the polyatomic group and analyzed. If this further attempt succeeds, a left branch is created for the group and a right branch is created for the remainder of the formula. If this second test fails, another element or group in parentheses is removed and another attempt made; this process of removing groups continues until the hypothesis is proven true or until only a single element remains. By this process, groups isolated with parentheses [e.g., NO_3^- in $Fe(NO_3)_3$] and those not isolated (e.g., CO_3^{2-} in H_2CO_3) can be found.

The process of matching the whole or part of the formula to a known polyatomic group is not straightforward. The elements within the group may be written together or separated (e.g., CH_3COO^- instead of $C_2H_3O_2^-$ for acetate). There may also be a substitution of one element for another (for instance, sulfur for oxygen in CO_3^{2-} to yield CS_3^{2-}). Some chemical groups can be written in different orders (for example, OCl^- versus ClO^-) while others may not ($O_2PO_2^{3-}$ for PO_4^{3-}). Elements placed in a different order can also signify completely different groups, as in the example of OCN^-, cyanate ion, when compared to CNO^-, isocyanate ion. Finally, two groups with the same number and type of elements in the same order can have different names, depending on the charge associated with the group as in the case of NO_2^-, nitrite, or NO_2^+, nitryl. All of this is considered by the group analyzer, and if the analysis is successful, the result is recorded at the node and the processing backtracks to handle any part of the formula not yet completed. While this type of formula fragmentation is slower than simply trying to match the pattern of a known group somewhere in the formula and then examining the remainder once the group has been accounted for, it is essential to prevent falsely combining elements to form groups. For example, if the formula HOCl were analyzed in a left-to-right manner, the formula would be split into OH^- and Cl nodes. The subsequent routine that reconciles the oxidation states of the groups would accept this grouping by assigning a +1 charge to the chlorine. This in turn would cause the naming routine to name the compound chlorine hydroxide, instead of the correct name, hypochlorous acid. In a similar fashion, a right-to-left analysis of KSCN would create CN^-, S, and K nodes. However, in this case, the oxidation-state reconciler would fail and report an error.

The proper recognition of polyatomic groups is so critical to success in converting formulas into names that a special group table (Table IV) is built along with the parse tree. This is necessary because some groups can have a variety of charges and therefore names. For instance, the species "O_2" in compounds can have a charge of −4 (named "dioxide"), −2 (named "peroxide"), −1 (named "superoxide"), or have no charge when representing the element oxygen. Moreover, pointers to CHEMPROF's master group list do not always suffice because some polyatomic groups, such as the thioarsenate ion, are extensions of the master group list created by the parser using substitution. In the case of thioarsenate, oxygen is substituted for sulfur when the original formula is not found in the master group list. After the parse tree is completed, the groups and elements found are recorded in the expert trace.

Table IV. Attributes Stored for Each Group

group name	pointer to master group list
oxidation number	pointer to next group

Table V. Trace from the Name to Formula Expert

1	I0 P{4O{1{0	input formula
2	I1 2	# of elements present
3	T0 6	chemical type = molecular
4	E0 P	elemental symbol
5	E0 O	elemental symbol
6	I2 +	end of header
7	E1 O	ready to name element
8	E7 O}−}2	element with charge
9	E5 multiatom unit	a component identified
10	E3 −2	charge on species
11	E6 10	# of species.
12	T1 oxide	unprefixed stem
13	T3 10	numeric prefix
14	T2 deca	prefix indicated by similar components
15	E2 decoxide	name of component
16	E1 P	ready to name element
17	E7 P}+}5	element with charge
18	E5 multiatom unit	a component identified
19	E3 5	charge on species
20	E6 4	# of species
21	T1 phosphorus	unprefixed stem
22	T3 4	numeric prefix
23	T2 tetra	prefix indicated by similar components
24	E2 tetraphosphorus	name of component
25	N1 tetraphosphorus	final name piece in order
26	N1 decoxide	final name piece in order
27	N3 tetraphosphorus decoxide	final name of compound
28	ZZ 0	end of trace

Trace. An expert trace is the list of steps that the expert performs to find the solution to the problem that it is given and the arguments (parameters) that it uses to take these steps (Table V). The beginning steps in the trace include the chemical formula, the number of different elements present, and a list of elements and groups required in the final name. Groups that are not completely defined, such as MnO_4 (which can have a charge of −1 or −2), are specially noted for the teaching logic. As more information about the formula is garnered, such as chemical classification (element, ion, ionic compound, etc.) or charge assignment, it is written into the trace. To prevent the trace from becoming too large, it is pruned of any steps which the expert takes down a wrong pathway.

Classification. After completing the parse tree, the information gathered is used to start making decisions. Input formulas are partitioned for classification purposes into those with only one element and those with multiple elements. The one-element case can be analyzed using a binary tree (Figure 1). First, the charge on the chemical formula is examined. If the charge is 0, the formula is then checked as to whether it has a common name (e.g., O_3 is named ozone). If it does not, then the atomic multiplicity (e.g., 1 for Na, 4 for P_4) is checked. In the case where the charge is nonzero, the formula can be passed directly to the naming routine which can handle both simple ions (e.g., I^-) and polyatomic ions (e.g., I_3^-). The final name, albeit a simple one, is then added to the trace. If only one element is present, but the formula does not conform

Table VI. Errors Reported by the Formula-to-Name Expert

0	Expert can't name this. Typing error?
1	A carbon compound that's beyond me.
2	This element must be monatomic.
3	Element has only specific polyatomic states.
4	Wrong charge on the ion.
5	This does not form polyatomic ions.
6	Expert doesn't do complex boron compounds.
7	A noble gas improperly present.
8	This is an illegal metal-metal compound.
9	Expert couldn't understand the formula.
10	Chemical has a group I don't recognize.
11	Elements appear to be misordered.
12	Too many parentheses to evaluate.
13	Subscript too large to evaluate.
14	Parentheses don't match.
15	Too many elements in formula.
16	0 is not a legal subscript.
17	Expert doesn't do most carbides.

to a one-element model, an error statement is added to the trace (Table VI). This completes the analysis of the one-element case.

Formulas that contain more than one element are given a preliminary classification based on several factors: if neither a metal nor a polyatomic ion is present, the chemical is classified as a molecular compound. If a metallic element or ion is recognized and no nonmetals or metalloids are present, the chemical is classified as an ionic compound. If a metal atom and one or more ligands are noted, it is classified as a coordination compound. If the formula can be found in a list of common names, it is named immediately, and the process terminates. Following this, a formula must be evaluated to determine if it is within the scope of the expert's ability and whether it has correctable errors. For example, it is scanned for unusual (HOH) or incorrect (ClCl) formulas and rewritten in standard form (patterns such as XYX or XX generally do not exist in inorganic compounds). Formula combinations that are indicative of organic compounds are also sought and rejected as previously described in the literature.[3] Borohydrides and inappropriate noble gas compounds are also excluded from further analysis; acids that require a "hydro" prefix are specially noted. Tables I and III are used extensively for these evaluations. Finally, if the formula is chemically sound, the order of the elements and groups based on their electronegativity is checked while taking into consideration exceptions such as NI_3. Only after the formula passes this preliminary chemical feasibility evaluation is the analysis for multielement compounds undertaken.

Chemical formulas are next split into two categories: those with two elements (binary compounds) and those with more than two elements or with a charge. Although both categories require the same analysis routines, the two-element case, which contains many common inorganic chemicals, has fewer degrees of freedom in terms of charge assignments and ordering than those formulas with more elements and therefore provides the program with more leverage in the naming process.

For two-element compounds, the formula that was cleaned of correctable errors is tested against a file containing common names. If a common name is found, the analysis is completed, and the common name recorded in the trace. If not, the formula is processed by the oxidation-state reconciler (discussed next) and then the naming routine. If this process is unsuccessful, a second iteration is made with looser constraints on oxidation numbers, permitting those states that are valid only under special conditions (e.g., O as +2 as in OF_2) to be tried. If this process is successful on either iteration, the steps used to accomplish the naming are added to the trace. If it fails, the cause of the failure is saved and reported to the program that called the expert. For cases with more than two elements, the common name check is not retried, but the oxidation-state reconciler and the naming routines (discussed later) are applied. However, releasing the constraints on the oxidation numbers may generate some technically correct yet bizarre combinations of oxidation numbers. Under such conditions, the correctness of the name cannot be guaranteed.

If naming has been successful, the name returned is refined as necessary to conform to conventions. For example, if the chemical is an acid, it is converted into the acid form of the name (e.g., renaming hydrogen phosphate as phosphoric acid). Stem changes (e.g., changing "phosph" to "phosphor") or the addition of the prefix "hydro" (e.g., renaming hydrogen chloride as hydrochloric acid) may also be necessary. The word "ion" is added to the names of formulas with charges, and the word "alloy" is added to the formulas that are composed solely of metals. Both the ordered name components and the composite name are returned in the trace.

Oxidation-State Reconciler. The routines that assign oxidation states and that perform the naming function deserve closer examination. The routine that assigns oxidation states is an essential, although not obvious, part of the naming process. The reason for this can be understood if one considers the entire process used to name a compound such as $FeCl_3$. The chemist analyzes this problem in a series of steps (probably performed subconsciously): (1) recognition of iron as a multivalent metal, (2) recognition of chloride as the anion, (3) assigning an oxidation state of −1 for the chloride, and (4) computing a +3 charge for the metal because there are three chlorides. These deductions will be combined with the naming rules to produce the name iron(III) chloride. For the computer to accomplish a similar analysis, more and smaller steps are required. The system must recognize that (1) two elements are present, (2) iron is a multivalent metal, (3) chlorine is a nonmetal, (4) the compound is ionic, (5) the elements are in the correct order, (6) no common name exists, (7) the nonmetal valence state is assigned before a multivalent element, (8) chlorine's most common oxidation number is −1, (9) there are three chlorines, so the total charge is −3, (10) one iron with an oxidation state of +3 is chemically possible, (11) a multivalent metal is written with a Roman numeral, (12) iron(III) is the cation, (13) anionic chlorine is named "chloride", (14) no prefix is necessary for the anion chloride, and (15) the complete name is iron(III) chloride. As can be seen, many of the steps involve recognizing or manipulating the oxidation numbers of the elements in the formula.

The reconciling of the oxidation states is a recursive procedure. To begin, the parse tree must be resorted so that elements and groups are ordered based on their ease of assignment. The first to be assigned are groups of known charge, hydrogen and invariant metals.[3] After this nonmetals, metalloids, variable metals, and groups with several possible charges are addressed. A pointer starts at the root node of the tree and selects the first available valence from CHEMPROF's element description table for the species or the special group list (Table IV). It computes a running charge total and recursively looks at the next node. When the last node is encountered, the oxidation number required for that node to effect a charge balance is calculated. If the last number assigned is legal for the species, the oxidation numbers are confirmed (instantiated), and the naming process begins in reverse order, as the routine works its way out of the recursion. If the oxidation number of the last node is not permissible, then the routine fails and proceeds up one level where the next possible oxidation number for that node is selected. This procedure is repeated until the root node is reached. If the process has succeeded, the name is ready for final refining before being put into the trace. If it has failed, the control program can accept failure or again call the reconciler with looser constraints on allowable oxidation numbers.

Table VII. Rules Governing the Assigning of Names to Elements of a Formula

```
Oxidation States:

IF (oxidation state > -1) AND (component = variable metal)
    AND (each atom has an integer charge)
  THEN append appropriate Roman numeral to element name.

IF (oxidation state > 0) AND (component <> univalent metal)
    AND (component position = first)
  THEN append appropriate Roman numeral to element name.


"ide" suffix:

IF charge < 0
  THEN append 'ide' to element stem.


Numeric Prefix:

IF (compound = [univalent metal, multivalent metal, hydrogen])
    AND (number of atoms > 1) AND (each atom has an integer oxidation
state)
    AND (((component number > 2) AND ((number of groups > 1)
          OR (univalent metal+multivalent metal+hydrogen > 1)))
      OR ((component number = 2) AND (species = negative ion)))
  THEN set appropriate prefix.

IF (compound IN [univalent metal, multivalent metal, hydrogen])
    AND (each atom has an integer charge)
    AND (species = negative ion) AND (no multi-atom components)
  THEN set appropriate prefix.

IF (species is an ion) AND (component = halogen)
    AND (More than 1 element present)
  THEN set appropriate prefix.

IF (species is composed of two halogens)
      AND (component is the less electronegative)
  THEN set appropriate prefix.

IF (nonmetals-hydrogens+metalloids > 1) AND (number of component <> 2)
    AND (compound IN [univalent metal, multivalent metal, noble gas])
    AND (number of groups = 1) AND ((at least one multi-atom element)
      OR (component IN [oxygen, sulfur, selenium]))
  THEN set appropriate prefix.

IF (compound IN [multivalent metal, nonmetal, metalloid, noble gas])
    AND (extended oxidation state list in use)
  THEN set appropriate prefix.
```

Naming Process. The naming routine is invoked at each node once the oxidation reconciler has successfully obtained a charge balance. The process is highly rule driven. Assigning names to the individual elements involves applying the correct rule for both the type of chemical and for the position of the element within the chemical (e.g., iodine tribromide, but bromine monofluoride). Elements in different parts of the periodic table must be treated differently. For example, only oxygen and sulfur in column 6A can form "per" anions. "Oxide" can swallow up the last letter of some numeric prefixes (e.g., tetraoxide becomes tetroxide), but the anions of the rest of the elements in column 6A do not. Special groups, such as I_3^- and Hg_2^{2+}, must be detected and appropriately named, while all elements in alloys must be named as metals. The set of rules in Table VII is then applied to determine if a prefix should be added to a particular element, if an oxidation state should be included, or if the name should end in "ide".

The naming process at each node is carried out separately. Therefore, the type of information that is stored in Table I is essential to allow this routine to function properly. For example, when faced with a node containing two bromines, the naming routine does not know if the compound is $CaBr_2$, SBr_2, or Br_2O. It must rely on the abstracted information in Table I and the charge information stored at the node to be successful. In effect, Table I is the concentrated description of the nature of the formula. If the expert did not use this approach, it would have to have a large rule base that related specific information from one part of the formula to the rest of it. Carried to its logical conclusion, one would simply put every possible chemical in the rule base and just do a pattern match on the whole formula.

FORMULA-RECOGNITION EXPERT

Sometimes the formula entered by the user is already known to the computer system. This occurs because the computer has given the user the formula as an exercise or because constraints limit or uniquely define the user's input. The formula-recognition expert attempts to match the user formula with the system or reference formula, and if this is not possible, to identify formula errors. A list of the detectable errors for matching is given in Table VIII.

Table VIII. Errors Detected by Formula Recognition Expert

missing subscript	unmatched parentheses
wrong subscript	incorrect charge
ordering error	wrong sign on charge
extra parentheses	improper charge
missing parentheses	unneeded charge
mislocated parentheses	space in formula
missing element	combining error (e.g., HOH)
subscript of 1	+1 or −1 instead of + OR −
extra element	

If the string input by the user and the expected string are identical, the match is quickly recognized and no further action required. If the two strings do not coincide, the strings are broken into their individual elements. This is accomplished by using arrays of structured variables where each variable contains the name and the amount of each element present as well as indicators concerning the opening and closing of parentheses. In the process of disassembling the strings, errors such as spacing, inappropriate placement of charge, the use of "1" or "0" as a subscript, and mismatched parentheses are detected. After disassembly of both the user input and reference string, cases with incorrect or missing charge, charges with a wrong sign, and extra or missing parentheses are identified by comparing the attributes of the two strings set during the disassembly process. All of the above errors are readily detectable by simply scanning the user and reference formulas.

The next step, identifying and supplying information regarding the detection of missing or surplus elements in the user formula, is more complex. To accomplish this, each element in the user input string is compared to the reference string; elements not found in the reference string are deleted from the user string. Similarly, each element in the reference string is compared to the user string; elements not found in the user string are added to the user string. Such a double search is essential because the user may have reversed two elements (e.g., HO_3N instead of HNO_3), split the appearance of an element into two occurrences (e.g., HONO instead of HNO_2), or grouped the double occurrence into a single one (e.g., $N_2H_4O_3$ instead of NH_4NO_3). Each of these error types is identified as a grouping error, which should not be confused with missing or extra element errors. Furthermore, these grouping errors cannot be accurately detected solely from a linear comparison of the formulas and, hence, require the double search.

The final step in the formula recognition is to detect errors in element order, parentheses, or subscripts. This is accomplished by a simultaneous linear scan of the disassembled formula arrays, left to right, since both arrays now contain the same elements. If necessary, the user array is reordered to match the reference array, and incorrect occurrences of an element are modified. Due to the complexity of the detection task, only one error is tolerated anywhere within the evaluation process. A second error immediately causes the termination of the recognition process and the reporting of the two detected errors, with the errors being returned to the calling program in a two-word bit mask.

APPLICATION OF THE EXPERTS

Our use of the formula-to-name expert and the formula-recognition expert has been within the context of the CHEMPROF project. The traces from the formula-to-name expert are converted to explanations of how to assign a particular name (Figure 3) that are available to students as they practice. These traces could also serve as the basis for the teaching logic

How CHEMPROF names P_4O_{10}:

Recognize that P_4O_{10} is a molecular compound (because it contains only nonmetals or metalloids, and no ions).

To name molecular compounds, name the first element and then the second element with an ide ending.

The element P is named phosphorus. Since there are 4 P's, the prefix tetra must be added to give tetraphosphorus.

The element O is named oxide. Since there are 10 O's, the prefix deca must be added to give decoxide.

Therefore P_4O_{10} is named tetraphosphorus decoxide.

Figure 3. This figure shows the explanation of a formula-to-name conversion that CHEMPROF gives to the user based on the trace shown in Table V. It explains each rule that the expert applied. (On the screen, elements, names, and emphasized words are in color.)

to lead the students through the name-generation process as we did with traces from the oxidation number expert.[3]

Conversely, the teaching logic processes the error bit mask from the formula-recognition expert to pinpoint the user's errors. Here, there is no means of tracing what should have happened because the expert has not been applying chemical knowledge but simply parsing knowledge. The limit of only two errors when the formula-recognition expert is invoked keeps the error analysis from growing into an overly long paragraph. This expert should also be useful in identifying which of several candidate formulas the user was trying to enter if the user made a simple typing error, a capacity which will be useful when users are typing in information while trying to solve a more complex problem.

DISCUSSION

The human mental process for arriving at a name from a chemical formula appears to be part pattern recognition and part analysis based on rules which are rapidly applied. Sometimes the patterns are deceptive and the rules fail because the chemicals are uncommon (e.g., many people do not recognize $COCl_2$ as phosgene). The human expert apparently first tries to match the formula as a whole to a known formula using no rules and then resorts to smaller patterns (e.g., polyatomic ions) if the larger pattern match fails. The smaller the patterns, the larger the rule component in the naming process. Since the element symbols are soon encountered, this is a bounded process which will produce a descriptive name, even if because an obscure naming rule is not known it is not the preferred IUPAC name.

The computer-based naming process cannot exactly imitate the human expert who uses interconnections between processing levels but must settle for the human journeyman level of performance. The fundamental problem was to develop a rule-based approach to convert chemical formulas to names that would partition the process into steps that could then be refined to improve performance. If one looks at a dozen different formulas, it logically seems that there must be some first step in the naming process that can be applied to all of them, regardless of the final name. The building of the parse tree was an effort to find this first step even though this step is useless in the case of common names, unless the formula is written in an unusual form, such as HOH. Once the tree is built, however, the rest of the routines have a common data structure upon which they can work independently.

The necessity of assigning oxidation states to the various parts of the formula before naming could start was dictated by the desire to maintain chemical integrity of the naming process. Clearly, one can name nonexistent compounds such as CO_3 or AlF_5 by applying IUPAC rules. We felt it was important, however, to correct users who submitted such illegal names. Unfortunately, there is no foolproof method of detecting all illegal entries because oxidation states that occur under highly unusual conditions could be used under other conditions when they are inappropriate. An effort to prevent this from happening is made by using a two-pass approach to the assigned oxidation states in which only common oxidation states are used during the first pass. This heuristic appears to be adequate under most circumstances, although some states are so unusual that a special mechanism might be considered to keep them suppressed except when their specific case is detected. For the most part these have been excluded from consideration as being beyond the scope of the program (e.g., carbides, organic compounds, and borohydrides). The use of recursion as the basis of the valence assignment process has worked well, although occasional procedural fixes have been necessary to handle some of the complexities noted below which do not fit into the partitioning theory model we have developed.

The naming step was made more complex by rule exceptions involving nonmetal binaries where the order of electronegativity is reversed (e.g., NI_3) making the ordering routines unduly complicated. Pairs such as NO_2^- and NO_2^+ where one is positive and the other negative also tax the ordering process since there is no consistent way to weigh the relative electronegativity of groups and elements. The unusual naming possibilities for oxygen and sulfur are also troubling. While in most compounds oxygen has a reliable oxidation state of −2, when it appears as "O_2" (as in Na_2O_2), it has to be regarded as a potential polyatomic ion with an oxidation state of −1. Hence, the same combination can be either a simple ion (oxide, O^{2-}), or a polyatomic ion (peroxide, O_2^{2-}), thus defeating a major partitioning scheme needed to separate the domain into solvable subsets. Similar behavior occurs with S_2 as well. Solutions to these problems had to be implemented procedurally since parts of the mechanism to accomplish the solutions had to be placed in functionally different portions of the program. The clear partitioning was thereby lost.

Complicating the whole process is the possibility of errors in the user formula. Each portion of the process is responsible for the detection of errors which are uniquely related to its area of activity. They range from incorrect subscripts, misordering of elements, inappropriate noble gas compounds, or the charge being placed in the middle of the formula. Detection of an error terminates the conversion process (unless that error is correctable), carries the error to the exit routine, and returns it to the program calling the expert. The sophistication of our error detection, as summarized in Tables VI and VIII, is considerably more than Cassen could attain a decade ago with the NOMEC program written in BASIC.[12] Both the expert modules we described are written in Turbo Pascal.

Both experts have been refined through use in classroom situations and are now adequate to handle the common, and many not-so-common, inorganic formulas. The use of the parse tree in the formula-to-name expert has been successful in that it has allowed a substantial partitioning of the naming process and formulation of the naming rules. Our subsequent efforts will focus on converting chemical names into chemical formulas and then incorporating both of these experts into routines solving higher level chemical problems.

ACKNOWLEDGMENT

We gratefully acknowledge a grant from the Undergraduate Teaching Improvement Council of the University of Wisconsin System, which helped fund the evaluation of these experts in the classroom.

REFERENCES AND NOTES

(1) Eggert, A. A.; Middlecamp, C. H.; Kean, E. CHEMPROF—A Tutor for General Chemistry. *J. Art. Intell. Educ.* **1990**, *2* (1), 47–62.

(2) Eggert, A. A.; Middlecamp, C. H.; Kean, E. CHEMPROF—An Intelligent Tutor for General Chemistry. *J. Chem. Educ.* **1991**, *68*, 403-407.
(3) Eggert, A. A.; Middlecamp, C. H.; Kean, E. An Oxidation Number Assignment Expert for CHEMPROF. *J. Chem. Inf. Comput. Sci.* **1990**, *30*, 181-187.
(4) Nomenclature for Inorganic Chemistry. *Handbook of Chemistry and Physics*, 60th ed.; CRC Press, Inc.: Boca Raton, FL, 1979; pp B-27-B-48.
(5) Barr, A.; Feigenbaum, F. A. Understanding Natural Language. *The Handbook of Artificial Intelligence*; William Kaufmann, Inc.: Los Altos, CA, 1981, pp 223-321.
(6) Tomita, M. *Efficient Parsing for Natural Language*; Kluwer Academic Publishers: Boston, 1986.
(7) Allen, J. *Natural Language Understanding*; Benjamin/Cummings: Reading, MA, 1987.
(8) Loeffler, P. A. Fundamental Concepts in the Teaching of Chemistry. *J. Chem. Educ.* **1989**, *66*, 928-930.
(9) Lancashire, R. J. Naming Inorganic Compounds. *J. Chem. Educ.* **1987**, *64*, 900-901.
(10) Fernelius, W. C. Correct Methods for Naming Compounds. *J. Chem. Educ.* **1987**, *64*, 901-903.
(11) Fernelius, W. C. Numbers in Chemical Names. *J. Chem. Educ.* **1982**, *59*, 964.
(12) Cassen, T. A Versatile Program for Drill in Inorganic Nomenclature and Formula Writing. *J. Chem. Educ.* **1981**, *58*, 49.

Correlations between Chemical Structure and Normal Boiling Points of Halogenated Alkanes C_1–C_4†

ALEXANDRU T. BALABAN,*,‡ NIKHIL JOSHI,§ LEMONT B. KIER,*,§ and LOWELL H. HALL∥

Organic Chemistry Department, Polytechnic Institute Bucharest, Splaiul Independentei 313, 77206 Bucharest, Romania, Department of Medicinal Chemistry, Medical College of Virginia, Virginia Commonwealth University, Richmond, Virginia 23298-0540, and Department of Chemistry, Eastern Nazarene College, Quincy, Massachusetts 02170

Received October 16, 1991

Correlations were investigated between topological indexes representing the chemical structures of 532 halo- and polyhaloalkanes with 1–4 carbon atoms and their boiling temperatures at normal pressure. Effects of constitutional isomerism and replacing one halogen by another on the boiling points of mono-, di-, tri-, and tetrahaloalkanes were studied.

INTRODUCTION

Halocarbons as a class have found many applications in modern society. They have been used as pesticides (DDT, γ-hexachlorocyclohexane), solvents (chloroform, carbon tetrachloride), alkylating agents (mustards), anesthetics (halothane), plastics (PVC), etc. Recently the use of chlorofluorocarbons (CFCs) in cryogenic systems and aerosols has stirred up a controversy regarding their environmental effects.[1] The high C–F bond energy makes these compounds extremely stable and resistant to environmental degradation. Hence, these compounds remain in the atmosphere for a long time. When they reach the stratosphere they are decomposed photochemically. The decomposition products of these CFCs catalyze the breakdown of the ozone layer, which is an important shield against the cancer-causing short-wavelength ultraviolet rays.

Many major industrial producers of CFCs are now replacing CFCs without any hydrogen atoms by CFCs with hydrogen atoms. The presence of hydrogen atoms makes these compounds degrade 20–200 times faster. However, the volatility of these compounds is also different from the CFCs without any H atoms. In this study we examine effects such as the replacement of halogen atoms by hydrogens in haloalkanes and attempt to obtain quantitative relationships relating the structure of the halocarbons to their boiling points. These equations could then be used to predict the volatility of haloalkanes.

The data used in this study were collected from several sources: Beilstein's *Handbuch der Organischen Chemie*, the *CRC Handbook of Chemistry and Physics*, Heilbron's *Dictionary of Organic Compounds*, and some other books[2] and primary literature sources. Since at normal pressure compounds with boiling points (bps) above 300 °C cannot be distilled without decomposition, the highest number of halogens possible for C_1–C_4 halo derivatives is ten for F, eight for Cl, four for Br, and two for I.

BOILING POINTS OF HALOALKANES

As seen in Figure 1, boiling temperatures of all halomethanes vary nonlinearly with increasing numbers of halogen atoms. For the heavier halogens, the curvature is less evident because the increase in molecular weight leads to a marked increase in boiling points; however, for fluorine compounds the boiling point curve versus the number of hydrogens replaced by fluorine atoms has a maximum when about half of the hydrogen atoms have been replaced. This phenomenon is encountered in higher haloalkanes even when less than half of the hydrogens have been replaced by fluorine atoms. Several other interesting trends have been observed in the boiling points of haloalkanes:

(i) As with the alkanes, branching also lowers the boiling points of the haloalkanes. Intermolecular forces are weaker for branched molecules than for linear ones, and even more so for spherically-shaped molecules. In isomeric systems this is demonstrated by the observation that in monohalo-*n*-alkanes the bp drops on moving the halogen from the terminal carbon toward the middle of the chain.

(ii) In the case of dihalo-*n*-alkanes, geminally-substituted compounds boil at lower temperatures than vicinally-substituted ones; with an *n*-butyl chain the bp increases in the sequence: 2,2- < 1,1- < 2,3- < 1,2- < 1,3- < 1,4-dihalo-*n*-butanes.

(iii) Isomerism leads to diagrams such as Figure 2 for polyhaloethanes or Figure 3, panels A and B for polyhalopropanes (hal = F or Cl). It can be seen that the highest bp is attained for the terminally-disubstituted dihalo derivatives. Figure 4 compares bps of α,ω-dihaloalkanes with n = 1–9 carbon atoms, and Figure 5 displays bps of 1-haloalkanes with n = 1–10. The same nonlinearity can be seen as in the dia-

† This work was conducted at the Virginia Commonwealth University and supported by a grant from Sterling Drug, Inc.
‡ Polytechnic Institute Bucharest.
§ Virginia Commonwealth University.
∥ Eastern Nazarene College.

0095-2338/92/1632-0233$03.00/0