Converting Chemical Names to Formulas: A Second Expert Problem

Arthur A. Eggert,* Anthony T. Jacob, and Catherine H. Middlecamp

Chemistry Learning Center, University of Wisconsin—Madison, Madison, Wisconsin 53706

Received January 4, 1993

The ability to write a chemical formula for an inorganic chemical, given its chemical name, is essential both to the study and practice of chemistry. The need for this ability extends to computer programs employed for instruction and research. To study this name-to-formula conversion problem, the authors have developed a model in the form of a PC-based expert system that maps names into chemical formulas. The model uses two systems of checking to detect spelling errors and determines subscripts by use of formal charges. Spelling and spacing errors are cleared up using a declarative database, but the name decoding is highly procedural in implementation.

INTRODUCTION

The growing usage of computers by chemists increases the importance of developing software that can converse fluently using chemical names and formulas.[1,2] In a previous article, we described two expert models (programs) for working with inorganic chemical formulas.[3] The first addressed the conversion of a chemical formula (entered by a user) to its chemical name. The second compared an entered formula with a known formula, reporting any errors in the entered formula. The purpose of this previous work was to support CHEMPROF, an intelligent tutoring system for general chemistry.[4,5] The complementary problem, the conversion of names to formulas, is the subject of this paper.

Conceptually this problem is easier than the former one, because a name corresponds to one and only one chemical formula. In contrast, chemical formulas can be called by one or more names. Nevertheless, the errors in a chemical name that can arise from spelling, typing, or a lack of chemical knowledge make such a conversion nontrivial.

Inorganic chemical nomeclature is also nontrivial and has numerous rules and idiosyncracies that are periodically debated and revised. The most recent set of nomenclature recommendations was issued in 1990 by the International Union of Pure and Applied Chemistry (IUPAC).[6] For the most part, our expert model follows these recommendations. It does, however, default to American spellings (e.g., "cesium" rather than "caesium") and to the naming conventions commonly used in introductory chemistry texts (e.g., "hydrogen carbonate" rather than "hydrogencarbonate").

In working with chemical names, the computer must handle two types of problems. In the first, the computer does not know which name the user is trying to enter. The problem is to take a string of characters and convert these into a chemical formula recognizable to a chemist. Difficulties arise if the name entry is misspelled, has incorrect chemical proportions, or is not the chemical intended. Regardless of any such errors, if a computer model can derive the chemical formula that matches the one that a chemist would associate with the flawed input, then the model is good. If the computer could always come up with the same chemical formula as the chemist, even when there is significant deviation from the standard form, then the program would be a perfect model of chemical name recognition. This goal is not likely to be achieved in the foreseeable future.

In the second case, the computer has knowledge of what the entry should be. The problem is to determine what is wrong with this name entry and report this to the user. This is primarily a parsing problem, although both mechanical (e.g., "soduim" instead of "sodium") and chemical (e.g., "ate" instead of "ite") errors may be involved. Unfortunately, mechanical errors can sometimes obscure chemical errors. For example, the user may enter "sulfte" when trying to type "sulfite", but should have been entering "sulfate".

The problem of computer text recognition has been of interest to computer scientists and cognitive psychologists for three decades.[7–9] In the general case, modeling of the input requires that a substantial amount be known about the real world by the parser and that a complex set of relationships and functions be representative of informational nuances. The solution to this general problem has been elusive because humans use highly complex semantics to detect the meaning of character strings and do not simply rely on the syntax of the language. Computers are good at syntax because it can easily be specified by patterns. However, they are not good at semantics which requires extremely complex relationships between words and even a cultural knowledge of meaning.[10] In the problem of chemical name recognition, we can greatly restrict the domain, since we require only a few thousand words and perhaps a few dozen formats to name the vast majority of chemicals encountered in inorganic chemistry. Clearly, for the purposes of undergraduate chemistry courses, we can devise a reasonably complete model for inorganic chemical nomenclature, leaving outside of it only the chemical species of interest to specialized research groups.

SCOPE OF THE PROBLEM

In formulating a model for naming inorganic compounds that could be used as the basis of a computer program, the starting point must be the development of categories of possible user errors.[11] The most obvious error is in spelling or typing, and Table I lists four different types. One type (1A) results when the user actually thinks the word is spelled differently. For example, a number of people misspell the element fluorine as "flourine". A reasonable set of these 1A errors can be identified and specifically sought for wherever they might appear. Another error (1B) results from the omission of a letter, e.g., typing "carbnate" for carbonate. This error can appear anywhere in a word but tends to appear more frequently toward the middle where it may be less readily detected by the user. A third type of error (1C) is the addition of an extra character, e.g., typing "broimide" for bromide. The fourth error (1D) results from transposing letters, e.g., entering

Table I. Catalog of Naming Errors

Type	Nature	Description
	1. Spelling	
1A		user knowledge base error
1B		omission of letter
1C		spurious letter added
1D		two letters transposed
	2. Spacing	
2A		user knowledge base error
2B		space omitted
2C		extra space
2D		space shifted
	3. Parenthesis	
3A		parentheses not matched
3B		required parentheses missing
3C		inappropriate characters enclosed
	4. Oxidation State	
4A		needed oxidation state missing
4B		unneeded oxidation state
4C		wrong oxidation state
	5. Name Parts	
5A		duplicate parts
5B		misordered parts
5C		missing part(s)
5D		part misnamed
5E		part unrecognized
5F		alternate name for part
	6. Prefix	
6A		unneeded prefix
6B		missing prefix
6C		incorrect prefix
	7. Suffix	
7A		ide/ine confusion
7B		ate/ite confusion
7C		unnecessary suffix
7D		suffix improperly used
	8. Whole Name	
8A		wrong chemical name
8B		unrecognizable name
8C		alternate legal name

"cholrine" instead of chlorine. Errors 1B, 1C, and 1D are most often due to errors in typing.

Spacing errors form another set (see Table I). One of these errors (2A) arises from the systematic addition of a space where it does not belong, e.g., entering "copper (II)" instead of copper(II). Another (2C) is the addition of an extra space anywhere in the name. Error 2A is not just a special case of 2C, since the former is intentional, while the latter is not. A third spacing error is the omission of a space (2B), causing words to run together. The final spacing error (2D) is a combination of 2B and 2C, appearing as a space being shifted one character to the left or right. All errors except 2A are assumed to be artifacts of keyboard entry, rather than a chemical misunderstanding.

Another family of errors involves parentheses. Having unmatched parentheses (3A) such as in "copper(II ion" may result from a typing mistake. Omitting necessary parentheses (3B) as in "copper II ion" represents a user knowledge error. Placing inappropriate characters between parentheses (3C) can also indicate a lack of user knowledge.

Errors in the usage of oxidation state are of three types. In the first case (4A), a needed oxidation state is omitted; e.g., "copper sulfate" is entered instead of copper(II) sulfate. In the second case (4B), an oxidation state is supplied that should not be, e.g., using "magnesium(II)" instead of magnesium. Finally, the oxidation state itself may be incorrect (4C), as with "sodium(II)" which is not chemically feasible.

Chemical names may contain more than one word (e.g., "sodium hydrogen carbonate"). Each separate word could be called either a name fragment or name part. Since the former might be misleading to chemists (chemically, the word fragment implies a piece of a molecule), we have opted for the term "name part". Errors in name parts are diverse in nature and are the most common errors made intentionally to confuse the computer model. Name part errors such as duplicate parts (5A) (e.g., "sodium sodium carbonate") and misordered parts (5B) (e.g., "carbonate sodium") seldom occur by accident, but the model must still be able to handle them. A missing parts error (5C) may or may not be intentional. Misnaming parts (5D) or giving undecipherable strings for the names of parts (5E) is more commonly a result of a lack of chemical knowledge, the latter perhaps of very bad typing. Finally, using an alternate name (5F), such as "bicarbonate" instead of "hydrogen carbonate", may lead to the correct determination of formula but must still technically be regarded as an error because it is not what the model expects.

Prefix errors can be categorized as unneeded (6A), missing (6B), or incorrect (6C). An example of an unneeded prefix is the "di" in "magnesium dibromide". An example of a missing prefix occurs in "phosphorus iodide", where iodide should be triiodide. An incorrect prefix such as "di" on aluminum dioxide would force an impermissible oxidation state. Sometimes this error (6C) arises because the word "ion" is omitted; e.g., "aluminum dioxide ion" is a correct name for AlO_2^-, whereas "aluminum dioxide" is not.

Suffix errors do not correspond exactly to those for prefixes, because oxidation states can obviate the need for prefixes. There are also two special cases of suffix errors that warrant special consideration: mixing up "-ine" and "-ide" (7A) and "-ate" and "-ite" (7B). Both of these are common errors and can indicate a lack of chemical knowledge on the part of the user. Suffix errors also include those that are unneeded (7C) or incorrect (7D). For example, an unnecessary suffix is the "ic" on "manganic dioxide", and an improper suffix would be the "ide" in "lithide".

Some errors apply to the whole name. These include giving the wrong name (8A) or an unrecognizable name (8B), the latter meaning that nothing can be identified in the name. If the user gives an alternate name for a chemical (8C), i.e., "ferric chloride" for "iron(III) chloride", this needs to be identified to the user.

The means by which errors are detected and processed will depend upon the reason for which the name string is being parsed. If the user's ability to name chemicals is being evaluated, then as many errors as possible must be detected and identified for the user. This is done, for example, when a program is helping students learn chemical nomenclature. Conversely, when a program is trying to obtain chemical information from a user, the goal may be to correct errors, rather than classify them, thus facilitating information transfer. Different processes are involved in these two cases, and therefore different expert models are necessary. We will next look at how these two models are created.

NAME VERIFICATION EXPERT

Name verification is useful in two cases. In the first, the computer program knows a chemical name, gives the user its corresponding chemical formula, and requests input of the chemical name. In the second, the computer program is working with a set of chemical names and is trying to match the user's input with one of them. In the former case, the computer's diagnosis of errors can aid the student in better naming chemicals in the future. In the latter, the computer's knowledge of the matching errors can help it distinguish which of two similar chemicals was meant.

Table II. Ine/Ide Pairs

hydrogen	hydride	sulfur	sulfide
boron	boride	chlorine	chloride
carbon	carbide	arsenic	arsenide
nitrogen	nitride	selenium	selenide
oxygen	oxide	bromine	bromide
fluorine	fluoride	tellurium	telluride
silicon	silicide	iodine	iodide
phosphorus	phosphide	astatine	astatide

Table III. Common Prefixes

mono	mon	di
tetra	tetr	tri
penta	pent	per
hexa	hex	bi
hepta	hept	thio
octa	oct	hypo
nona	non	
deca	dec	

The first task in verification is to clean up the user-supplied name. A cleansing pass of the name entails the removal of any double spaces (not counted as an error), the detection of any extra spaces next to parentheses, and the detection of any mismatched parentheses. The name is next cut apart at the spaces and placed into an array of text strings. If any of these text strings are prefixes, they are attached to the word that follows. Because chemical names rarely contain more than four words, if more than four are present, the additional words are compressed for later examination. Any errors are noted in a bit mask for the whole name.

A known name is set up in the same way, but this time no errors are expected. The number of words is counted. Prefixes and the presence of the suffixes "-ide" and "-ine" are noted. A list of alternate words for the name parts (e.g., "ferrous" for "iron(II)") is set up if such alternates have been supplied. A specified dictionary of chemical names is loaded.

A word by word match is next attempted. If the input exactly matches the expected name or alternate, the model returns the indication that the match was successful, along with any scanning errors previously detected and corrected.

If the match is not completely successful, it is necessary to determine the errors in the input string relative to the expected name strings. For example, is there an error in word spacing? Are there missing or spurious spaces? To accomplish this, the parser must determine whether the input contains legal chemical name parts or misspelled name parts or is undecipherable. This is done working from left to right in an iterative process.

If a word has already been identified and placed in proper form, it is skipped in each succeeding pass. If not, the word is compared against all words in the expected name. If a match is found, it is noted against the position in the expected name where the match succeeded. If no match is found, the word is run through a general spelling checker using a dictionary of inorganic chemical terms. If the word matches a chemical name in the dictionary, it is marked as identified but wrong because that name part is absent from the input name. If no match is found, an effort is made to combine the word with any carryover from the previously identified word and the process is repeated. This is done to discover spacing errors. If a match is finally detected, but there are letters left over at the end of the word, these are placed in a carryover bin for the next word.

While this is in progress, checks are also made for "-ine"/"-ide" (Table II) and "-ate"/"-ite" errors and for the correctness of oxidation states. Missing or incorrect oxidation states are noted. Prefixes are checked to see that they are correct, present or absent, as appropriate, and spelled correctly (Table III). Impermissible spellings such as "tetraoxide" are detected at this point. Spelling checks are done with and without any prefixes to maximize the chance of catching any errors. Whenever possible, a mistake is corrected so that the model can detect further errors in the input. Complex errors are those which combine parts of both the expected and alternate names (e.g., "sodium hydrogen bicarbonate").

Most errors are detected through this combination of tests. The process is repeated as long as a name part exists that is unreconciled to the input. However, because each pass either determines the first word it considers or marks it as undecipherable, the next pass starts one word further into the name. This restriction prevents infinite looping. The analysis problem is also reduced by making use of the fact that no two letter names exist and very few exist with three letters. When there are leftovers of these sizes, they are invariably fragments remaining from spelling errors which have been corrected and may be discarded.

When all else fails, spaces between unidentified words are removed, and the words are redivided on the basis of the closest match of the left end of the combination to known words in the dictionary. This detects extra spaces and shifted spaces. Endings such as "ic", "ous", "ate", "ite", and "yl" are used to guess the exact point of division. On the basis of all the errors detected, a bit mask is prepared for each word in the input name.

The final step is recording the errors in the bit mask that apply to the whole name. These errors include duplicate name parts, name parts out of order, spacing errors, and missing name parts. The array of error bit masks is then passed from the model back to the calling program.

Three errors were the most complex to address. The first involves an error in spacing combined with a misspelling. For example, if "calciusulfate" is entered instead of calcium sulfate, the model must insert a space between two words, the first of which is misspelled by truncation and therefore misleading in the number of letters which should be skipped before seeking the second word. The second is whether to seriously consider the guesses that the spelling checker returns, particularly those which are more than a few letters longer or shorter than the unknown part. If a suggestion matches a part of the expected name, the question is easy, but if it indicates a misspelled word which is not part of the expected name, it may be the result from improperly considering the first few letters of the next part (e.g., "potassium iodide" could be matched to "potassamide"). The third is determining whether the user answered by randomly pushing keys and identifying the entry as such, rather than reporting some combination of legitimate mistakes.

Because of the multiple iterations through the name and numerous checks for combinations of errors, this model has numerous subroutines and is structurally very deep. Debugging the model was difficult because of the interactions possible between the various types of errors. While an effort was made to capture as much of the knowledge as possible into declarative forms (e.g., arrays of structure variables and spelling lists), this was only successful to a limited extent. Almost all the parsing knowledge is procedural.

NAME-TO-FORMULA EXPERT

While the name-to-formula model might be expected to closely parallel the name verification model, this is not the

Table IV. Letter Combinations Starting Chemical Words ("Name Starters")

acet	acti	alco	alum	amer	ami
amm	amo	anti	arg	ars	ast
aur	azi	azo	bar	benz	ber
bism	bsmu	ble	bor	brim	brom
borm	cad	cal	cacl	camp	car
crbo	cab	cass	cer	ces	char
chl	chor	clor	chol	hlor	chr
chom	cin	cit	cob	colu	cop
coru	cupr	cupp	cur	cyan	cayn
deut	diam	diaz	dys	dsp	eins
erb	eth	eur	fer	flu	flo
form	frma	fran	fruc	gad	gal
germ	grm	gly	gluc	gold	glod
grap	graf	haf	hel	hema	hol
hyd	imi	ind	iod	ido	iri
iron	iorn	irn	kal	kies	kryp
krip	kyp	lact	lant	lau	law
lead	lime	lith	lut	lye	lie
mal	magn	mang	marb	men	merc
mol	meth	natr	neo	nep	nic
nik	niob	nitr	nito	nob	osm
oxa	oxi	oxo	oxy	ozo	pal
phos	picr	plat	plu	pol	pota
tass	prae	pras	pro	prus	quar
qui	rad	rhe	rho	rub	rut
sal	slat	sam	scan	sel	sil
sli	slv	sod	soi	sor	stea
suc	sul	stan	stib	stro	tab
tant	tart	tech	tel	terb	thal
thio	thor	thro	thul	tin	tit
trit	tirt	tung	ura	ure	van
verm	vine	ving	vitr	wat	wolf
wlfg	xen	ytt	zin	zir	

case. One difference arises because when converting a name to a formula, the computer has no expected name parts to use as a guide. Consequently, spelling and spacing errors are significantly more difficult to handle. A second difference in this model arises because it is necessary to make chemical sense out of the name after it is cleaned up. In contrast, the task for the name verification model ends when the parsing is done. Finally, the name-to-formula model is not concerned with the nature of input errors but rather in being able to correct for them whenever possible. As a result, this model makes only passing reference to the errors in the parsing phase unless these errors cannot be cleaned up for the formula assignment phase.

Parser. The process begins with an effort to correct any spelling and spacing errors. The assumption is made that the user will correctly type in the first few characters of the name or make errors that belong only to a limited set, e.g., "clor" instead of "chlor". Possible character combinations for starting chemical names ("name starters") are listed in Table IV. Thus, the first step is to strip the name of any spaces and attempt to match its first three to five letters against allowable name starters. For each of these name starters, a list of possible name parts is extracted from tables that will be discussed later.

An example of what results from this extraction process is shown in Table V for the input "calcium sulite". The name column shows possible matches, of which three have already been voided because they are not feasible for the first part of a name. The analysis progresses through this table of possibilities one character position at a time. Each word is scored relative to the compacted input name and that score compared with the best score so far for the character position. This process is repeated until the entire list of possibilities has been exhausted. Corrections to the entered name are made, if necessary, on the basis of the best match whenever possible.

Table V. Spelling Correction of Calcium Sulite

position	pointer	name	score	best so far	comment
1	20	calcium	10.000	calcium	element
1	98	californium	5.636	calcium	element
1	2063	(voided)	0.000	calcium	calcia—too short
1	2064	(voided)	0.000	calcium	calcite—too short
1	2065	(voided)	0.000	calcium	calomel—too short
8	16	sulfur	6.500	sulfur	element
8	1013	sulfonyl	4.875	sulfur	cationic group
8	1026	sulfuryl	4.875	sulfur	cationic group
8	1113	sulfite	9.714	sulfite	anionic group
8	1114	sulfate	8.429	sulfite	anionic group
8	4142	(voided)	0.000	sulfite	X-sulfate; no X
8	4216	(voided)	0.000	sulfite	X-sulfonate, no X
8	6016	sulfide	8.286	sulfite	elemental anion
8	7016	sulfuric	5.750	sulfite	elemental -ic
8	8016	sulfurous	4.333	sulfite	elemental -ous

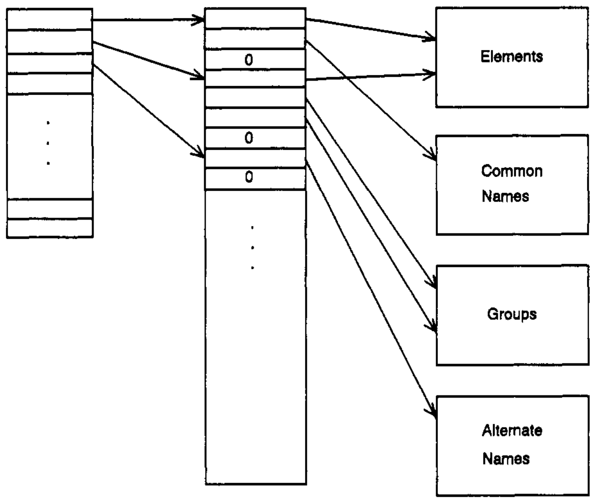

Figure 1. Structure of the name starter file. The left file contains the three to five letter name starters and pointers to the central file. The central file contains arbitrarily long lists of pointers to tables with specific classes of compounds.

However, no correction is attempted unless the name match is above a certain threshold score. This minimum level of confidence is used because the subsequent naming routine, based on chemical knowledge, is able to analyze name parts better than the parser which relies solely on word form and tables of legal chemical names.

The foundation of the parsing and correction routine is a table containing the name starters listed in Table IV and diagrammed in Figure 1. This table is a flat (sequential) file of name starters, each with a pointer that acts as an entry point into a second table. This latter table contains a series of pointers followed by zeros to indicate their ends. Each pointer corresponds to a possible name match to the name starter and guides the subsequent name analysis to other locations within various tables. These latter pointers have two fields, one identifying the type of pointer and the other the table ordinal to which it points (actual pointer values are shown in Table V). Each pointer directs further analysis to a variety of tables depending on the individual case at hand. Four of these cases are directed to the periodic table: element names, elements that can have "ide" suffixes, elements that can have "ic" suffixes, and elements that can have "ous" suffixes. Other cases point to tables of group names (Tables VI and VII), common names (Table VIII), or alternate element names (e.g., wolfram). Finally, one case is dedicated to names that are formed in an unusual manner, for example, "sulfurous" rather than "sulfous". The sequence of pointers associated with a name match can be as short as one (e.g., "alco") or as long as 12 (e.g., "magn").

Table VI. Recognized Groups

acetate	aluminate	aluminosilicate	amide
ammine	ammonium	antimonate	antimonite
aquo	argenate	argenite	arsenate
arsenite	aurate	azide	benzoate
biphosphite	bismuthate	borate	borohydride
cadmate	carbamate	carbide	carbonate
carbonyl	chromate	chromyl	citrate
cobaltate	cobaltinitrite	cobaltite	cyanamide
cyanate	cyanide	decamolybdate	diarsenate
dichromate	dioxide	diphosphate	dihydrogen phosphide
diphosphite	disilicate	disulfate	disulfide
dithionate	dithionite	dithiophosphate	ethoxide
ferricyanide	ferrocyanide	formate	gallate
germanate	hydronium	hydroxide	hypomanganate
hyponitrite	hypophosphate	hypophosphite	hyposulfite
imide	iridate	isocyanate	lactate
malate	maleate	malonate	manganate
manganite	mercurate	metaaluminate	metaarsenite
metaborate	metaniobate	metaphosphate	metatungstate
methoxide	molybdate	molybdyl	monohydrogen phosphide
osmate	neptunyl	nickelate	niobate
nitrate	nitrite	nitrosulfate	nitrosylsulfate
nitroplatinate	nitroprusside	nitrosyl(III)	nitrosyl(IV)
nitryl	octamolybdate	orthotellurate	oxalate
oxohyponitrite	oxyfluoride	pallidate	pallidite
paramolybdate	paraperiodate	peroxyborate	pentaborate
pentathionate	permanganate	peroxide	peroxodisulfate
perrhenate	perruthenate	persulfide	peroxomonosulfate
pertechnetate	phosphate	phosphite	phosphonium
phosphoryl(III)	phosphoryl(V)	picrate	platinate
platinite	plumbate	plutonyl	propionate
pyrophosphite	pyrosulfite	pyrosulfuryl	rhenate
rhodate	ruthenate	salicylate	selenate
selenite	selenocyanate	selenothionate	selenyl
silicate	sorbate	stannate	stearate
succinate	sulfate	sulfite	sulfonyl
sulfuryl	superoxide	tantalate	tartrate
technetate	tellurate	tetraborate	tetragermanate
tetrasilicate	tetrathionate	tetravanadate	thiocyanate
thionate	thionyl	thiophosphate	thiophosphoryl(III)
thioplatinate	thiosulfate	thorate	thiophosphoryl(V)
titanate	triphosphate	trisilicate	trithionate
tungstate	tungstyl	uranate	uranyl
vanadate	vanadyl(IV)	vanadyl(V)	zincate
zirconate			

Table VII. Recognized Groups with Halogen Substitution

xstem + ate	xstem + ite	xstem + oaluminate
xstem + oantimonate	xstem + oantimonite	xstem + oargenate
xstem + oargenite	xstem + oarsenate	xstem + oarsenite
xstem + oaurate	xstem + oberyllate	xstem + obismuthate
xstem + obromate	xstem + ocadmate	xstem + ochromate
xstem + ocobaltate	xstem + ocobaltite	xstem + ogallate
xstem + ogermanate	xstem + oiodate	xstem + oiridate
xstem + omanganate	xstem + omanganite	xstem + omercurate
xstem + omolybdate	xstem + onickelate	xstem + oosmate
xstem + opallidate	xstem + opallidite	xstem + ophosphate
xstem + ophosphite	xstem + oplatinate	xstem + oplatinite
xstem + oplumbate	xstem + orhenate	xstem + oruthenate
xstem + oselenate	xstem + osilicate	xstem + ostannate
xstem + osulfate	xstem + osulfonate	xstem + osyl
xstem + otantalate	xstem + otellurate	xstem + othorate
xstem + otitanate	xstem + otungstate	xstem + ouranate
xstem + yl	xstem + ozirconate	boro + xstem + ide
di + xstem + ophosphite	hexa + xstem + orhodate	hypo + xstem + ite
oxy + xstem + ide	penta + xstem + orhodate	per + xstem + ate
per + xstem + yl		

Table VIII. Recognized Common Names for Chemicals

alcohol	alumina	ammonia	arsine
azoimide	baking soda	beet sugar	benzene
benzoic acid	benzol	beryl	bichrome
bismuthine	bleach	blende	bleaching powder
boric acid	borane	borax	brimstone
bromoform	buckyball	calcia	calcite
calomel	camphor	cane sugar	carbolic acid
carborundum	chalk	charcoal	chili saltpeter
chili niter	chloroform	chrome yellow	corrosive sublimate
cinnabar	corn sugar	corundum	cream of tartar
cuprite	cyanamide	cyanogen	decaborane
deuterium	diamond	diarsine	diazine
diborane	digallane	digermane	diphosphine
disilane	disiloxane	dry ice	emery powder
ethanol	ethane	ethene	ether
ethyl alcohol	ethyne	ferrite	fluoristan
fluorite	fluorspar	fool's gold	formaldehyde
fructose	galena	gallane	germane
glucose	glycerin	grain alcohol	graphite
hematite	hexaborane	hexamine	hydrazine
ice	iodoform	iron black	kieselguhr
lactose	lamp black	laughing gas	lime
limestone	lye	manganite	magnesia
magnesite	magnetite	marble	marsh gas
methane	methyl alcohol	methanol	milk of barium
milk of calcium	milk sugar	muriatic acid	milk of magnesia
niter	nitrous	oil of vitriol	ozone
paris blue	pentaborane	peroxide	phenol
phosgene	phospham	phosphine	picric acid
potassamide	prussian blue	prussic acid	quartz
quick lime	quicksilver	rock salt	rust
salt	saltpeter	sand	silane
silica	silicane	slaked lime	soda ash
steam	stibine	table salt	tetraborane
tetrasilicane	thiocyanogen	trigermane	trisilane
tritium	tungstenite	urea	vermillion
vinegar	vitriol	water	wood alcohol
zincite	zircon	zirconia	

This approach has a number of points of interest. First, care must be taken with prefixes. Difficulties arise because the same prefix can have different meanings in different chemical contexts. For example, the prefix "di" in dichloride means "two of the species" and in diphosphate means "two of the central atom with more associated oxygen". The prefix "thio" means sulfur but has different meanings in thiosulfate and thiocyanate. Halogens can also be part of prefixes if they replace oxygens, such as in chloroplumbate. As much as possible, these cases are handled with a generic negative ligand matching procedure which is familiar with chemicals containing oxygen atoms that can be replaced by halogen, sulfur, or selenium atoms. Table VI shows the groups recognized by the parser, and Table VII identifies those groups for which halogen, sulfur, or selenium substitution can be made. These tables are also accessed by other experts and parsers and serve as an integral part of the declarative knowledge base of the system.

Second, there can be several excellent, even perfect, matches that occur during the parsing process. For example, both "sulfur" and "sulfuric" match "sulfuric acid". Thus, it is not always possible to accept the first match, even if the match is perfect. Furthermore, the match with the highest score may not be the correct choice. For example, when identifying "bicarbonat", the score for "carbonate" is lower than that of "carbon", even though "carbonate" is the correct match. Word position and creation of word fragments are used as additional weighting factors to resolve these situations.

Third, it may be difficult to identify when one name starter ends and the next one begins. The initial letters of the name part that follows may tempt the parser to include them in the name part currently under consideration (e.g., "zinc nitride" could perhaps be a misspelled form of "zincite" plus the extra letters "ride"). Spelling mistakes compound this problem. Finally, care must be taken not to meld descriptor words such as "ion", "acid", or "alloy" into the name.

Formula Assignment. The first task of formula assignment is to break the name up into parts and set flags based on the structure of the name. For example, if descriptors like "gas", "ion", "acid", or "alloy" are present, they are stripped from the name and the appropriate flags are set. The presence of two such descriptors is a fatal error that terminates the assignment process. If the chemical is an acid, for subsequent decoding it is renamed as if it were a hydrogen salt; e.g., sulfuric acid becomes "hydrogen sulfate". Special attention is required for acids whose names already contain hydrogen (e.g., hydrosulfuric acid), so that hydrogen is not entered twice. If the chemical contains metals, any given oxidation states are entered into the table with the appropriate name parts.

The next steps are to split off any ligands associated with a cation, to note these as ligands, to classify any other parts as cations or anions, and to reorder the parts, placing the cations first. Each part is then assigned a formula and a charge or oxidation state (which may be tentative). Assuming the parts were properly separated and any entry errors corrected by the parser, each part can now be converted to a formula independently of the neighboring parts.

The possibility that the name part represents a group (see Tables VI and VII) is now considered. A routine is called that attempts to match the name part with possible chemical groups. If a match is sucessful, the group's formula and charge are returned and added to the information known about this name part. Special cases, such as the prefix "bi" which requires hydrogen in the formula, or the prefix "di" in dioxide which fixes the number of the oxygen atoms at two, are now considered. Whether the group is a cation, an anion, or a ligand is also ascertained.

If the part is not a recognizable group, an effort is made to process the name part as an element or simple ion. To accomplish this, the name part is matched to the stems of the elements' names. If this match is successful, the name part is split into its stem and suffix. If the match is unsuccessful, a list of cationic groups (e.g., "nitryl") and alternate element names (e.g., "deuterium") is checked. If a match is now found, the appropriate splitting of stem and suffix is performed. The name part is then processed in one of four ways, depending on whether the suffix (1) matches that of an element and there is only one name part present (e.g., "helium"), (2) matches the element or cationic group identified or contains a suffix of "ide" (e.g., "mercury(I) ion" or "nitride"), (3) is "ic" or "ous" (e.g., "ferric" or "cuprous"), or (4) is "ate" or "ite" (e.g., "chlorate" or "chlorite"). The processing for each of these four cases will now be described.

The first case is easily handled: an oxidation number of zero is assigned, diatomic and polyatomic elements are given appropriate subscripts, and appropriate flags signify that the identification is complete.

In the second case (ions with only one element), the chemical symbol for the element is added to what is known about the name part, with due care being taken for special cases such as the "mercury(I) ion". If the element has only one possible oxidation number or charge, it is affixed and marked as established. If it has more than one, the most common oxidation number or charge is selected and the flag is set as not established. If a numeric prefix was used with the element, the subscript of the element is set and marked as established; otherwise, it is set to one and marked as unestablished. For cations, the number of associated ligands, if any, is checked for consistency with the coordination number and added to the formula if appropriate.

In the third case ("ic/ous"), the procedure is similar to the previous one, but with a few notable differences. First, there is no anionic case to handle, but it is necessary to check whether the element with the "ic" or "ous" suffix is chemically feasible. If the element makes chemical sense, and if an inappropriate oxidation state has not been attached, the symbol and charge are added to the known information and marked as established. If the species does not make chemical sense, as in the case of "ferric(II)", a fatal error condition is set which will eventually be reported to the caller with the solution trace. Any numeric prefixes and ligands present are processed as described in the preceding paragraph.

In the final case ("ate/ite"), the procedure is used only to process groups that are not in the group tables. (Groups in the tables were discussed earlier.) Although the group tables are extensive, it is impractical to make them exhaustive. If an anionic group is detected that is not part of the tables, this routine attempts to develop a formula for it. In the element table, an array indicates what the "ate" and "ite" states for the element would be. On the basis of this information, an initial formula is derived. Rules are then employed for prefixes such as "ortho", "per", "hypo", "pyro", "thio", and "meta" to determine the refined formula and its associated charge that will be subsequently saved as established information. Finally, hydrogen is added if the prefix "bi" is present.

Should all of these attempts fail, a final effort is made to strip numerical prefixes, formation prefixes (e.g., "thio") and ligands from the name part. If the name part can be reduced to a more primitive form, it is again analyzed with the entire series of routines above with the prefix and ligand flags set. Only if an unnameable root string is found will the model accept defeat.

After a formula has been established for each name part, the formula parts are combined to form a legal formula. Each formula part has an associated oxidation number or charge (the oxidation number or charge defaults to the most common value when necessary) and count (default is 1). If one or both of these values have been established from the name part, this value cannot be changed. If the word "alloy" is part of the name and all parts are metallic, the counts and charges are established directly from the name and the formula written. If the word "ion" is part of the original name, the counts and charges are also established directly from the name and a net charge is attached. In all cases, except for ions which must possess a net charge, the model systematically adjusts the unestablished charges and counts to balance the formula to possess no net charge. If the model is unsuccessful, it reports failure; otherwise, the identified charges and counts are established, and the formula is written. Two common causes of failure occur when either the entered name is overconstrained (e.g., "triammonium sulfate") or it is underconstrained (e.g., "sodium potassium phosphate").

To further aid the user, the model produces error messages when it cannot understand the input or cannot complete the formula writing process (Table IX). In addition, it produces a trace of its actions so that the program calling the model

Table IX. Name-to-Formula Error Messages

1.	The character (char) should not be used here.
2.	You incorrectly used the parenthesis in (string).
3.	CHEMPROF doesn't understand (string).
4.	You need to specify the acid.
5.	CHEMPROF doesn't recognize that acid.
6.	Is that the correct charge on (string)?
7.	CHEMPROF doesn't think such a compound can exist.
8.	All your components are cations.
9.	You used an improper cation suffix in (string).
10.	You used an improper anion suffix in (string).
11.	CHEMPROF doesn't recognize the species (string).
12.	All your components are anions.
13.	You have entered the wrong number of ligands.
14.	There are too many words in the your chemical.
15.	The chemical proportions seem incorrect.
16.	Try this again! (*Response to apparent gibberish*)
17.	The name given is inadequate to write a formula.
18.	Words have been incorrectly mixed together.
19.	More information is needed to specify this ion.
20.	I'm not sure about (string). Is this an acid?

Table X. Trace from the Name-to-Formula Expert

1	I0	calcium bisulifte	input name
2	T0	6	chemical type = IONIC
3	W3	sulfite	misspelled part of name
4	W4	sulfite	corrected part of name
5	I1	calcium bisulfite	name after preliminary correction
6	R0	calcium bisulfite	clean name
7	R1	calcium	name of part
8	R1	bisulfite	name of part
9	D3	calcium	name of part being evaluated
10	D4	+	cationic species
11	D3	bisulfite	name of part being evaluated
12	D4	-	anionic species
13	E1	calcium	name of part being converted
14	E2	+2	as cation, charge is +2
15	E5	1	no. of copies present so far
16	E6	Ca	chemical symbol for species
17	E9	Ca}+}2	species representation
18	S1	bisulfite	name of part being converted
19	X1	hydrogen sulfite	preferred name of part
20	S2	HSO{3	formula for group identified
21	S3	−1	as anion, charge is −1
22	E5	1	no. of copies present so far
23	E9	HSO{3}−	species representation
24	B3	Ca	representation for writing formula
25	B4	1	balance coefficient
26	B3	HSO{3	representation for writing formula
27	B4	2	balance coefficient
28	N3	Ca(HSO{3){2	final formula of chemical
29	ZZ	0	end of trace

```
PCl5 is the chemical formula for which compound?

                    Phosphorous chloride

        You misspelled phosphorus. You made a prefix error
        for pentachloride. The correct answer is phosphorus
        pentachloride.
```

Figure 2. Computer request for a student to name a formula, the student's response (underlined), and the teaching logic's efforts to use the errors discovered by the name-recognition expert to remediate the student.

can determine how decisions were made (Table X). Efforts made to correct the entered name are also noted in the trace (note steps 3 and 4).

APPLICATION OF THE EXPERTS

In the CHEMPROF project, we extensively used the expert built on the name-recognition model in teaching students how to name chemicals, given their formulas. Students were given a series of chemical formulas to name, progressing from elements through compounds containing polyatomic ions. The names they proposed and the known names of the chemicals, with possible alternates, were supplied to the name-recognition expert. The expert compared the two and returned bit maps with any errors noted. The teaching logic used this information to produce a description for the student of any errors made. An example is shown in Figure 2 where the student names PCl_5 as "phosphorous chloride".

The name-to-formula expert was used to give the students step-by-step instruction on how to write chemical formulas from chemical names. Two approaches were used. In the first, the computer allows the student to choose a class of chemicals and then provides the name of a chemical from this class. The student then attempts to type in its chemical formula. The teaching logic, relying on the expert solution, gives the student feedback about his or her answer. In the second, the student chooses a chemical name and enters it, and the computer merely returns the corresponding chemical formula.

```
How CHEMPROF determines the formula for calcium bisulifte:

  Recognize that calcium bisulfite is an ionic
  compound because it contains a metal and nonmetal
  (or a metalloid or polyatomic ion).

  The chemical symbol for calcium is Ca.
    The charge on calcium is +2 (Ca^2+).
  You've misspelled bisulfite as sulifte.
  The preferred name for bisulfite is hydrogen sulfite.
  The chemical formula for hydrogen sulfite is HSO_3.
    The charge on hydrogen sulfite is -1: (HSO_3^-).
  For ionic compounds, the sum of the charges must total ZERO.
  This occurs with 1 Ca^2+ and 2 HSO_3^-.
                  1(+2) + 2(-1) = 0
  Therefore the formula for calcium bisulfite is Ca(HSO_3)_2.
```

Figure 3. Explanation of a name-to-formula conversion that CHEMPROF gives to the user based on the trace shown in Table X. It explains the various steps that the expert applied. (On the screen, the boldface words are in color.)

In both cases, the name-to-formula expert returns a trace, as shown in Table X for the incorrect entry "calcium bisulifte". A solver works through the trace to produce the step-by-step instructions needed to convert the name into the formula, and these appear in Figure 3. Since the name-to-formula expert is programmed to overcome user input errors, the fact that the name was converted to a formula does not imply the name originally entered by the student was correct. The student can access an explanation of the answer using a help key. This gives the student the ability to have the computer explain how to name any particular chemical of interest.

DISCUSSION

Converting inorganic names into formulas would seem to be a straightforward process, but it is seriously affected by both spelling and spacing errors. In practice, spacing errors occur infrequently, but the expert models must still be able to handle them if they occur. This versatility adds significantly to the complexity of the models' structure. The fact that the entered name parts may not conform to an actual name part because of a typing error makes the interpretation much more difficult. The approach of removing all spaces to eliminate the variableness of this problem would work very well if it could be guaranteed that a spelling and spacing error would not occur simultaneously. With spelling errors and potentially missing letters, redividing the words can be particularly troublesome. Without a good division, the matching algorithm may compute an incorrect score and an obvious name match may be missed.

The models rely on both a general spell checking subroutine and a spelling checker that uses chemical rules to interpret names. Both spelling procedures are sensitive to letters near the beginning and ending of the word. The former has an inorganic chemical name dictionary to improve the likelihood that an acceptable word will be obtained on a near match. The latter is designed to circumvent problems when oxygen atoms are replaced by other nonmetal atoms (i.e., sulfur, selenium, and halogen substituted compounds). Thus, the two checkers complement each other because one is governed solely by mechanical similarities and the other is much more domain oriented. The word recognition of the general checker may be improved by enlarging its dictionary of chemical parts, but these extra entries would also increase the probability of an incorrect suggestion for a name part on a near match. Optimization studies need to be done on this problem.

When the name-to-formula expert was first envisioned, the major thrust was to make it as general as possible. Initially, it was hoped that rules could be developed to allow the expert to decipher almost any chemical group from the periodic table and the chemical rules of oxidation states and structures. Unfortunately, two things made this approach impractical. First, spelling errors rendered such an approach unfeasible because there was no pattern to use to correct spelling errors with a knowledge of the possible chemical names. Therefore a table of legal chemical names was developed to preface the efforts of this routine. Second, the degree of naming consistency was far less than first assumed. For example, while "thiocarbonate" has all its oxygens replaced with sulfur atoms, "thiosulfate" has only one oxygen atom replaced by sulfur. This meant an extensive table of exceptions would have been required for the names that were processed by this mechanism. The general name builder still is part of the expert, but it is now the method of last resort, rather than the primary method of analysis. As long as a large number of names is needed for spelling correction purposes, they might as well have associated formulas so that exact matches can be reported back immediately without calling the builder. On the other hand, those cases that are more complex versions of simpler names (e.g., "thiopermanganate") must be left for the general name builder to decode if possible.

Even though the final formula writer is designed to give legal formulas for chemicals and the name-to-formula expert is sensitive to the chemical sense of the compounds, it is still possible to produce a formula from a name that makes no chemical sense. For instance, even though "iron(II) permanganate" is a legal formula and makes chemical sense from the vantage of balanced charges (i.e., its chemical syntax is correct), it is nonetheless a nonsensical chemical compound.

The name-to-formula expert was validated on a set of 157 chemicals whose names were chosen to cover the domain of names being used in undergraduate inorganic chemistry. Most of the names were common, although special cases such as "mercury(I) ion" were also included to check the expert's ability. Some of the names were intentionally incorrect to test the expert's correction powers and ability to reject chemically impossible names (e.g., "dicalcium phosphate").

The bit map concept for the name-recognition expert is intended to allow the return of as many as possible of the detected errors to the calling program. If the calling program is a teaching program, it needs all the information available about the student's performance to provide an appropriate remediation. If the program is attempting to recognize the desired chemical from a group of similar chemicals, the number and type of errors will indicate the best match. Since the name-recognition expert relies on a complex set of rules which are performed multiple times in an iterative manner, a trace of what was done would be too complex for the calling program to use effectively. The goal is to determine what was wrong with the user input and not how the error was detected.

The name-recognition expert has been tested with a series of names that contain spelling, spacing, naming logic, and nonsensical errors. It was refined with the idea that, while it might not identify all the errors, it ought to give a reasonable explanation of what was wrong with the entered name. Our definition of reasonableness is defined as "being of use to the user in recognizing at least one error" and "not containing any information which is obviously false."

Future work in the area of formula-name interconversion is needed to reduce the conversion methodology into compact routines that can be added with minimal effort to programs where the recognition and interepretation of chemical names and formulas are essential to the user in the laboratory.

ACKNOWLEDGMENT

The authors 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) Pierce, T. H., Hohne, B. A., Eds. *Artificial Intelligence Applications in Chemistry*; American Chemical Society: Washington, D.C. 1986.
(2) Hohen, B. A., Pierce, T. H., Eds. *Expert System Applications in Chemistry*; American Chemical Society: Washington, D.C., 1989.
(3) Eggert, A. A.; Jacob, A. T.; Middlecamp, C. H. Converting Chemical Formulas to Names: An Expert Strategy. *J. Chem. Inf. Comput. Sci.* **1992**, *32*, 223–237.
(4) Eggert, A. A.; Middlecamp, C. H.; Kean, E. CHEMPROF-A Tutor for General Chemistry. *J. Art. Intell. Educ.* **1990**, *2* (1), 47–62.
(5) Eggert, A. A.; Middlecamp, C. H.; Kean, E. CHEMPROF-An Intelligent Tutor for General Chemistry. *J. Chem. Educ.* **1991**, *68*, 403–407.
(6) Leigh, G. J., Ed. *Nomenclature for Inorganic Chemistry: Recommendations 1990*; Blackwell Scientific Publications: Oxford, England, 1990.
(7) Barr, A.; Feigenbaum, F. A. Understanding Natural Language. *The Handbook of Artificial Intelligence*; William Kaufmann, Inc.: Los Altos, CA, 1981; pp 223-321.
(8) Tomita, M. *Efficient Parsing for Natural Language*; Kluwer Academic Publishers: Boston, 1986.
(9) Allen, J. *Natural Language Understanding*; Benjamin/Cummings: Reading, MA, 1987.
(10) King, M., Ed. *Parsing Natural Language*; Academic Press: New York, 1983.
(11) Brown, J. S.; Burton, R. R. Diagnostic Models for Procedural Bugs in Basic Mathematical Skills. *Cognit. Sci.* **1978**, *2*, 155–192.