Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 2. Development of a Formal Grammar

D. I. COOKE-FOX, G. H. KIRBY,* and J. D. RAYNER

Department of Computer Science, University of Hull, Hull HU6 7RX, England

Received May 4, 1988

A context-free, phrase structure grammar is presented for IUPAC systematic organic chemical nomenclature. Although this grammar is incomplete, it describes many of the syntactic constructions used to name hydrocarbons, acids, alcohols, aldehydes, ketones, and ethers. Some conjunctive and radicofunctional nomenclature is covered in addition to substitutive nomenclature. A selection of trivial names and terms, allowed by the IUPAC rules, is included, leading to trivial and semisystematic forms of nomenclature. The building of the grammar from the informally expressed IUPAC rules and examples is described.

INTRODUCTION

There is currently no complete, context-free, phrase structure grammar that fully describes all the possible syntactic constructions allowed by IUPAC systematic organic chemical nomenclature. Only as outlined in part 1 of this series[1] have formal grammars for various subsets of that nomenclature been devised. The possible syntactic constructions of IUPAC organic chemical nomenclature as a whole are only loosely defined, by a number of informally expressed rules and examples in the IUPAC *Nomenclature of Organic Chemistry*,[2] universally referred to as the "Blue Book". An example of the way the rules are expressed in the Blue Book is Rule A-3.1, which states

"Unsaturated unbranched acyclic hydrocarbons having one double bond are named by replacing the ending "-ane" of the name of the corresponding saturated hydrocarbon with the ending "-ene". If there are two or more double bonds, the ending will be "-adiene", "-atriene", etc.
Examples:
2-Hexene
$CH_3—CH_2—CH_2—CH_2═CH_2—CH_3$
1,4-Hexadiene
$CH_3—CH═CH—CH_2—CH═CH_2$"

The rules in the Blue Book are not in a form that may be directly processed by a computer for automatic name recognition. A context-free grammar that describes the same syntactic constructs as given in the Blue Book must be derived and processed by computer to produce a chemical name recognizer. This is the first stage in the grammar-based approach to the computer translation of chemical nomenclature, introduced in part 1[1] of this series. The present paper describes how the development of such a grammar has been approached. Part 3[3] describes how the grammar is processed by a parser generator program to produce a recognizer that is used to parse names.

CHOOSING THE CHEMICAL NAME FRAGMENTS

The chemical name fragments appropriate to a context-free grammar are not easily chosen from a name, except for the punctuation symbols such as commas, hyphens, and parentheses. The choice of alphabetic fragments is governed by the meaning of the fragment. A chemical name fragment that has meaning is called a morpheme.

The chemical name pentane has two valid morphemes, "pent" and "ane", which indicate respectively the presence of a chain of five carbon atoms and the information that the chain is the parent structure and is saturated. "pe", "nta", and "ne", although fragments of the name pentane, are not morphemes because they cannot be assigned meaning. The extraction of morphemes from chemical nomenclature was described by Garfield in 1961 (republished in 1985[4]), a long time ago in relation to the history of computer science. However, apart from isolated applications to a few specific classes of compound, Garfield's pioneering work was not followed up for compounds in general until the work commenced in this department nearly 20 years later.

The terminal symbols of a context-free grammar to describe the language of chemical nomenclature have been derived by a study of the constructs allowed in the appropriate sections of the Blue Book. All the name fragments that perform the same function within a chemical name may be assigned to one symbol, which becomes the terminal symbol of the grammar for that fragment class. For example, the four name fragments "meth", "eth", "prop", and "but" can all be followed by the fragment "ane" to form the names of the four simplest straight-chain hydrocarbons. These four name fragments were therefore assigned to the single terminal symbol "aliph-root", while the single name fragment "ane" is represented by the terminal symbol "ane-mark".

Once the terminal symbols of the grammar were isolated, production rules were developed to govern the possible combinations of the terminal and nonterminal symbols to form valid chemical names. Terminal symbols correspond to parts of speech (noun, verb, aliph-root, etc.), while nonterminal symbols represent phrases, for example, "locant-sequence" or "substituent", each of which is made up of a number of associated fragments.

To aid computer analysis of the meaning, the whole of the name is read into the computer and processed from the right to the left, so the grammars are written as if reading from the rear of the name forward. However, multicharacter name fragments are included in the normal way with the characters of each fragment being reversed by the parser generator before they are assembled into a chemical name fragment dictionary. All the example grammars given are in the format now required by the SLR parser generator.[5] This program is an implementation of a simple LR parsing algorithm described by Aho and Ullman.[6]

The overall form of a grammar can be quickly illustrated by reference to a simplified locant sequence as in Figure 1. Four terminal symbols represent the letters, digits, and necessary punctuation, given that a series of digits can also be recognized as a number. Alternatives for a given grammar symbol (terminal or nonterminal) are given in a list separated by commas and terminated with a semicolon, while symbol sequences, as in any one production rule alternative for a nonterminal symbol, are shown in plain sequence. Thus, three nonterminals represent, first, an individual locant (formed from a number either with or without a letter, and for reading right

```
TERMINALS

    number = '0','1','2',...'9';
    letter = 'a','b','c',...'z';
    comma  = ',';
    hyphen = '-';

RULES

    locant = number, letter number;
    locant-sequence = comma locant locant-sequence, $;
    locant-part = hyphen locant locant-sequence hyphen, $;

ROOTSYMBOL

    locant-part
```

Figure 1. Illustration of grammar form.

to left), second, a sequence of locants separated by commas, and, finally, the complete construct within enclosing hyphens.

The "$" symbol, in the second rules for locant-sequence and locant-part, indicates that these nonterminals may be "empty". That is, a name may exist without a locant-part; a locant-part may exist with only one locant and no locant-sequence. A locant-part occurring at the left end of a name would require no left-hand hyphen, so the example as given is not to be taken as rigorous, merely as illustrative. The "rootsymbol", otherwise known as the "distinguished" nonterminal symbol, is the nonterminal corresponding to a complete sentence of the language represented by the grammar.

BUILDING A GRAMMAR FOR HYDROCARBON NOMENCLATURE

There are formally two ways in which a grammar may be constructed, top down and bottom up. In the top-down approach, the grammar designer takes the distinguished symbol of the grammar and writes rules to expand it into strings of terminal and nonterminal symbols. Further rules are then constructed to expand each of the nonterminal symbols similarly, until only terminal symbols remain. These may then be equated to the vocabulary of the language. For example, the distinguished symbol "name" may be expanded to either of the nonterminals "organic-name" or "inorganic-name". The nonterminal symbol "organic-name" can then be expanded to and beyond "aliph-name", "acid-name", "alcohol-name", etc., each of which represents a different class of organic nomenclature. A grammar formed by top-down development is called a prescriptive grammar, and this approach enables the designer to concentrate on functional subsets of the language as the grammar is built incrementally.

In the bottom-up approach, the designer starts by listing examples of all possible constructs of the nomenclature under study, to identify relevant name fragments and assign them to terminal symbols. Rules are then devised to associate these terminal symbols with nonterminal symbols, until the distinguished symbol is reached. A grammar formed by this bottom-up approach is called a descriptive grammar since it is based on a description of actual examples of usage.

The grammar development process adopted in this work represents a combination of both approaches. Functional areas of the nomenclature have been added over time in a top-down fashion by adding alternative rules for the distinguished symbol or closely related, high-level, nonterminals. Within each functional area, a bottom-up approach has been applied to identify new terminal symbols relevant to the scope of the alternative rules.

The two possible approaches described above are illustrated separately through the development of two hydrocarbon grammars. These also bring out the difficulties involved in grammar design. The first is a development of the grammar devised by Rayner[7] with minor modifications. It is a modular grammar assuming the European convention of placing locants

```
TERMINALS

a-mark         = a ;
e-mark         = e ;

aliph-root     = meth , eth , prop , but;

val3to9        = tri , tetra , penta , hexa ,
                 hepta , octa , nona ;

gmult-cont     = pent , hex , hept , oct , non ;

val10to12      = dec , undec , dodec ;

dec-mark       = dec ;

an-mark        = an ;
en-mark        = en ;
yn-mark        = yn ;
aliph-suffix   = e ;
cyclo-mark     = cyclo ;

g-mult         = mono , di , tri , tetra ;
gmult-mark     = a ;

number         = '0' , '1' , '2' , '3' , '4' ,
                 '5' , '6' , '7' , '8' , '9' ;

comma          = ',' ;
hyphen         = '-' ;
```

Figure 2. Terminal symbols for the example hydrocarbon grammars.

directly before the functional group, modification, or substituent to which they refer. The second approach developed by Cooke-Fox[8] was designed to recognize all the possible constructs allowed by the Blue Book to name a simple hydrocarbon. The same terminal symbols were used in each case and are given in Figure 2. The terminal symbols are underlined in Figures 3 and 4, where the nonterminals are easily seen as a column on the left of the list of rules. Once again, "$" indicates a null alternative.

In considering the following discussion, it may be helpful to consider that a context-free grammar, once developed, can be viewed both as a top-down generator of sentences for the language and as a bottom-up recognizer. The parsing of a name, though, is a bottom-up process where, by reference to the fragment dictionary, the input name is broken into fragments that are each assigned a terminal symbol class. The terminal symbols are repeatedly reduced to a string of terminals and nonterminals by application of the appropriate rules, until the rootsymbol is encountered when all of the name has been processed.

First Example Hydrocarbon Grammar. The first example hydrocarbon grammar was developed with the constraint that a hydrocarbon name consisted of a preceding optional "cyclo-mark" (for monocyclic hydrocarbons), followed by an "aliph-root" terminal or a multiplying term, followed by a saturation sequence. The grammar rules are given in Figure 3. The rootsymbol of the grammar is "name", which has the single nonterminal symbol "org-name" as the right part of the first rule production. This permits easy expansion of the grammar to recognize inorganic names by allowing the production rule alternative "inorg-name". "org-name" also has a single nonterminal as the right part of the production, whereas the more extensive grammar we have developed[9] allows other rule alternatives for alcohols, acids, aldehydes, ketones, and ethers.

The nonterminal "aliph-name" has a right part consisting of a terminal symbol "aliph-suffix" followed by a nonterminal "aliph-part". "aliph-suffix" represents the single letter "e", which is appended to the aliphatic name to make a simple hydrocarbon. This "e" may be replaced by a functional group to form a different class of compounds. For example, if the

```
RULES

name            = org-name ;
org-name        = aliph-name ;
aliph-name      = aliph-suffix aliph-part;
aliph-part      = aliph-parent ;
aliph-parent    = chain-parent ,
                  ring-parent ;

chain-parent    = sat-seq  aliph-stem ;
ring-parent     = sat-seq aliph-stem cyclo-mark ;

aliph-stem      = aliph-chain ;
aliph-chain     = aliph-root ,
                  mult-value ;

sat-seq         = yn-sat yn-ext ,
                  en-sat en-ext ,
                  an-sat ;

yn-sat          = yn-mark   gm-part  loc-part ;

yn-ext          = enyn-ext en-sat  en-ext ,
                  a-mark ,
                  $ ;

enyn-ext        = e-mark , $ ;

en-sat          = en-mark  gm-part   loc-part ;

en-ext          = a-mark , $ ;

an-sat          = an-mark;

gm-part         = g-mult ,
                  gmult-mark mult-value ,
                  $;

mult-value      = gmult-cont ,
                  val10to12 ,
                  dec-mark val3to9 ;

loc-part        = hyphen  locant  loc-seq  hyphen , $ ;

loc-seq         = comma  locant  loc-seq , $ ;

locant          = number ;
```

Figure 3. First example hydrocarbon grammar.

"e" in "methane" is replaced by "ol", the result is an alcohol name, methanol. So for simple alcohols all that is needed is an additional rule of the form

alcohol-name = alcohol-mark aliph-part ;

In this example grammar, the parent may be a simple aliphatic chain or a single-ringed alicyclic. This is handled by the rule

aliph-parent = chain-parent , ring-parent ;

The only difference here between the right parts of the rule productions "chain-parent" and "ring-parent" is the occurrence of the "cyclo-mark" terminal symbol, occurring to the right of the "ring-parent" production rule. In more complex cases these two productions may bear little resemblance to each other. Each of the production rules in this case is started by the nonterminal "sat-seq" that controls the unsaturation part of the name.

The "sat-seq" production rule has three rule alternatives that allow for the three possible endings to the name. The simplest of these, "an-sat", is a production rule whose right part is a single terminal symbol "an-mark", having a 1:1 mapping onto the morpheme "an". The production rule

an-sat = an-mark ;

is hence superfluous to the syntactic definition of the language, since the same language could be described by replacing right-part occurrences of the nonterminal "an-sat" with the terminal symbol "an-mark". However, the "an-sat" production rule is required to control the later extraction of semantic information and the construction of a numerical representation of the chemical name.[3]

The rule alternative comprising the nonterminals "en-sat en-ext" recognizes names that contain one or more double bonds. The "en-sat" production rule first indicates the presence of the terminal symbol "en-mark". This may have an optional multiplying term and optional locants, which if present are separated by commas and preceded and followed by hyphens. Second, the "en-ext" production rule has a right part with two rule alternatives, "a-mark" or "$" (the null string). The latter alternative allows the one rule to process both multiple unsaturations where an "a" is required and single unsaturations where it is not, for example, deca-3,5-diene in contrast to dec-2-ene.

Notice that the "a" does not have to be present to allow the name to be parsed as syntactically correct according to the grammar: the "a" has no meaning within the name, being present for ease of pronunciation. The appropriate presence or absence of such letters is checked later in the semantic processing.[3]

In the same way, the syntax rules allow for any number of locants, including no locants, to precede a multiplying term, and the correlation of the number of locants with the multiplying term is not checked during the syntax analysis. It is possible syntactically to check this correlation, but it would require the separation of the name fragments in the "mult" syntactic terminal symbol class, so that each multiplying term has its own terminal symbol, and a number of separate production rules would then be required to recognize each of the new terminal symbols in the same context. For the sake of generalization and compactness of the syntax rules, the correlation of the number of locants and the value of the multiplying term is checked later in the semantic phase.

The third rule alternative of "sat-seq" is for the case where triple bonds are present. This has, as its right part, the nonterminals "yn-sat yn-ext". The nonterminal "yn-sat" expands to the terminal symbol "yn-mark" preceded by an optional multiplying term and a locant sequence. The nonterminal "yn-ext" has three rule alternatives, for the cases where the triple bond is preceded by one or more double bonds; where there is more than one triple bond and an "a" is inserted between the parent and the unsaturation part; and where there is only a single triple bond. These rules are constructed analogously to those for "en-sat" and "en-ext".

The first example hydrocarbon grammar is simple to build and easy to interpret but does not rigorously generate all the possible valid constructs described by the Blue Book. From this grammar a sample of the valid names that would be accepted is

pentane
dec-1-ene
deca-1,3-diene
dodeca-1,3,5-triene-7,9-diyne
but-1-yne

but it will also "correctly" parse certain names that are syntactically invalid according to the IUPAC nomenclature rules, for example, dec-2,4-dien-6,8-triyne, which has the following faults: (1) An "a" is missing after "dec". Our grammar rules indicate that it is optional. (2) An "e" is missing after "dien". Again our grammar rules say it is optional. (3) The number of locants disagrees with the multiplying term for the triple unsaturations.

Each of these errors is picked up by later semantic routines[3] in which a check is made if an "a" is present, to see if a multiplying term follows it, and, if not, a very specific and appropriate error message is given. A check is also made on

```
RULES

name            = org-name ;
org-name        = aliph-name ;
aliph-name      = aliph-suffix  aliph-parent ;

aliph-parent    = an-mark  aliph-part ,
                  en-mark  en-ext ,
                  yn-mark  yn-ext ;

en-ext          = loc-part  aliph-part ,
                  aliph-part  pos-loc-part ,
                  gm-part gm-en-ext ;

gm-en-ext       = loc-part  a-mark  aliph-part ,
                  a-mark  aliph-part  pos-loc-part ;

yn-ext          = loc-part  loc-yn-ext ,
                  aliph-part  pos-loc-part ,
                  gm-part  gm-yn-ext ;

loc-yn-ext      = aliph-part ,
                  en-mark  en-ext ;

gm-yn-ext       = loc-part  loc-gm-yn-ext ,
                  a-mark  aliph-part  pos-loc-part ;

loc-gm-yn-ext = a mark  aliph-part ,
                  e-mark  en-mark  en-ext ;

aliph-part      = chain-parent ,
                  ring-parent ;

chain-parent    = aliph-stem ;
ring-parent     = aliph-stem cyclo-mark ;
aliph-stem      = aliph-chain ;
aliph-chain     = aliph-root ,
                  mult-value ;

gm-part         = g-mult ,
                  gmult-mark mult-value ;

mult-value      = gmult-cont ,
                  val10to12 ,
                  dec-mark  val3to9 ;

loc-part        = hyphen  locant  loc-seq  hy-part ;
pos-loc-part    = loc-part , $ ;

loc-seq         = comma  locant  loc-seq , $ ;
locant          = number ;

hy-part         = hyphen, $;
```

Figure 4. Second example hydrocarbon grammar.

```
decane                     = an mark  aliph-stem

dec-2-ene                  = en-mark  loc-part aliph-stem
2-decene                   = en-mark  aliph-stem loc-part

dec-2-yne                  = yn-mark  loc-part  aliph-stem
2-decyne                   = yn-mark  aliph-stem  loc-part

deca-1,3-diene             = en-mark  gm-mult  loc-part a-mark
                                aliph-stem
1,3-decadiene              = en-mark  gm-mult  a-mark  aliph-stem
                                loc-part

deca-1,3-diyne            = yn-mark  gm-mult  loc-part  a-mark
                               aliph-stem
1,3-decadiyne             = yn-mark  gm-mult  a-mark  aliph-stem
                                loc-part

dec-1-en-3-yne            = yn-mark  loc-part  en-mark  loc-part
                               aliph-stem
1-decen-3-yne             = yn-mark  loc-part  en-mark  aliph-stem
                               loc-part

deca-1,5-dien-3-yne       = yn-mark  loc-part  en-mark  gm-mult
                               loc-part  a-mark  aliph-stem
1,5-decadien-3-yne        = yn-mark  loc-part  en-mark  gm-mult
                               a-mark  aliph-stem  loc-part

dec-1-ene-3,5-diyne       = yn-mark  gm-mult  loc-part  e-mark
                               en-mark  loc-part  aliph-stem
1-decene-3,5-diyne         = yn-mark  gm-mult  loc-part  e-mark
                               en-mark  aliph-stem  loc-part

deca-1,7-diene-3,5-diyne = yn-mark  gm-mult  loc-part  e-mark
                               en-mark  gm-mult  loc-part  a-mark
                               aliph-stem
1,7-decadiene-3,5-diyne  = yn-mark  gm-mult  loc-part  e-mark
                               en-mark  gm-mult  a-mark  aliph-stem
                               loc-part
```

Figure 5. Names of some simple hydrocarbons with their terminal symbols.

the match of the number of locants with any multiplying term present.

Second Example Hydrocarbon Grammar. The first example hydrocarbon grammar is further limited by the omission of the commonly used alternative naming practice exemplified by 1,2-decadiene, with locants before the parent. In producing the second example hydrocarbon grammar, given in Figure 4, all the possible alternatives for the simple hydrocarbon subset were written out according to Rule A-3 in the Blue Book. The fragments of these names were then equated with their terminal symbols as illustrated in Figure 5, and the bottom-up approach was applied.

Names in Figure 5 have as their first symbol either an "an-mark", an "en-mark", or an "yn-mark". The production rule for "aliph-name" was therefore chosen as

aliph-name = an-mark aliph-stem ,
en-mark en-ext ,
yn-mark yn-ext ;

The nonterminal symbol "en-ext" represents four alternative strings of terminal symbols (which occur also within the sequences beginning "yn-mark", as underlined in Figure 5):

loc-part aliph-stem
aliph-stem loc-part
gm-mult loc-part a-mark aliph-stem
gm-mult a-mark aliph-stem loc-part

In amending these strings to become the right part of a production rule whose left part is "en-ext", two factors need to be considered. First, there are occasions in naming an unsaturated ring or chain when there is no need to state the locant at which the double or triple bond is situated. Such a case is ethene, where the double bond can only be in one place, or cyclopentene, where all the positions are equal and so a locant need not be stated. In these cases the locant is assumed to be 1 unless there is something else within the structure that would indicate other default settings.

In the first example hydrocarbon grammar, the missing locants are handled by allowing the locants to be optional at all times, the nonterminal symbol "loc-part" having a right part of either a locant sequence or the null string. In the second example hydrocarbon grammar, that is not possible because if "loc-part" were able to be replaced by the null string, then "en-ext" could be expanded to the nonterminal "aliph-part" by either the first or the second rule alternative. The SLR parser generator detects such a situation as an ambiguity in the grammar for which it is not able to produce parsing tables.

The use of a null right part allows greater flexibility to the grammar creator in writing the rules, but it does at times lead to ambiguities that should and can be avoided. In the above case "loc-part" remains with no null rule alternative, and another rule production is introduced, "pos-loc-part", which has two right-part rule alternatives, "loc-part" or "$".

When locants can occur at the beginning of the name, there should be no hyphen at the start of the locant sequence, whereas this hyphen must be present for locant sequences within the name. Thus, in the second grammar, the rule for "loc-part" is amended to allow the corresponding hyphen to be absent by use of the new rule for "hy-part". However, this still allows a hyphen to be present, and it further allows the hyphen to be absent in a locant sequence within the name.

A grammar, to be handled by SLR, should generally not have duplicated terminal or nonterminal symbols as the first symbol on the right part of multiple rule alternatives. In the production rule for "en-ext" given above, the nonterminal symbol "gm-part" starts the right part of two rule alternatives. To obtain an SLR grammar from this grammar, the strings following "gm-part" are replaced by a nonterminal symbol "gm-en-ext". The "en-mark" production rule becomes

```
en-ext = loc-part aliph-stem,
         aliph-stem pos-loc-part ,
         gm-part gm-en-ext ;
```

The rule "gm-en-ext" is formed from the two strings

```
loc-part a-mark aliph-stem
a-mark aliph-stem loc-part
```

Again there is a possibility that no locants need to be stated even when a multiplying term is present, for example, in propadiene, where the two double bonds can only logically be in one arrangement. So the production rule for "gm-en-ext" is

```
gm-en-ext = loc-part a-mark aliph-stem ,
            a-mark aliph-stem pos-loc-part ;
```

In a similar way the production rules for "yn-ext" can be formed, making reference where appropriate to the "en-ext" rules. The terminal "aliph-stem" can be replaced by the more general nonterminal "aliph-part", which has the terminal symbol "aliph-stem" as one of the rule alternatives but which also has appropriate production rules to handle alicyclic parents.

The second grammar will handle all the possible examples of unbranched, unsubstituted IUPAC organic nomenclature such as 2-decene, dec-2-ene, and 3-penten-1-yne but will still allow invalid names such as -3-pentene and 3-pentadiene.

The first grammar recognizes names such as pent-2,4-diene, which are not strictly correct according to IUPAC rules because such names lack an "a" after the parent ("pent") morpheme. In that grammar the "a" is optional at all times. The presence or lack of the "a" does not affect the meaning of the name, and it can be checked in the semantic phase, when a specific and meaningful warning message may be given. The second example hydrocarbon grammar will reject such a name as being syntactically incorrect and will not process it, even though the error is a minor one and does not affect the meaning of the name.

For the overall grammar design, a combination of both top-down and bottom-up approaches has been adopted, since it confers greater flexibility on the handling of elementary textual errors discussed above.

TRIVIAL NOMENCLATURE

As described in part 1,[1] the IUPAC nomenclature can be subdivided into systematic, semisystematic, and trivial forms, where a trivial term can be regarded merely as a label for a particular structure or substructure and the text of the term is without inherent meaning. The usage of such trivial terminology is variously restricted by the systematic rules in the extent to which trivial terms can be used in systematic constructions.

A number of fully trivial names are allowed by the IUPAC rules in deference to custom and practice, provided no attempt is made to build longer names by substitution. Somewhat more trivial terms are allowed within otherwise fully systematic names, thus creating the semisystematic form. In some cases such terms may be used entirely analogously to systematic terms in a given context, while in other cases restrictions are placed on the circumstances in which trivial terms may be so used.

```
TERMINALS

    special-case   =  ethylene,
                      allene,
                      ...,
                      isobutane,
                      neopentane,
                      ...,
                      chloroform,
                      phosgene;

    o-mark         =  o;
    o-sub-root     =  chlor,
                      brom,
                      ...;

    pref-sub-root  =  chlorosyl, ...,
                      iodyl, ...,
                      carboxy, ...,
                      formyl;

    phen-rad       =  phenyl;

    rad-mark       =  yl;

    triv-sub-A     =  isopropyl, ...,
                      tert-butyl;

    triv-sub-C     =  vin,
                      all;

    unsub-alcoxy   =  isopropoxy, ...,
                      tert-butoxy;

RULES

    org-name       =  aliph-name, ...,
                      special-case;

    substituent    =  prefix-sub,
                      subst-sub,
                      unsubst-sub, ...;

    prefix-sub     =  o-mark o-sub-root,
                      pref-sub-root;

    subst-sub      =  rad-mark  rad-sub,
                      phen-rad;

    rad-sub        =  sat-seq  aliph-stem, ...,
                      triv-sub-C;

    unsubst-sub    =  triv-sub-A, ...,
                      unsub-alcoxy;
```

Figure 6. Selected example rules for trivial terms.

It has proved possible to embrace many trivial and semisystematic constructions within the grammar rules, in the former case merely by adding specific rules to handle each instance. In the latter case, the additional rules need to be carefully considered, and some ingenuity is required to ensure that the necessary restrictions are incorporated for the trivial terms without disturbing the systematic constructions.

The two approaches are illustrated by the grammar extracts given in Figure 6, which shows terminal symbols, fragments, and rules for special-case complete trivial names and certain trivial substituent terms. Some terms such as isopropyl and *tert*-butyl (Rule A-2.25) must not be further substituted, while others such as vinyl and allyl (Rule A-3.5) may be. Hence these terms are grouped separately under the rules for "unsubst-sub" and "subst-sub", respectively, and the "subst-sub" class only may be permitted also in rules governing complex, bracketed substituents (not illustrated). The use of systematic branched-chain substituents is simply allowed by the rule for "rad-sub", which links back to the "aliph-stem" rules illustrated in the earlier figures.

SCOPE OF THE GRAMMAR

The grammar developed from this work comprises 114 terminal symbols representing about 350 morphemes. These are manipulated by 233 production rule alternatives. This grammar covers much of hydrocarbon nomenclature and has been extended to recognize the names of many acids, alcohols, aldehydes, ketones, and ethers. The basic nomenclature system recognized is substitutive nomenclature, although conjunctive nomenclature used to name acids, aldehydes, and alcohols is recognized as is radicofunctional nomenclature to name ketones, ethers, and alcohols. The full grammar is given as an appendix in the microfilm version of the journal.

Substitutive Nomenclature. This is the predominant system used to form IUPAC systematic names. The substituents on a parent structure are assumed to replace a hydrogen atom. If one of the substituents is a principal group, it is cited as a suffix; otherwise, the substituents are cited as prefixes to the name of the parent structure.

(a) Parent Structures. The parent structures recognized are aliphatic chains, aromatic rings, or alicyclic rings.

(i) Aliphatic Chains. Chains of up to 99 carbon atoms in length are recognized (Rule A-1.1 in the Blue Book[2]). These may be saturated or unsaturated. When unspecified, the locant of a single unsaturation is assumed to be 1 for convenience. If more than one unsaturation is present, then the correct number of locants is given directly before the multiplying term. The position of the unsaturations in names such as butadiene cannot be inferred. The name needs to be input in full, that is, buta-1,3-diene.

Certain trivial names are also recognized by the grammar (Rules A-2 and A-3), some of which cannot be substituted.

(ii) Alicyclic Rings. Monocyclic rings from cyclopropane to cyclononanonacontane are recognized and may be saturated or unsaturated. Bicyclic rings with a bridge containing no atoms are recognized, for example, bicyclo[4.4.0]decane.

(iii) Aromatic Rings. Benzene and its six common trivial derivatives are recognized (Rule A-12.1). The locants of disubstituted benzene derivatives may be given either by numerals or by the ortho, meta, para notation.

The majority of the trivial and semisystematic names given in Rules A-21.1 and A-21.2 for fused ring systems are also recognized.

(b) Principal Groups. The following principal groups are recognized by the grammar.

(i) Carboxylic Acids. All the systematic aliphatic chain names may be used as the basis for the acid names. Chains that terminate with a single carboxylic acid group (COOH) are recognized as being the aliphatic chain name with "oic acid" replacing the terminal "e" (Rule C-401.1). Likewise, a chain that has an acid group at either end is formed by adding "dioic acid" following the terminal "e". Aliphatic chains with more than two carboxylic acid groups are named by adding "carboxylic acid" to the end of the name prefixed by the multiplying term to describe the number of carboxylic groups present and the locants (Rule C-402.1), for example, octane-1,3,5-tricarboxylic acid.

Alicyclic acids and aromatic acids are named by adding "carboxylic acid" prefixed by the multiplying term and locants (if any) to the end of the name. Thirty-eight trivial aliphatic and aromatic acid names are also recognized, including formic acid and benzoic acid.

(ii) Aldehydes. Aldehydes are handled by production rules very similar to those that handle the carboxylic acid nomenclature. Hence, a similar range of aldehydes is handled as acids. A monoaldehyde name is formed from aliphatic chains by the replacement of the terminal "e" with "al" and a dialdehyde by the addition of "dial" to the end of the name (Rule C-302). If more than two aldehyde groups are present, then the suffix "carbaldehyde" is used prefixed by the multiplying term and locants (Rule C-303). This is the form of nomenclature used to name aromatic ring aldehydes. If a functional group that has a higher priority than the aldehyde group is present, for example, a carboxylic acid group, then the aldehyde is placed as the prefix "formyl" with multiplying terms and locants as necessary. A number of trivially named aldehydes that are derived from the trivially named carboxylic acids by using Rule C-305 are also recognized, for example, acetaldehyde.

(iii) Ketones. Currently only aliphatic chain ketones are recognized in accordance with Rule C-312. A ketone name is formed by replacing the "e" at the end of the aliphatic name by "one" with an optional locant or by adding "one" prefixed by a multiplying term and locants. No trivial ketones are recognized.

(iv) Alcohols. Alcohols are named in accordance with Rule C-201 by replacing the terminal "e" with "ol" for monoalcohols and by adding "ol" prefixed by multiplying term and locants for polyalcohols. Alcohol groups are also recognized by the prefix "hydroxy", with a multiplying term and locants if required, when they are not the principal group on the parent or if they occur on a side chain (Rule C-201.2). Four trivially named aromatic alcohols are recognized, including phenol.

(v) Ethers. Some ethers may be named in accordance with Rule C-211 by adding the alkyloxy or aryloxy radical name as prefix to the parent name. Such radical names recognized include chain and ring systems amended by -yloxy and nine contracted forms such as methoxy.

(c) Substituents. The following substituents are recognized as prefixes to the parent structures given above.

(i) Alkyl Chains. Any of the systematic parent alkanes described above may be used as a substituent by replacing the terminal "e" by "yl". These substituents may then be further substituted, in which case the complex substituent formed is given in parentheses. Hence, *o*-chloropropylbenzene is recognized as a valid name and is interpreted as a benzene ring with a chlorine atom and a chain of three carbons on adjacent atoms of the ring. In contrast, (2-chloropropyl)benzene is also recognized and is interpreted as a benzene ring with one substituent, a chain of three carbons having a chlorine atom attached to the second. Any level of complexity is recognized, provided parentheses are used throughout.

(ii) Aromatic Rings. All of the trivial aromatic ring systems recognized may be used as radicals to a parent structure by replacing the terminal "e" with "yl". The exceptions to this rule are benzene, naphthalene, and anthracene, which when cited as prefixes are given as "phenyl", "naphthyl", and "anthryl". In most cases the radical needs to be prefixed by a locant to indicate the position of attachment of the parent structure to the ring system.

(iii) Cycloalkyls. Only singly ringed alicyclic compounds are recognized as prefixed substituents. These may be saturated or unsaturated and may be further substituted.

(iv) Radicals That Can Only Occur as Prefixes. A number of radicals can only appear as prefixes in substitutive nomenclature, as given in Table 1 of Rule C-10.1. The grammar recognizes the majority of these radicals.

Conjunctive Nomenclature. Conjunctive nomenclature has been used to name some carboxylic acids and alcohols in accordance with Rule C-51. It is a simple task to extend the production rules to cover aldehydes, but this has not yet been done. Conjunctive nomenclature is used in naming chains containing a functional group where the chain is directly attached to a ring system. The attached ring system may be any of the aromatic ring systems recognized above. Substituents attached to the chain are named by using Greek characters as locants. These need to be written out in full because of the lack of Greek characters in the ASCII character set and on terminal keyboards. An example of a conjunctive name recognized by the software is beta,delta-dimethyl-1-naphthalenevaleric acid.

Radicofunctional Nomenclature. In general, substitutive nomenclature is preferred to radicofunctional nomenclature, but the latter is included to demonstrate the versatility of the grammar-based technique in recognizing different forms of nomenclature. The Blue Book describes the use of radicofunctional nomenclature in Rules C-21–24. Both monovalent and divalent functional groups are recognized, with the radical part being an aliphatic chain radical or a phenyl group. The monovalent functional groups recognized include halides, alcohol, and thiol; the divalent functional groups include ketone, sulfone, and ether.

CONCLUSION

A context-free, phase structure grammar has been developed for the IUPAC systematic organic chemical nomenclature. While the grammar is incomplete, a substantial part of organic nomenclature is covered, and the description of how the grammar was developed should enable others to devise rules to cover more parent structures, principal groups, and substituents.

The capability of the grammar-based approach to cover a variety of nomenclatural forms is demonstrated by the inclusion of not only substitutive nomenclature but also examples of conjunctive and radicofunctional nomenclature and some semisystematic and trivial nomenclature.

The benefits of a formal grammar for any language are unambiguity and facilitation of machine processing. It is hoped that the presentation of this grammar for IUPAC systematic nomenclature and the description of the substantial quantity of work that this has entailed may lead to the publication in the future of nomenclature rules in a more amenable form for computer processing.

Further papers in this series develop the use of the grammar described here for the automatic recognition and translation of chemical nomenclature.

ACKNOWLEDGMENT

We gratefully acknowledge funding of this research by the Laboratory of the Government Chemist and advice on the use of IUPAC systematic nomenclature received from E. W. Godly and I. Cohen.

Supplementary Material Available: Current grammar for the IUPAC systematic organic chemical nomenclature (14 pages). Ordering information is given on any current masthead page.

REFERENCES AND NOTES

(1) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 1. Introduction and Background to a Grammar-Based Approach. *J. Chem. Inf. Comput. Sci.* (first of three papers in this issue).

(2) International Union of Pure and Applied Chemistry. *Nomenclature of Organic Chemistry, Sections A-F and H*; Pergamon: Oxford, U.K., 1979.

(3) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 3. Syntax Analysis and Semantic Processing. *J. Chem. Inf. Comput. Sci.* (third of three papers in this issue).

(4) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. In *The Awards of Science and Other Essays*; ISI Press: Philadelphia, 1985.

(5) Thomas, M. I. *A Basic SLR Parser-Generator*; Report No. 80/3; Department of Computer Science, University of Hull: Hull, England, 1980.

(6) Aho, A. V.; Ullman, J. D. *Principles of Compiler Design*; Addison-Wesley: Reading, MA, 1978; pp 204-214.

(7) Rayner, J. D. Grammar Based Analysis by Computer of the IUPAC Systematic Chemical Nomenclature. Ph.D. Thesis, University of Hull, Hull, England, 1983.

(8) Cooke-Fox, D. I. Computer Translation of IUPAC Organic Chemical Nomenclature to Structure Diagrams. Ph.D. Thesis, University of Hull, Hull, England, 1987.

(9) The full grammar is given as an Appendix to this paper in the microfilm edition of the Journal.

Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 3. Syntax Analysis and Semantic Processing

D. I. COOKE-FOX, G. H. KIRBY,* and J. D. RAYNER

Department of Computer Science, University of Hull, Hull HU6 7RX, England

Received May 6, 1988

The construction of computer software to translate from IUPAC systematic organic chemical nomenclature into concise connection tables (CCTs) is described. A parser to perform the syntax analysis phase was produced by application of a modified SLR parser generator to the context-free grammar developed in the second paper of this series. Semantic processing of names accepted by the parser involves a second pass through the name by the established path. Semantic information associated with the morphemes, in the form of CCT fragments, is extracted from the dictionary and used to build the CCT. Data structures are described that are used to check the chemical validity of the CCT, and hence the name, and to ensure alphabetic ordering of substituent prefixes.

INTRODUCTION

Part 1 in this series[1] described the background to this project in which computer techniques and software have been applied and developed to translate IUPAC systematic organic chemical nomenclature into concise connection tables and hence to displays of the corresponding structure diagrams. The grammar-based approach that was adopted required first the development of a formal grammar for IUPAC systematic organic nomenclature, and such a context-free phrase structure grammar for a number of classes of compounds was reported in part 2.[2]

Having a context-free grammar allows application of the process commonly used for the translation, or compilation, of computer programs into machine or similar instructions. The process of compilation is complex but can be considered a series of phases, namely, lexical analysis, syntax analysis or parsing,