and J. Note that the symmetry is now correctly perceived due to the 1547 ≠ 2057 tiebreak in row F.

The symmetry classification in row K is stable (recognized by being identical with previous classification in row I).

As described in the text, CANON continues by breaking the lowest tie (symmetry class 2, nitrogens) to produce 12 distinct labelings. Starting with the lowest labeled atom and branching to lower labeled atoms at forks in the structure, the unique SMILES, CCC(CO)CCC(CN)CN, is established by GENES.

REFERENCES AND NOTES

(1) Weininger, D. *J. Chem. Inf. Comput. Sci.* **1988**, *28*, 31.
(2) Balaban, A. T. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 334.
(3) Joachim, C.; Gasteiger, J. *Top. Curr. Chem.* **1987**, *74*, 93.
(4) Wipke, W. T.; Dyott, T. M. *J. Am. Chem. Soc.* **1974**, *96*, 4834.
(5) Morgan, H. L. *J. Chem. Doc.* **1965**, *5*, 107.
(6) Bersohn, M. *Comput. Chem.* **1987**, *2*, 113.
(7) Hagadone, T. R.; Howe, W. J. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 182.
(8) Uchino, M. *J. Chem. Inf. Comput. Sci.* **1980**, *20*, 116.
(9) Freed, E. E. Harvey Mudd College, personal communication.
(10) Wenger, J. C.; Smith, D. H. *J. Chem. Inf. Comput. Sci.* **1982**, *22*, 29.

Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 1. Introduction and Background to a Grammar-Based Approach

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

Since Garfield's pioneering work over 25 years ago in the linguistic aspects of systematic chemical nomenclature, leading to an algorithm for translating chemical names to formulas, very few reports of grammar-based analysis of systematic chemical nomenclatures have appeared in the literature. These have applied only to a few specific classes of names. While the major abstracting services use automated methods to process chemical nomenclatures, the limited details that have been published point to ad hoc approaches based on dictionaries of morphemes. This paper introduces a series that covers in detail the various aspects of the application of grammar-based techniques to the recognition of IUPAC systematic chemical nomenclature and hence the translation of chemical names to structure diagrams. Some necessary elements of language and grammar are discussed here in the context of the automatic recognition of chemical nomenclature.

INTRODUCTION

There are three broad categories of chemical language by which structural information is represented and communicated. These are the nomenclatures used to name compounds, formulas and line notations used as shorthand representations of compounds, and structure diagrams used as the primary means of communication of structural information and compounds. Chemical structures are also represented by connection tables, which are used internally by most computer-based transformation techniques as a topological description of molecular structure. However, connection tables are rarely used for communication between people and are not regarded as languages. The translation or interconversion of these languages by automatic means is an important application of computer science to chemical structure representation and processing. A review with references to those interconversions that have been reported is given by Rush.[1]

Computer translation from and to a systematic nomenclature has received little attention, and a recent book[2] has said that existing programs are very large and complicated and will be successful in this translation in considerably less than 100% of cases. This situation is associated with the slowness with which systematic nomenclature, as typified by the schemes devised by the International Union of Pure and Applied Chemistry (IUPAC), is accepted and used, with the continuing use of much semisystematic and trivial nomenclature, and with the questionable need for fully systematic nomenclature as perceived by the chemical industry.[3] In the U.K., the Chemical Nomenclature Advisory Service of the Laboratory of the Government Chemist encourages the use of systematic nomenclature following the principles set by IUPAC and is prominent in advising European Commision Services on these matters. Egan[4] and Egan and Godly[5] have discussed some of the benefits of using IUPAC systematic nomenclature, while the issues and problems associated with the use of chemical nomenclature are covered in the book edited by Lees and Smith.[6]

Work supported by the Laboratory of the Government Chemist has been in progress in this department for some years to investigate the application of grammar-based techniques, as developed for compiling computer programming languages, to automatic name recognition and translation into structure diagrams. In this project attention has been paid to certain classes of compounds of industrial importance, including some cases of semisystematic and trivial nomenclature. A particular feature of the project has been the use of inexpensive and readily available computing facilities as exemplified by the IBM PC and compatible microcomputers. An outline of the project in its early stages has been published.[7]

The first step in the translation of chemical nomenclature by grammar-based techniques is to develop a grammar that formally describes the syntax of the nomenclature. From the grammar, a parser can be produced to recognize names that satisfy the grammar and to check the semantics, or meaning, of the names. Names that are syntactically correct may nevertheless be chemical nonsense. Only after satisfying semantic checking is an intermediate form of a name constructed, the concise connection table.[8] Further processing leads to representations suited to communication to other computer software or to the display of a structure diagram.

CHEMICAL NOMENCLATURES

Overview of Nomenclature Styles. An excellent review of the development of chemical nomenclature is given by Cahn and Dermer.[9] Following the Geneva Congress of 1892, the maintenance of the rules of chemical nomenclature was taken on by the International Union of Pure and Applied Chemistry (IUPAC), who published revisions to the rules of organic

0095-2338/89/1629-0101$01.50/0

chemical nomenclature in 1930, 1957, and again in 1979.[10]

The other internationally recognized scheme of nomenclature is that of Chemical Abstracts Service (CAS). The CAS development of a systematic method of naming compounds initially arose from the preparation of the first 10-year cumulative subject index. The CAS nomenclature is used to index *Chemical Abstracts*, and to that end CAS has dispensed with many trivial names.

Chemical nomenclatures can be divided into three parts: systematic, trivial, and semisystematic or semitrivial.

(1) *A systematic name* is composed wholly of syllables specially coined or selected to describe the structural features of the name, for example, pentane, where "pent" implies a structure of five atoms and "ane" implies a saturated carbon chain. By the use of systematic chemical nomenclature, it is possible to give a complete, unique, unambiguous representation of the structure of an organic chemical compound.

(2) *A trivial name* is a label for a compound and gives no information about the structure of the compound. Acetic acid is an example of a trivial name, corresponding to the systematic name ethanoic acid.

(3) *A semisystematic or semitrivial name* is part systematic and part trivial. Usually the parent structure is a trivial name that has been expanded by using systematic nomenclature, for example, 3-chlorophenol.

IUPAC has retained those trivial names that still have a significant use in industry and commerce and uses these trivial naming roots as the basis of semisystematic names. For example, the widespread use of the name acetic acid ensures that in the near future it is unlikely to be replaced in common usage by the systematic name ethanoic acid. Because in individual circumstances it is not always possible to agree on the most desirable type of name, there are cases where alternative names are equally correct according to IUPAC rules.

Chemical nomenclature has developed as a written, rather than a spoken language. Problems occur with oral communication, not only through the difficulty of pronouncing the long multisyllable systematic names but also through different names having the same sound, for example, fluorine, an inorganic gas, and fluorene, an organic compound.[11] However, there are few examples of two substances having the same written name.

It is interesting to note that, in the development of the rules from the Geneva Congress to the IUPAC rules of 1979, it is the syntax of the rules that has changed, rather than the morphology. This is not the way natural languages develop, where it is the morphology or word construction that changes most rapidly.[12] Even where radically new, more systematic, nomenclature systems have been proposed,[13,14] the familiar vocabulary and punctuation forms are largely retained.

Computer Processing of Chemical Nomenclatures. Around 1960, Garfield investigated general linguistic aspects of systematic nomenclature and developed a method for the calculation of molecular formulas based on the determination of significant parts of each chemical name. This was the first work on the direct conversion of names to formulas, and Garfield described a manual algorithm by which formulas could be formed from atom and bond information, derived in turn from a dictionary of name parts, or morphemes.[12,15] Garfield identified the commonly occurring groups of letters, termed "morphs", and proceeded to investigate which of these were significant and which were not. Those of significance, called "morphemes", were admitted to the vocabulary and included in the dictionary, while the others were discarded.

At this point, the morphemes were classified as denoting structure formation or structure modification. The recognition of structure-forming morphemes within a supplied name caused the addition of specific elements to the accumulating formula, while modifying morphemes caused adjustment up or down of particular element counts, typically of hydrogen. Hence, the molecular formula of the name could be computed. The only classification of morphemes in this work was as "former" or "modifier", and no formal syntax was developed to control the combinations of morphemes accepted. Names were analyzed from left to right, with the longest possible morpheme being matched at each stage, and ad hoc methods were used to handle the bracketing of substituents.

Chemical Abstracts Service is extensively involved with the automated computer handling of its own chemical nomenclature, designed to facilitate the indexing and retrieval of journal abstracts concerning chemical compounds. The introduction of computers to the CAS system in the early 1960s led to the development of well-defined translation procedures for converting CAS systematic names to atom–bond connection tables.[16] Computer programs based on these procedures were subsequently reported,[17] and these were used to validate the CAS index names contained in early records and to convert them to a common internal form for storage. These routines have since been extended to provide editing and verification facilities for newly appearing index names.[18]

The method used is essentially ad hoc in nature, but it can cope with a wide selection of CAS names due to the very tight rules of this nomenclature. It would be fair to say that some of the gradual adjustments to the CAS nomenclature, over successive indexing periods, have been due to the requirements of the developing nomenclature translation programs.

As with Garfield's work, the CAS procedures are based on dictionaries of basic word roots, and, once these are identified, their semantic data are again subject to essentially ad hoc treatment. The drawback to this approach is that special routines have to be invoked frequently, for instance, when bracketed, highly branched substituents are dealt with.[17]

The CAS nomenclature is fully systematic and related to that of IUPAC insofar as there are many terms in common between the two systems. However, the CAS nomenclature has very different rules for the relative ordering of terms within names, designed originally to aid the indexing and retrieval operations that are the main occupation of CAS. The appearance of the "heading parent"—the main indexing term—as the first part of the name when read from left to right, followed by a "substituent part" and finally an optional "name modification" prompts an apparently simple sequential algorithm for name processing.

In the report of the CAS conversion procedures,[17] a simple sequential example of processing is given, after which a number of "special characteristics" are introduced as needing extra processing. Of these, the occurrence of "complex radicals" within the substituent part caused problems through the multiple, bracketed nesting of terms. Such constructions are best defined and handled through recursive techniques that were not available at the time that the CAS procedures were first developed. In the IUPAC nomenclature of organic compounds, the parent clause appears in its "natural" place to the right of any substituents. Thus, if names are processed from right to left, as has been done in our work, the handling of substituents follows naturally, needing no special treatment.

The first use of a computerized grammar analysis process was by Elliott.[19] A set of programs was used that had been developed earlier to assist in the production of compilers for computer-programming languages. Elliott developed a context-free grammar of the type required by this system, which expressed the rules for constructing the names of simple hydrocarbons according to the IUPAC nomenclature. The grammar analysis produced a set of tables that, in conjunction with a dictionary of nomenclature terms, were used to drive a recognizer for names from this subset of the nomenclature.

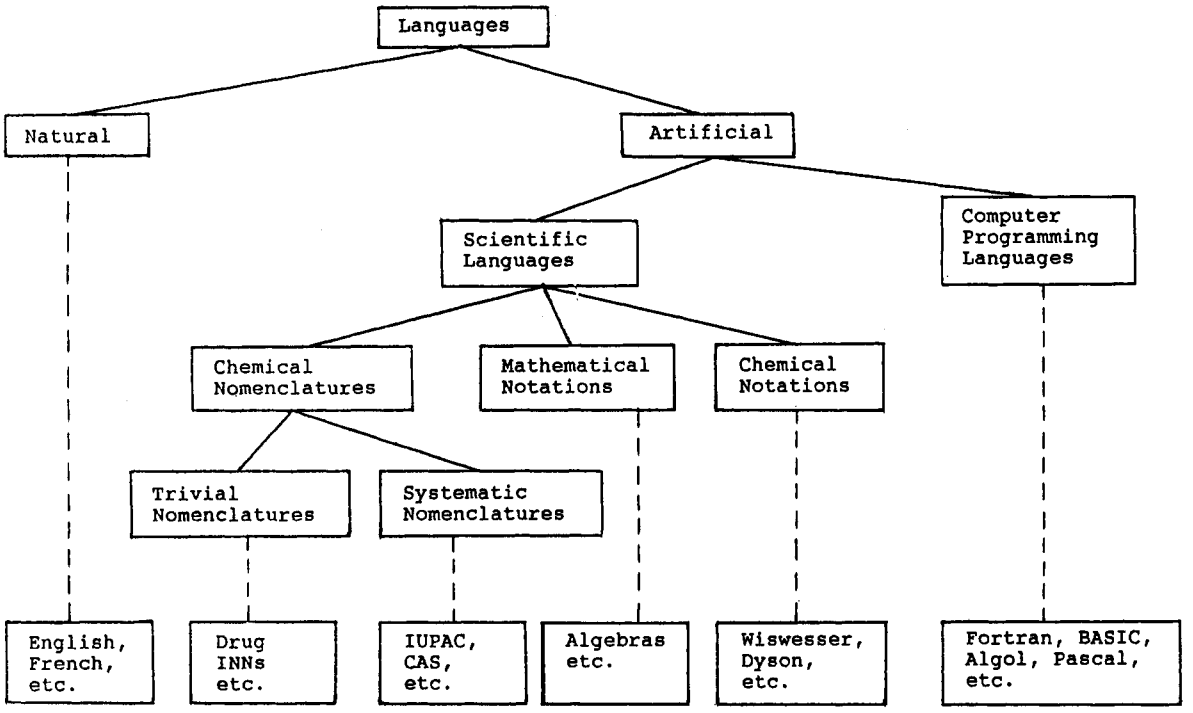

Figure 1. Place of chemical nomenclature within a classification of languages.

Not only was it possible with this scheme to extract structural information for the output of formulas, but due to the syntactic foundations of the work it was also possible to check the construction of input names and to handle bracketed and nested constructions within the same system. However, the scope of the program was limited in comparison with the broad range of constructions allowed in the IUPAC nomenclature.

Two other studies of nomenclature translation have been reported. Work by Stillwell[20] was limited to the area of steroid nomenclature, using a small dictionary of steroid nucleus names and common substituent terms. By means of ad hoc processing methods, a tabular representation of molecular geometry was formed for each input name, and diagrammatic output was then produced. Stillwell made no use of formal syntactic methods and concentrated on the stereochemical aspects of steroid structures in his output.

Carpenter[21] used syntactic methods to produce stick diagrams from nomenclature, via a connection table. As did Elliott earlier, Carpenter used a preexisting syntax-analysis package to process a grammar for simply branched hydrocarbons, and he also implemented the expansion of line formulas to names for the same grammar. This was possible since in this limited class of nomenclature it is not unreasonable to write bracketed formulas that contain the necessary structural information.

Carpenter considered that such translation schemes were of limited applicability owing to the use of trivial names and parents in IUPAC systematic nomenclature. The current project has demonstrated that the interpretation of semisystematic names is possible with grammar-based methods.

Grammars have been reported for other chemical languages. Chemical formulas can be recognized and parsed when presented in the single-line formats described by two published grammars.[22,23] GENSAL, developed for the Sheffield University Generic Chemical Structure Research Project,[24] is a specially devised formal language suited to use by chemists. It allows generic chemical structures to be described by the user and converted by computer to internal representations of structures that are used for searching databases. In the same project, a simple topological grammar has been devised with which chemical structure fragments, such as the alkyl, alkenyl, and alkynyl radicals, can be generated or recognized.[25] That paper reviews and demonstrates work toward grammars for the third category of chemical language, namely, structure diagrams.

LANGUAGES AND GRAMMARS

Natural and Artificial Languages. Languages can be broadly classified as natural or artificial in the manner of Figure 1. The natural languages are those developed by humanity over the millennia, which tend to be both ill-defined in any formal sense and subject to uncontrolled evolution in the introduction of fresh vocabulary, dialects, and so on. In contrast, artificial languages are those introduced or developed by interested parties to fulfil specific needs. Although in some cases they too may be not rigorously defined, and also subject to change, nevertheless, examples exist of artificial languages that are both well-defined and static. In Figure 1, the languages named are arranged in an approximate order of rigorous definition, with the fully natural languages on the left and the fully artificial on the right. Thus, we consider chemical notations such as Wiswesser line notation (WLN)[26] and chemical formulas to be more formally specified than mathematical notations whose evolution has been more "natural". The chemical nomenclatures are considered to be artificial languages, though at the natural extreme. They are by no means static and may be subject to changes of definition from time to time.

Phrase Structure Grammars. A language may be defined in terms of a grammar that is seen intuitively to have two components: a vocabulary of words from which sentences can be built and a set of rules that govern the juxtaposition of words in each sentence. The rules thus restrict syntactically valid sentences to a subset of all possible combinations of vocabulary words, but they have no control over the semantics of such sentences. A chemical name may be regarded as a sentence in one of the languages that are chemical nomenclatures.

It is a simple, though lengthy and tedious, task to draw up a vocabulary for any known language, but to express the rules of a grammar some additional material is necessary. Grammar rules are written not only in terms of the vocabulary words, or rather classifications of these called *terminal symbols*, but also in terms of the names given to particular phrases or

constructions in the language, *the nonterminal symbols*. One particular phrase name, *the distinguished symbol*, denotes the overall unit, or sentence, of the language.

A *production* or *grammar rule* consists of a left part and a right part separated by a production symbol, which is commonly written as a right arrow (→), an equal sign (=), or two colons followed by an equal sign (::=). In the most general case both the left and right parts may contain terminal and nonterminal symbols. For certain types of grammar, the left part consists of just one nonterminal, which names the phrase, with on the right a list of terminals and nonterminals that define the components of the phrase. In this case, a sentence of the language defined by such a grammar may be generated by application of a sequence of production rules starting from the distinguished nonterminal symbol.

Chomsky Classification. Chomsky[27] defined four types of phrase structure grammars that may be used in the analysis of natural and artificial languages. Each type of grammar described a class of languages that is a proper subset of the class of languages defined by a lower numbered grammar type. That is, type 0 grammars are the most general and can define all the languages of type 1, type 2, and type 3 grammars, the last being the most restrictive. The type of a Chomsky phrase structure grammar is defined by the restrictions placed on the format of the production rules, which in turn restrict the language that can be described.

Chomsky type 3 grammars generate languages called finite state or regular languages. All type 3 grammars have production rules of just two forms. Either a single nonterminal symbol is rewritten as a single terminal symbol, or a single nonterminal symbol is rewritten as a single nonterminal symbol followed by a single terminal symbol, or it is rewritten as a single terminal symbol followed by a single nonterminal symbol. An example of a language described by a type 3 grammar would be where one or more a's can be followed by one or more b's, and the numbers of each may be different, for example

aabbb, aaaab, ...

Chomsky type 2 grammars are called context-free grammars, and they have production rules where a single nonterminal symbol is rewritten by a string of terminal and nonterminal symbols. There may be any number of symbols in the right part, with no restriction on the arrangement of the terminal and nonterminal symbols. An example language that may be described by a context-free grammar but not by a regular grammar is

ab, aabb, aaabbb, aaaabbbb, ...

It is not possible to describe this language with a regular grammar because of the symmetry around the junction point between the string of a's and b's. As indicated above, a regular grammar would only be able to describe a language where any number of a's are followed by an equal or different number of b's. The context-free grammar has a production rule whose right part has a nonterminal symbol sandwiched between two terminal symbols, which allows the grammar to define languages of this symmetric type.

Chomsky type 1 grammars are called context-sensitive grammars and have still weaker restrictions on the form the production rules may take. Both the left and right part of the production rules may have one or more symbols, but the left part must have at least one nonterminal symbol. An example of a context-sensitive language is

abc, aabbcc, aaabbbccc, ...

whose grammar is given by Cleaveland and Uzgalis.[28] This is not a context-free grammar, since we can describe a language with an equal number of a's and b's, but the additional restriction that there be the same number of c's cannot be described with a context-free grammar.

Chomsky type 0 grammars are called recursively enumerable and provide the most general description.

The relationship between the Chomsky classification of grammars and programming language definitions is well documented.[28,29] The context-free grammars are the most common grammars used in this connection. There is a type of recognizer algorithm for each type of phrase structure grammar, and recognizers for types 2 and 3 have been widely developed to compile computer programs.

Phrase Structure Grammars and Chemical Nomenclature. Chemical nomenclature is a language whose syntax has been described by context-free phrase structure grammars.[19,30,31] A recognizer may thus be written to recognize chemical names in the same way that a compiler recognizes valid computer programs. However, a pure recognizer that is able to show the conformity of a given input string to a given grammar is not sufficient, since it is also necessary to identify the precise detailed meaning of the name to produce an alternative, translated representation. Thus, the development of the grammar must also take into account not only the nature of the source language itself but the manner in which practical translation will occur during later stages of parsing and semantic processing. Assumptions and apparent restrictions introduced during grammar construction can often be checked and relaxed during these later stages of the overall translation process.

The vocabulary of chemical nomenclature consists of a number of punctuation symbols (commas, hyphens, and enclosing marks such as parentheses), integers, and strings of alphabetic characters. Much of the language of organic chemical nomenclature can be described by a regular grammar, but there are constructs that are not regular. For example, the normal nesting order of enclosing marks is {[()]}, where parentheses are used first as the innermost marks and then brackets and braces, repeating the sequence if further nesting is needed.

The choice of enclosing mark is thus dependent on those used before, so the language is not even context free. However, if enclosing marks are restricted to just parentheses, their use is similar to the context-free language {ab, aabb, aaabbb, ...} described earlier, with the a's representing opening parentheses and b's closing.

Given this restriction, the language of organic chemical nomenclature can be represented by a context-free grammar. The grammar does not describe a fully artificial language, but rather a quasi-natural one that has developed by custom and practice over a period of time and is still developing. Thus, there is a need for the grammar to be easily modified to cope with future changes in nomenclature definitions. Significantly, chemical nomenclature differs in two ways from high-level programming languages, whose grammars generally are also context free.

First, it is readily apparent that chemical names (e.g., iododecane) have few if any clearly defined delimiting characters between the significant parts of the name (the "words" of the sentence), whereas in most programming languages the space is frequently used as a separator. This lack of delimiters, other than self-delimiting symbols such as hyphens, commas, and parentheses—as in 5,6-bis(2-iodopropyl)decane—requires a more complex method for recognizing name fragments and finding their terminal symbols, since it is generally necessary to establish the extent of the next fragment to be processed. For example, iododecane must be split into iodo-decane, not into io-dodecane.

Second, the semantically most significant and style-determining part of a chemical name is typically toward the right-hand end, whereas in most programming languages

significant terms occur to the left of subordinates. For example, compare the substituted decane name given above with the Pascal statement below, where the **while** on the left determines the interpretation of what follows:

while ch <> '.' **do** read(ch);

The mapping of an essentially right-rooted nomenclature onto techniques devised for left-rooted programming languages is eased by the shortness of chemical names relative to the typical program. This allows the entire text of a name to be absorbed and processed from right to left, presenting an effectively left-rooted language to the translator.

DISCUSSION

IUPAC systematic chemical nomenclature is a language to which a grammar-based approach may be applied for the automatic recognition and translation of names. Structure diagrams are the primary means of communication of chemical structure information between chemists and a target into which other representations can usefully be translated. Since the structural meaning of the syllables that are used to form a systematic name and the rules that govern the assembly of the structural information into a complete structural representation are both known, then a structure diagram can be produced automatically from a systematic name. The structure diagram for a trivial name cannot be derived in this way, but can only be obtained by knowing the structure corresponding to each trivial name. The structure diagram of a semisystematic name can only be derived if the structure of the trivial part is known.

Subsequent papers in this series include the discussion of the development of a formal grammar for the IUPAC nomenclature;[32] the parsing, syntactic, and semantic analysis of systematic chemical names;[33] the generation of connection tables and display of structure diagrams; techniques for handling semisystematic and specialist nomenclature; and the correction of errors in systematic nomenclature.

ACKNOWLEDGMENT

We acknowledge the support provided for this project by the Laboratory of the Government Chemist and the particular help and advice provided by R. Lees, E. W. Godly, and other staff of its Chemical Nomenclature Advisory Service.

REFERENCES AND NOTES

(1) Rush, J. E. Computer Hardware and Software in Chemical Information Processing. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 140–149.
(2) Gray, N. A. B. *Computer Assisted Structure Elucidation*; Wiley: New York, 1986; p 214.
(3) Silk, J. A. Realistic vs. Systematic Nomenclature. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 146–148.
(4) Egan, H. What's in a Name? *Chem. Br.* **1984**, *20*, 126–129.
(5) Egan, H.; Godly, E. W. Organisation and Chaos in the World of Nomenclature. *Chem. Br.* **1980**, *16*, 16–25.
(6) Lees, R, Smith, A. F., Eds. *Chemical Nomenclature Usage*; Ellis Horwood: Chichester, England, 1983.
(7) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D From Names to Diagrams—by Computer. *Chem. Br.* **1985**, *21*, 467–471.
(8) Rayner, J. D. A Concise Connection Table Based on Systematic Nomenclatural Terms. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 108–111.
(9) Cahn, R. S.; Dermer, O. C. *Introduction to Chemical Nomenclature*, 5th Ed.; Butterworths: London, 1979.
(10) International Union of Pure and Applied Chemistry. *Nomenclature of Organic Chemistry, Sections A–F and H*; Pergamon: Oxford, U.K., 1979.
(11) Coyle, J. D.; Godly, E. W. *Chemical Nomenclature*; Open University: Milton Keynes, England, 1984; Chapter 7.
(12) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. In *The Awards of Science and Other Essays*; ISI Press: Philadelphia, 1985; p 453.
(13) Lozac'h, N.; Goodson, A. L.; Powell, W. H. Nodel Nomenclature—General Principles. *Angew. Chem., Int. Ed. Engl.* **1979**, *18*, 887–889.
(14) Hirayama, K. *The HIRN System. Nomenclature of Organic Chemistry, Principles*; Maruzen: Tokyo, 1984.
(15) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. *J. Chem. Doc.* **1962**, *2*, 177–179.
(16) Vander Stouw, G. G.; Naznitsky, I.; Rush, J. E. Procedures for Converting Systematic Chemical Names of Organic Compounds to Atom Bond Connection Tables. *J. Chem. Doc.* **1967**, *7*, 165–169.
(17) Vander Stouw, G. G.; Elliott, P. M.; Isenberg, A. C. Automatic Conversion of Chemical Substance Names to Atom Bond Connection Tables. *J. Chem. Doc.* **1974**, *14*, 185–193.
(18) Vander Stouw, G. G. Computer Programs for Editing and Validation of Chemical Names. *J. Chem. Inf. Comput. Sci.* **1975**, *15*, 232–236.
(19) Elliott, P. M. Translation of Chemical Nomenclature by Syntax Controlled Techniques. M.Sc. Thesis, The Ohio State University, Columbus, OH, 1969.
(20) Stillwell, R. W. Computer Translation of Systematic Chemical Names to Structural Formulas—Steroids. *J. Chem. Doc.* **1973**, *13*, 107–109.
(21) Carpenter, N. Syntax Directed Translation of Organic Chemical Formulae into their 2-D Representation. *Comput. Chem.* **1975**, *1*, 25–28.
(22) Barker, P. G. Syntactic Definition and Parsing of Molecular Formulae: Part 1. Initial Syntax Definition and Parser Implementation. *Comput. J.* **1975**, *18*, 355–359.
(23) Kirby, G. H.; Milward, S. Syntax to Facilitate the Word Processing of Chemical Formulas. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 57–60.
(24) Barnard, J. M.; Lynch, M. F.; Welford, S. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 2. GENSAL, a Formal Language for the Description of Generic Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 151–161.
(25) Welford, S. M.; Lynch, M. F.; Barnard, J. M. Computer Storage and Retrieval of Generic Chemical Structures in Patents. 3. Chemical Grammars and Their Role in the Manipulation of Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 161–168.
(26) Smith, E. J. *Wiswesser Line Formula Chemical Notation*; McGraw-Hill: New York, 1968.
(27) Chomsky, N. Three Models for the Description of Language. *IRE Trans. Inf. Theory.* **1956**, *2*, 113–124.
(28) Cleaveland, J. C.; Uzgalis, R. C. *Grammars for Programming Languages*; Elsevier: New York, 1977; p 18.
(29) Hopcroft, J. E.; Ullman, J. D. *Introduction to Automata Theory, Languages and Computation*; Addison-Wesley: Reading, MA, 1979.
(30) Rayner, J. D. Grammar Based Analysis by Computer of the IUPAC Systematic Chemical Nomenclature. Ph.D. Thesis, University of Hull, Hull, England, 1983.
(31) Cooke-Fox, D. I. Computer Translation of IUPAC Organic Chemical Nomenclature to Structure Diagrams. Ph.D. Thesis, University of Hull, Hull, England, 1987.
(32) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 2. Development of a Formal Grammar. *J. Chem. Inf. Comput. Sci.* (second of three papers in this issue).
(33) 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).

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

0095-2338/89/1629-0106$01.50/0

```
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.

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):

```
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.

```
loc-part
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 at-

tached 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,

0095-2338/89/1629-0112$01.50/0

semantics and code generation.

In lexical analysis, the input string (a program in the case of compilation, but here a chemical name) is split into valid fragments and corresponding syntactic tokens are fed to the syntax analysis phase. The possible tokens that appear in a chemical name are the fragments defined in the nomenclature rules, and they can be associated with the syntactic tokens by dictionary look-up.

In syntax analysis not only are valid sentences of a language recognized but also the underlying structure of the sentence can be determined. An efficient parser can be automatically constructed from a context-free grammar by a software tool called a parser generator.[3] A number of such tools are in existence for the various parsing techniques that have been used to write compilers, including operator precedence, recursive descent, and LR, LL, and LALR parsing. We use a simple LR parser generator, SLR, based on an algorithm by Aho and Ullman[3] that we have extended from its initial implementation[4] to support various requirements of the present work.

A sentence accepted by a parser is one that belongs to the language described by its underlying grammar. That is, in the case of nomenclature, acceptance depends at this stage solely on syntactic form, and no meaning can as yet be ascribed to the name. Thus, no statement can be made by the parser as to the chemical validity of a name it accepts. This is the concern of the semantic phase where the validity of a name is determined by the structural information implied by the name and its correctness according to the laws of chemistry. A systematic name is composed of fragments specially selected to describe the structural features of the name. So, if the rules that govern the assembly of this structural information are known, then a structure diagram can be produced from any systematic name.

In any given language, fragments of a sentence that may be ascribed some individual meaning are termed "morphemes". However, the structural information associated with each systematic name morpheme must be represented within the computer system and assembled into a complete representation before the structure diagram can be drawn from it. This corresponds to the code generation phase of a compiler. While fragment codes[5] are able to describe the structural meaning of morphemes, they do not give the required information on the association of these structural fragments in the molecule. We preferred to use the concise connection table (CCT) that was originated by Rayner[6] specifically as a solution to this problem. Each entry or group of entries in the CCT depicts the structure for a morpheme and also describes the way in which this structural fragment is connected to others to form the whole structure.

IUPAC systematic organic nomenclature includes some trivial names that are still in common usage. However, the structure diagrams for these names cannot be derived in the manner outlined above since a trivial name is merely a label and contains no inherent structural information. Nevertheless, the structure diagram of a semisystematic name can be derived if the structure of the entire trivial part is represented within the grammar as for a systematic fragment.

The details in the next section explain the capabilities that we have incorporated in the parser to cope with the variety of nomenclatural constructions. An appreciation of these is necessary for the discussion, in the following section, of structure elucidation by semantic processing and, in a future paper, of error detection and correction.

SYNTAX ANALYSIS

SLR Parsing Process. LR parsers, of which SLR is a derivative, are so named because they scan the input from left to right and form a rightmost derivation in reverse. In contrast to typical computer-programming languages, for which the LR method was originally devised, the IUPAC nomenclature is a right-rooted language—that is, the main structure-determining parts of a name occur toward the end of the name, whereas in a typical computer program each construct is introduced by a unique keyword. Since chemical names are relatively short compared to programs (a few hundred characters at most, as against many thousands), an input name may be absorbed entirely before analysis and processed from right to left. Thus, the grammars developed and illustrated in this paper appear to represent names written "backward". However, actual input is presented in the normal way.

```
TERMINALS

     A-MARK,  EN-MARK,  ANE-MARK,  ENE-MARK,  YNE-MARK,
     ROOT  ;  LOCANT ;  HYPHEN  ;  COMMA   ;  MULT  ;


RULES

     NAME = ANE-MARK  ROOT  ,
            SAT-SEQ  ROOT   LOC-PART  ;

     SAT-SEQ = ENE-MARK  ENE-EXT  ,
               YNE-MARK  YNE-EXT  ;

     ENE-EXT = MULT  A-MARK , $ ;

     YNE-EXT = MULT  YNE-EN ,
               LOC-PART  HYPHEN  EN-MARK  ENE-EXT ,
               $ ;

     YNE-EN = A-MARK ,
              LOC-PART  HYPHEN  EN-MARK  ENE-EXT ;

     LOC-PART = HYPHEN  LOCANT  LOC-SEQ ;

     LOC-SEQ = COMMA  LOCANT  LOC-SEQ , $ ;


ROOTSYMBOL   NAME
```

Figure 1. Simple SLR grammar for hydrocarbons.

LR parsers accept an input stream of grammar symbol tokens and use a stack and a parsing table to control their operation. The input tokens are derived from the raw source by lexical analysis, and in principle the stack contains a string

$$S_0X_1S_1X_2X_2S_2 \ldots X_nS_n$$

where S_n is at the top of the stack. Each X_i is an input token and each S_i is a state. It is not always necessary to have the input tokens on the stack, and they are not included within the nomenclature system.

Intuitively, each state corresponds to the parser being at a certain point in the processing of a rule of the grammar: as parsing proceeds, the parser moves between states. The current state value is used as one index to the parsing table, of which there are two parts, an Action table and a Goto table. A further table summarizes the grammar rule lengths and their left-side nonterminal symbol codes.

Figures 1–3 illustrate a simple grammar for unbranched hydrocarbons and its associated parser tables. It is not feasible in the space available here to demonstrate the operation of the technique on a meaningful subset of the actual grammar described in part 2.[2] Therefore, the example given here is illustrative only. Grammars accepted by the SLR parser generator are formed into three sections (see Figure 1):

(a) Terminals. These are the terminal symbols of the grammar, each representing possibly a number of different actual input morphemes (e.g., ROOT could be meth, eth, prop, etc.).

(b) Rules. The syntax rules of the grammar are presented in the form

$$L = a, b, c;$$

	A-MARK	EN-MARK	ANE-MARK	ENE-MARK	YNE-MARK	ROOT	LOCANT	HYPHEN	COMMA	MULT	E-O-I
0			S 28	S 26	S 11						
1											A
2						S 3					
3								S 5			
4											R 1
5							S 6				
6								R 13	S 8		R 13
7								R 12			R 12
8							S 9				
9								R 13	S 8		R 13
10								R 14			R 14
11						R 7		S 5		S 19	
12						R 3					
13								S 14			
14		S 15									
15						R 5				S 17	
16						R 8					
17	S 18										
18						R 6					
19	S 25							S 5			
20						R 9					
21								S 22			
22		S 23									
23						R 5				S 17	
24						R 10					
25						R 11					
26						R 5				S 17	
27						R 4					
28						S 29					
29											R 2

Figure 2. Action table for the grammar of Figure 1.

	NAME	SAT-SEQ	ENE-EXT	YNE-EXT	YNE-EN	LOC-PART	LOC-SEQ
0	1	2					
1							
2							
3						4	
4							
5							
6							7
7							
8							
9							10
10							
11				12		13	
12							
13							
14							
15			16				
16							
17							
18							
19					20	21	
20							
21							
22							
23			24				
24							
25							
26			27				
27							
28							
29							

a

Rule Number	Leftside Nonterminal	Rightside Length
1	NAME	3
2	NAME	2
3	SAT-SEQ	2
4	SAT-SEQ	2
5	ENE-EXT	0
6	ENE-EXT	2
7	YNE-EXT	0
8	YNE-EXT	4
9	YNE-EXT	2
10	YNE-EN	4
11	YNE-EN	1
12	LOC-PART	3
13	LOC-SEQ	0
14	LOC-SEQ	3

b

Figure 3. Goto table (a) and rule table (b) for the grammar of Figure 1.

where L is a single nonterminal symbol and the various sequences a–c, etc. are sequences of terminal and nonterminal symbols, which are the rule alternatives of L. The various alternatives for L are separated by commas, and the final alternative is followed by a semicolon.

(c) Rootsymbol. The rootsymbol of the grammar denotes the complete utterance of the language (sentence, name) with which the parser generator begins to analyze the grammar and construct the parser tables. It is the goal symbol of the parser when analyzing a name.

Thus, in Figure 1, there are 10 terminal symbols and 14 rule alternatives for 7 nonterminal symbols. The $ symbol as a rule alternative indicates that the rule may be null; that is, its corresponding fragments need not appear in a name at all.

Figure 2 presents the Action table for the grammar of Figure 1, showing what the parser must do for each state and input terminal symbol [including end-of-input (E-O-I)]. Where no action is specified, an input symbol is not valid in the given state; otherwise, A indicates that a complete input sequence may be accepted as valid, S n denotes adding the symbol to the Stack and moving to State n (technically called Shifting), R n denotes Removing symbols from the stack according to Rule n (Figure 3b) and then adding the left-side symbol of that rule to the stack and moving to a state given by the Goto table (Figure 3a), as if that left-side symbol had been directly input for the state in force when the first actual symbol of the rule was received (technically called Reducing). This process is illustrated in Figure 4 for the input sequence "2-butene".

Lexical Analysis. The parsing process depends upon the ability to determine correctly which terminal symbol corresponds to each morpheme of the input string and to identify those morphemes correctly as well. For the typical modern programming language, an SLR-based parser is able to process an input program in a single left to right pass, since each terminal symbol can be determined uniquely from well-

Stage	Stack	State	Input	Action	Leftside	Length	Old State	Goto
0	<empty>	0	ENE-MARK (ene)	S 26				
1	ENE-MARK 0	26	ROOT (but)	R 5	ENE-EXT	0	26	27
2	ENE-MARK 0 ENE-EXT 26	27	ROOT (but)	R 4	SAT-SEQ	2	0	2
3	SAT-SEQ 0	2	ROOT (but)	S 3				
4	SAT-SEQ 0 ROOT 2	3	HYPHEN (-)	S 5				
5	SAT-SEQ 0 ROOT 2 HYPHEN 3	5	LOCANT (2)	S 6				
6	SAT-SEQ 0 ROOT 2 HYPHEN 3 LOCANT 5	6	E-O-I (end)	R 13	LOC-SEQ	0	6	7
7	SAT-SEQ 0 ROOT 2 HYPHEN 3 LOCANT 5 LOC-SEQ 6	7	E-O-I (end)	R 12	LOC-PART	3	3	4
8	SAT-SEQ 0 ROOT 2 LOC-PART 3	4	E-O-I (end)	R 1	NAME	3	0	1
9	NAME 0	1	E-O-I (end)	A				

Figure 4. Example SLR parse for the name 2-butene.

delimited units of the program text. However, chemical nomenclatures differ from programming languages in this respect, since they generally contain few delimiting characters, in particular between letter-based morphemes. Whereas the space is used frequently in programs to separate adjacent symbols, chemical names contain strings of directly concatenated letters that represent a number of functionally different morphemes.

This lack of delimiters necessitates a method for recognizing morphemes within the input character string and associating each morpheme with a corresponding terminal symbol. To this end, the SLR parser generator has been extended to allow relevant morphemes to be listed against each terminal symbol in the overall grammar. These morphemes are then processed to construct a dictionary table that associates each morpheme with one or more terminal symbols as specified. The parser then reads this table to build a tree-structured dictionary before accepting names for analysis and translation.

For each valid morpheme within the tree dictionary, there is a pointer to a list of appropriate terminal symbol codes. On input of a name to the parser, the lexical analysis phase matches input characters against the tree to find the largest available morpheme. Terminal symbol codes for this morpheme are then checked against the Action table to find one that is valid. If no symbol is valid, a shorter morpheme may be chosen instead.

Backtracking. The complexity of the nomenclature is such that the parser may reach a point at which no valid morphemes may be available, although alternative shorter morphemes could have been chosen earlier. Since the entire name is absorbed before analysis begins, it is possible to backtrack the parser to such an earlier choice and then advance on a different path.

As a simple example, the morphemes in the name iododecane are "iodo", "dec", and "ane". "dodec" is also a valid morpheme in the dictionary and will be found at the same time as "dec" (we are parsing the name from right to left), but as "dodec" is the larger morpheme, it will be chosen in preference to "dec". The parser will then require further input from the lexical analyzer that will find "io" in the input stream. There is no entry for "io" in the dictionary, and so the parse fails and backtracks to find an alternative path. "dec" is an alternative morpheme that leads to the valid morpheme "iodo", and thus the name parses correctly.

Normally an SLR parser is deterministic; that is, it does not backtrack. For chemical nomenclature, due to the established nature of the morphemes, there may be a number of alternative terminal symbols that could be used as input to the parser. It has proved possible to augment the SLR method to allow backtracking and so overcome the lack of delimiters within chemical nomenclature.

If a name is input that is incorrect according to the IUPAC syntax, or that is correct but not known in the language described by a partially developed grammar, then the parser will enter an error state. The parser will attempt to backtrack and to retry alternatives but will ultimately fail, leaving the parser at the right-hand end of the name with no further alternatives to try. The name will be rejected at this point, and it has been found empirically that by marking out the leftmost position reached that led to an error state, an indication is given as to the type of error in the name. Therefore, a record is kept of the maximum extent of the parser through the input string, and this is used to suggest the position and extent within the name of the fragment that led to the error.

Where the parser successfully reaches the end of the input name, and the Action table indicates the name is acceptable, the sequence of terminal symbols identified for the name is recorded as a (syntactically) valid interpretation, and semantic processing can begin.

SEMANTICS

Semantic Information and the Grammar. Semantic information is held within the grammar alongside each appropriate morpheme. A morpheme in the dictionary has one of four types of information associated with it: (1) information relating to structure and given by one or more CCT records; (2) an integer that may be the representation of the length of a chain, a multiplication factor for a multiplying prefix, or the locant value for a Greek locant, depending on the interpretation of the morpheme; (3) an integer that specifies the atomic number of an atom entry for which a CCT record must be constructed when needed by using a copy of the integer value in the SIZE field of the record; (4) a null entry, the fragment having no structural meaning.

For example, the morpheme "but" that is used to construct butane has an integer value 4 associated with it. At a later stage the value is extracted and expanded by the appropriate semantic code to the CCT fragment "1 1 4 0". Similarly, the morpheme "chlor" is associated with the value 17, which is incorporated by different semantic actions into the CCT fragment "1 2 17 0". Some morphemes, for example, hyphens, commas, and parentheses, have no structural meaning in

themselves and so have no semantic information associated with them.

The semantic information associated with each of the morphemes only becomes available for semantic processing if the terminal symbol that represents the morpheme is shifted onto the parser's stack during the parsing of a name. As parsing may involve a number of backtracks and retries, the construction of the CCT is held back until the final path has been found. This path is followed again without the need to backtrack, and the semantic information is extracted to enable the building of the CCT. This semantic information is extracted from the dictionary during the Shift actions and then manipulated during the Reduce actions of the parser. Thus, the construction of the CCT has a significant influence on the way the syntax rules are written in that the Shift and Reduce actions take place in an order which ensures that information is not lost.

The Pascal code associated with each production rule alternative was placed into the grammar alongside the appropriate rule alternative and executed when the rule is processed. The rules are numbered by the SLR parser generator, and these numbers are used as Pascal case selector values. The SLR parser generator was amended to extract the Pascal code and put it into a file with the correct rule numbering ready for automatic inclusion into the main parse program on compilation. On any subsequent change to the grammar, the code relating to rules already present does not need to be changed, and on generating a new parsing table, the semantic rules are automatically renumbered and the case limbs relabeled.

Constructing the CCT. A chemical name is hierarchical with its substructures preceding the parent term. Each substructure in turn may have its own substructures that precede it. For example

6-(1-chlorobutyl)-8-(2-iodopropyl)tridecane

has the parent structure tridecane, with two substructures attached to it, butyl and propyl. Each of these substructures has a single substructure or substituent, chloro on the butyl and iodo on the propyl. This hierarchy extends from the rear of the name forward.

The same hierarchy is seen in the CCT for the above compound:

LOCT	TIPE	SIZE	SUBS
1	1	13	2
6	1	4	1
1	2	17	0
8	1	3	1
2	2	53	0

The first entry represents the aliphatic chain (TIPE = 1) tridecane (SIZE = 13) and has two substituents (SUBS = 2). The first substituent, the second entry, on carbon 6 of the parent (LOCT = 6) is the aliphatic chain butyl with a single substituent given by the following entry. Thus, the chloro, an atom of atomic number 17 (TIPE = 2, SIZE = 17), is on position 1 of its parent and has no substituents. Similarly, the final two entries represent the second complex substituent of the parent.

The hierarchy can be nested to many levels and may result in any number of CCT entries. It is far easier to construct the CCT from the top down, by parsing the name from right to left, than from the bottom up. However, the hierarchy in the CCT refers to parent and its immediate substructures rather than the right to left order in the name. Substructures appear in the CCT in locant order so, in the example given above, the chlorobutyl substructure with locant 6 appears in the CCT before the iodopropyl substructure (see Sorting the CCT).

As the CCT is being constructed, some checks can be made as to the structural validity of the name that has been input. It is not feasible to attempt to trap all the errors by syntactic means. The final grammar[7] as given in part 2 will accept the name pent-1,3-triene as a syntactically correct name, although it is not valid on two counts. First, the number of locants does not match the multiplying term and, second, an "a" is not present after the aliphatic stem.

The first error can be uncovered in the building of the CCT. Having shifted the terminal symbol that represents the morpheme "tri-mark", the numerical value of the multiplying term is recorded. If no multiplying term is present, the value 0 is recorded. On reading the locants, a count is kept of the number of locant values, and at the end of a locant sequence this total is compared with the original multiplying value. If they are not equal, an error flag is raised.

The nonpresence of the "a" in the name is allowed by the rule alternative

a-part = a-mark, $;

The semantic code for the rule "a-part = a-mark" requires explicitly that an "a" is present within the name and will give a warning message if the recorded multiplying value is 0. Alternatively, the code associated with the rule "a-part = $" will give a warning if this value is not 0.

Checking the Validity of the CCT. Two essential checks must be done on the CCT to confirm that the structure represented is chemically valid. First, the locants of substituents and modifications should be within the bounds of their parent structure and, second, the maximum valency of each atom within the structure should not be exceeded.

The above two checks require the examination of each of the parent structure CCT main entries with their respective substituent and modification entries. While as discussed above there is an implied hierarchy in the CCT, there is a problem in finding the position of the CCT main entries for substituents at the same hierarchical level, since their position is dependent upon entries for substituents and modifications of preceding substituents of the parent.

A means of simplifying semantic checks is to state the hierarchy of the chemical name explicity, so that each main entry is directly referenced from the main entry immediately above it in the hierarchy, rather than the implied hierarchy of the CCT. A semantic tree has been developed that allows the explicit description of the hierarchy of a chemical name using CCT fragments. It is the semantic tree, rather than the CCT directly, that is constructed by the second pass of the parser through the name.

The semantic tree consists of nodes arranged in the appropriate hierarchy, each of which points to CCT records to describe the actual structure. Each semantic tree node has seven fields, three of which are concerned with bidirectional linkage within the tree. The remaining four fields are pointers to groups of CCT entries concerned with the parent and any unsaturations, heteroatoms, and functional groups. The root of the semantic tree is a node that contains a pointer to a CCT main entry record for the parent compound. In the simplest of all semantic trees, the parent pointer points to the CCT main entry record of the parent compound with all the other pointers being NIL. Figure 5 shows the semantic tree for a name with several levels of nesting of substituents and containing unsaturations, namely

4-bromo-7-[1-(1-chloroethyl)-2-methylprop-1-enyl]undeca-1,5,9-triene

With the semantic information organized in this tree, the checking of locant values and correct valencies is straightforward. All the CCT main entries are in known positions, as are the substituent and modification entries that belong to them.

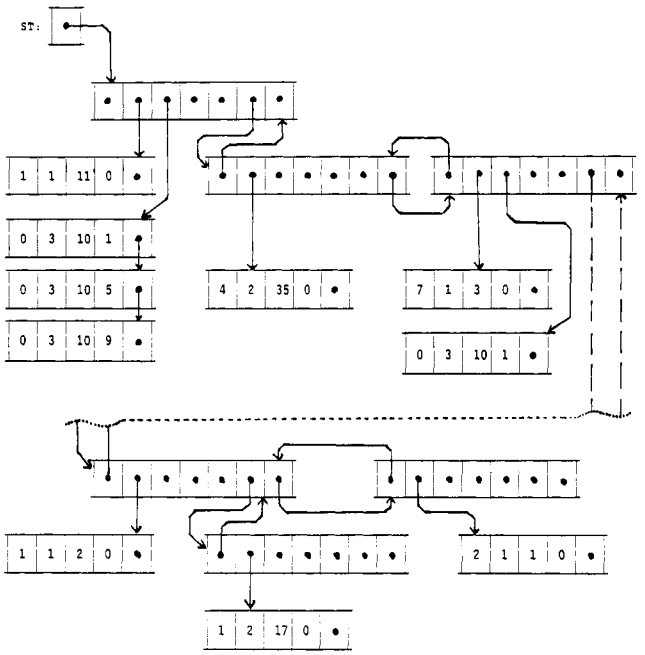

Figure 5. Example semantic tree for the name 4-bromo-7-[1-(1-chloroethyl)-2-methylprop-1-enyl]undeca-1,5,9-triene. Semantic tree node fields are, from left to right, PREV, PARENT, UNSATURATED, HETERO, SEFG, SUBSTITUTED, and NEXT, as outlined in the text. CCT records have fields LOCT, TIPE, SIZE, SUBS, and NEXT, from left to right.

In most cases the maximum locant value allowed by a parent structure or substructure is given by the CCT entry for that structure, but there are some cases where the size of the structure cannot be deduced from the normal CCT entry alone. For example, the size of a bridged alicyclic system is given by the addition of the SIZE fields of the main entry record with the SIZE fields of the subsequent chain–ring segment entries. These CCT entries will be clearly grouped together below a single semantic tree node for the parent structure.

The locants of the substituents may be found in the LOCT field of the main entry records. For bond modifications, the locant values are found in the SUBS field except for a double-bond modification on a bridge head where both the start and end locants are given in the following CCT entry. Hence, it is a simple task to check actual locant values against the range permitted by the parent structure.

The semantic tree allows the value of the SUBS field of each main entry to be calculated from the number of substituents and modifications on that entry. Before this structure was adopted, a count had to be kept of the number of substituents and modifications being added to each of the main entries as the CCT was being constructed.

By knowing the maximum valency of each atom type, the validity of substitutions may be checked. For each of the parents and substructures that may be substituted, assumptions have been made about the number of substituents allowed at each locant position. The actual number of substituents present can easily be checked within the semantic tree.

Alphabetic Ordering of Substituent Prefixes. The ordering of the substituent prefixes within a chemical name is important. If a chemical compound is to have a single systematic name, then there must a methodology to order the prefixes. In the first edition of the IUPAC organic chemical nomenclature rules this was governed by the complexity of the substituents. This has been replaced by the alphabetic ordering of the substituents, the rules governing the ordering being Rules C-16.1–3 in the Blue Book.[8]

While alphabetic ordering is a simple matter, the question of what characters of the substituent prefixes it is applied to is not. Simple substituent prefixes are arranged alphabetically, with multiplying affixes (if any) being added later. These do not therefore alter the alphabetic order of the prefix. Prefixes for subsubstituents are arranged in alphabetic order among themselves in the substituted substituent in the same way as are the substituents of a parent compound; then the complete names of the substituted substituents are treated as single, complex prefixes to the name of the parent compound. However, complex substituents are considered to begin with the first letter of the complete name of the prefix. If that begins with a multiplying term, then it is alphabetically ordered according to the first letter of that term. For example, the correct alphabetic ordering is

3,5,9-trichloro-6-(2,2-dibromobutyl)hexadecane

where the "c" in trichloro comes before the "d" of dibromo.

A check for the correct alphabetic ordering of substituent prefixes in an input name is done in the semantic phase. A warning is given if the order is incorrect, though the order of the entries in the eventual CCT and the meaning of the name are unaffected.

The name is stored in an array on input to the nomenclature translator. During the second pass through the parser, the indices to the prefixes within this array are extracted. Since the correct order of the prefixes is dependent upon the hierarchy within the name, alphabetic checking relies upon similar information as is used in constructing the semantic tree. In practice, the checking is facilitated by the separate construction of a substituent tree with two types of node to represent simple and complex substituents. Fields in each node hold indices to the start of the locant sequence and the first and last characters of the substituent in the name array, a pointer to the next node at the same hierarchical level, and, in the case of complex substituents, a pointer to a node for the first of any subsubstituents.

During the second pass through the name by the parser, a root node for the substituent tree is created when the parent structure is processed. Substituents and their locants, which may be made up of a number of morphemes, are recognized and represented as appropriate by simple and complex nodes. When the tree is complete, comparisons can be made on nodes at the same hierarchical level, starting from the deepest level and working upward until the root node is encountered.

When two simple nodes are compared, it is possible to detect two identical substituents and indicate to the user that the name should be rearranged with insertion of a multiplying term and removal of the duplicated substituent. If incorrect ordering is found, a warning message is given with identification of all terms involved. A comparison of two complex substituents checks first the alphabetic ordering, giving a warning message if the order is incorrect. If the two are identical, then the locants are checked and the substituent with the lower locant appears first in the name.

For the comparison of a simple and a complex substituent, the alphabetic characters are considered. If they are the same up to the point where the simple substituent ends and the complex one continues, then the simple one has precedence. Otherwise, the substituent with the lower character at the first point of difference has precedence. For example, chloro should come before chloromethyl, which precedes chlorosyl.

Sorting the CCT. The final step before a CCT is delivered by the semantic phase is to impose a canonical order. While there is already an order arising from the hierarchy of the table, the specific order of locants and of substituents at the same level depends upon the order in which they appear in the name. However, locants not in numeric sequence and substituents not in alphabetic order do not affect the meaning of the name. It is desirable, though, to generate only one CCT for variations on the same name in order to draw one corre-

sponding structure and to compare different structures by using the CCT.

The CCT entries are ordered by using the semantic tree. The groups of entries corresponding to unsaturations, heteroatoms, and functional groups on one parent structure are sorted by locant order and output directly after the CCT entries for the parent structure represented by that semantic tree node. Each substituent, which may have subsubstituents as indicated by the semantic tree hierarchy, is then sorted first by locant order and then, for entries with the same locant, on the TIPE field. This brings substituents with the same locant and type (chain, rings, atom) together, and these are finally sorted on the SIZE field to order them from smallest first to largest last.

DISCUSSION AND CONCLUSIONS

This paper has described the processing carried out by the parser software in analyzing the syntax and semantics of IUPAC systematic nomenclature for some classes of organic compounds. The software has been successfully implemented in Turbo-Pascal and used within the nomenclature to structure diagram translator, which runs on an IBM PC-XT or compatible microcomputer. It confirms that the grammar-based approach using techniques developed for processing computer-programming languages can be applied to other less artificial languages. Two modifications were necessary, though, to these techniques. Most importantly, backtracking had to be introduced in the syntax analysis phase. Second, reference to a dictionary of valid morphemes was essential in the lexical analysis phase because of the lack of delimiters in chemical nomenclature.

In part 2,[2] a comment was made about the difficulty of developing a formal grammar from nomenclature rules as described in the Blue Book.[8] In this paper a good illustration of the complicated processing necessary to implement the current rules has been given. The rules for the alphabetic ordering of substituents are not only complex but also inconsistent as to which characters alphabetic ordering is applied. This depends upon whether the substituent is simple or complex. It necessitated the implementation of a data structure specifically for this one task, with code to detect simple and complex substituents and perform different processing on the two types.

ACKNOWLEDGMENT

We gratefully acknowledge funding of this research by the Laboratory of the Government Chemist.

REFERENCES AND NOTES

(1) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 1. Background and Introduction. *J. Chem. Inf. Comput. Sci.* (first of three papers in this issue).
(2) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 2. Development of a Formal Grammar. *J. Chem. Inf. Comput. Sci.* (second of three papers in this issue).
(3) Aho, A. V.; Ullman, J. D. *Principles of Compiler Design*; Addison-Wesley: Reading, MA, 1977.
(4) Thomas, M. I. *A Basic SLR Parser Generator*; Report No. 80/1; Department of Computer Studies, University of Hull: Hull, England, 1980.
(5) Lynch, M. F.; Harrison, J. M.; Town, W. G.; Ash, J. E. *Computer Handling of Chemical Structural Information*; MacDonald-American Elsevier: London, 1971; Chapter 6.
(6) Rayner, J. D. A Concise Connection Table Based on Systematic Nomenclatural Terms. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 108-111.
(7) See ref 2 Appendix.
(8) International Union of Pure and Applied Chemistry. *Nomenclature of Organic Chemistry, Sections A-F and H*; Pergamon, Oxford, U.K., 1979.

Canadian Scientific Numeric Database Service†

GORDON H. WOOD,* JOHN R. RODGERS, and S. ROGER GOUGH

Canada Institute for Scientific and Technical Information, National Research Council of Canada, Montreal Road, Ottawa, Canada K1A 0S2

Received November 14, 1988

Modern computer systems and telecommunications networks are being harnessed in increasingly innovative ways to deliver evaluated scientific/technical numeric data to the desk or laboratory bench. As an example of this development, the Canadian Scientific Numeric Database Service (CAN/SND) is described. CAN/SND provides international online access to factual databases in crystallography, molecular biology, spectroscopy, and chemical thermodynamics. In addition, CAN/SND carries out research in data storage, retrieval, and analysis techniques. The paper gives a description of the databases currently available, examples showing the variety of scientific questions that can be answered, and an outline of plans for virtually linking related databases for interdisciplinary searching.

I. INTRODUCTION

This paper describes the Canadian Scientific Numeric Database Service (CAN/SND), what it is, what it offers, and where it plans to go. All readers are almost certainly familiar with the computer as a tool for performing calculations and automating measurements; many are aware that computers may be used to search large bibliographic databases. Probably relatively few think of the computer as a means of accessing evaluated scientific data from the international literature. It is this latter use that will be highlighted here.

Immediately following, section II defines some basic terminology and gives some perspective on scientific numeric database systems. Section III reviews the background to CAN/SND and the databases currently offered. Section IV

† Presented at the Herman Skolnik Award Symposium on Scientific Numerical Databases—Present and Future, sponsored by the Division of Chemical Information of the American Chemical Society at the Third Chemical Congress of North America, Toronto, Canada, June 7, 1988.

* Author to whom correspondence should be addressed.

0095-2338/89/1629-0118$01.50/0 Published 1989 by the American Chemical Society

Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 4. Concise Connection Tables to Structure Diagrams

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

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

Received October 12, 1989

The concise connection table (CCT) is the structure representation into which IUPAC systematic organic chemical nomenclature is translated by the methods described in previous papers in this series. Here, first, some enhancements to the original specification of the CCT are presented, resulting from experience of its use and extension of the functional groups and classes of names handled by the nomenclature translator. Then, software for the expansion of the CCT into an atom table is described which enables the structure to be displayed by character graphics. The conversion of the CCT into the Standard Molecular Data (SMD) format is reported, and this has enabled the molecular structure corresponding to input names to be displayed by a variety of graphics packages. The problems of linking the nomenclature translator and commercially available molecular graphics and database packages into one multiprocess software system under MSDOS on IBM PCs and compatibles are discussed.

INTRODUCTION

Previous papers in this series[1-3] have described the development of a formal grammar for IUPAC systematic organic chemical nomenclature and the production of a parser program to recognize valid names. The parser program also does semantic processing and generates a concise connection table (CCT) as a representation of the molecular structure of the corresponding name. The CCT was created explicitly for representing the structure of nomenclatural fragments.[4]

The current state of development of the CCT is outlined in the following section since a knowledge of the extensions is necessary in understanding how a structure diagram is produced from a CCT. The CCT can be converted into full atom-by-atom connection tables which provide a link not just into molecular graphics programs but also into a wide range of software requiring a description of chemical structure as input.

The production and display of molecular structure diagrams from CCTs completes the translation path from names to diagrams first discussed over 25 years ago by Garfield and republished in 1985.[5] Alternative approaches to this path were addressed in paper 1,[1] and quite recently a further development has resulted in software that recognizes Beilstein nomenclature and generates corresponding structure diagrams.[6]

ENHANCEMENTS TO THE CONCISE CONNECTION TABLE

Since the publication of the original specification of the CCT[4] a number of extensions and enhancements have been made as a result of several years' experience in its use. The chief motivation behind the development of the CCT was to provide an internal representation of chemical structure corresponding to the semantic decomposition of systematic nomenclature. The CCT is a linear table which represents implicitly the hierarchic structure of a molecule, as determined from the IUPAC nomenclature. Each CCT entry has four nonnegative integer fields, LOCT, TIPE, SIZE, and SUBS.

Essentially a nonzero LOCT field shows where a given substructure, described by this CCT main entry, is attached to its parent (sub)structure, described earlier in the CCT. TIPE 0 denotes an aromatic ring, 1 a chain, 2 a single atom, and 3 an alicyclic system, and SIZE gives the ring system or chain size or atomic number, respectively. Ring systems are fully described by additional ring segment entries for each ring component. SUBS indicates the number of substituents on the (sub)structure, these being described in subsequent CCT entries arranged in an implicit hierarchic order. A CCT entry with zero in the LOCT field is either a special entry having TIPE 0 or a modification entry where TIPE 1 indicates an electronic charge of value SIZE minus eight, TIPE 2 indicates a replacement heteroatom determined by SIZE, TIPE 3 represents a nondefault bond coded by convention in SIZE, and SUBS gives the locant of modification to the parent structure.

The principle underlying the CCT has been maintained in the introduction of additional CCT entry types for functional groups and steroid nomenclature. Further CCT modification entry types have been defined to represent certain stereochemical bond orientations and to improve the representation of multiple bonds in the bridges of polycyclic ring systems.

Special Entry for Functional Groups (SEFGs). The original CCT definition contained a number of points where extensions to the scheme could be introduced. One such was the area of special entries, characterized by LOCT and TIPE fields both having the value zero. To allow the simplified display of functional groups as conventional line formula segments with explicitly stated hydrogens [e.g., -COOH rather than -C(OH)=O or similar alternatives], the common groups are associated with a nonzero integer value held in the SIZE field of the SEFG (Table I). The locant at which the functional group is attached to its parent structure is then recorded in the SUBS (substitution) field, following the CCT convention of using this field as an alternative to LOCT in appropriate specific circumstances.

For example, from Table I it can be seen that decane-1,4-diol should now be represented as

LOCT	TIPE	SIZE	SUBS
1	1	10	2
0	0	13	1
0	0	13	4

rather than

LOCT	TIPE	SIZE	SUBS
1	1	10	2
1	2	8	0
4	2	8	0

Similarly, *m*-nitrotoluene now has the CCT

LOCT	TIPE	SIZE	SUBS
1	0	1	2
1	0	6	0
1	1	1	0
0	0	20	3

0095-2338/90/1630-0122$02.50/0

rather than

LOCT	TIPE	SIZE	SUBS
1	0	1	2
1	0	6	0
1	1	1	0
3	2	7	2
1	2	8	1
0	3	10	0
1	2	8	1
0	3	10	0

This latter example illustrates the increased brevity provided in the concise connection table through the use of SEFGs, including the avoidance of explicit description of the sometimes uncertain, conjugated bonding arrangements within functional groups.

Steroids and Stereochemical Bond Orientation. In extending our grammar of IUPAC organic nomenclature[2] to include aspects of steroid nomenclature, one consideration was the common feature of the steroid nucleus ring system and its particular locant-numbering arrangement. To preserve this high-level complexity within the CCT scheme, a new TIPE 5 main entry has been introduced which leads into a substantial area of extension.

A detailed discussion covering all aspects of steroid nomenclature handling from grammar development through structure diagram display will be presented in the next paper.[7] In brief, however, the TIPE 5 main entry (or steroid header entry; the digit 5 is mnemonically reminiscent of S for steroid) introduces a number of further entries (counted in the steroid header SIZE field) that describe details of the stereochemistry of the ring system and its standard chain substituents. The sizes of the four rings A, B, C, D of the steroid nucleus (typically 6 6 6 5) are given in the four fields of a partner CCT entry which always follows immediately the TIPE 5 main (steroid header) entry.

The introduction of facilities for handling steroids has led to the definition of new modification entry codes for bond stereochemistry. The original CCT definition provided for bonds other than those assumed by default according to the TIPE of a substructure, by means of the TIPE 3 modification entry. This used the SIZE field to denote a bond type by conventional coding, which has now been extended (Table II) to include designations of α, β, and ξ bonds as interpreted in the steroid context. Further codings are being considered to handle more general stereochemical features, by use of presently unallocated code values.

Polycycle Bridge Unsaturations. Ring systems are described in detail by ring segment entries (RSE) which follow a TIPE 0 or TIPE 3 main (ring header) entry. A bridge in a polycycle is represented by an RSE of TIPE 1, whose SIZE field contains the number of atoms in the bridge, which may be zero. The two points of attachment of the bridge segment to its parent system are given by the LOCT and SUBS fields, the LOCT field containing the lower valued locant and SUBS containing the offset (numeric difference) between that and the higher valued locant. (RSEs appear in the linear sequence of the CCT in order of ascending final locants, so that each new segment of the structure may be placed relative to atoms already introduced, and the locants ascribed to the atoms of the new segment are correct in the context of the final overall structure.)

As with aliphatic chains, the default bond type in a TIPE 3 ring system is covalent single, and variations from this are represented through TIPE 3 modification entries. Usually, the lower of two adjacent-valued locants of a bond modification is given in the SUBS field of the modification entry, and a SUBS locant value of zero is used to indicate a bond modification in the attachment of a substructure to its parent.

Table I. SIZE Codes for Functional Group Special Entries (SEFGs)

SEFG SIZE Code	nomenclature	line formula
1	amino	$-NH_2$
2	azido	$-N_3$
3	carbaldehyde	-CHO
4	carboxylic acid	-COOH
5	chlorosyl	-ClO
6	chloryl	$-ClO_2$
7	cyanato	-OCN
8	cyano	-CN
9	dihydroxyiodo	$-I(OH)_2$
10	dithiocarboxy	-CSSH
11	dithiosulfo	$-S_3OH$
12	hydroperoxy	-OOH
13	hydroxy	-OH
14	iodosyl	-IO
15	iodyl	$-IO_2$
16	isocyanato	-NCO
17	isocyano	-NC
18	isothiocyanato	-NCS
19	mercapto	-SH
20	nitro	$-NO_2$
21	nitroso	-NO
22	pentafluorothio	$-SF_5$
23	perchloryl	$-ClO_3$
24	seleneno	-SeOH
25	selenino	$-SeO_2H$
26	selenono	$-SeO_3H$
27	sulfino	$-SO_2H$
28	sulfo	$-SO_3H$
29	thioaldehyde	-CHS
30	thiocarboxy	-CSOH
31	thiol	-SH
32	selenol	-SeH

Table II. TIPE 3 Modification Entry Bond Codes

SIZE code	bond type	SIZE code	bond type
0	ξ	8	aromatic
1	α	9	single
2	β	10	double
6	polycycle bridgehead	11	triple
7	dative		

However, in the case of polycycle bridges, the locants of the bridgehead and the bridge end atoms are generally nonadjacent numerically and an alternative mechanism is necessary for these (relatively rare) situations.

The mechanism is indicated by the use of bond modification code 6 (see Table II) in the case of a double bond from bridge to bridgehead, and the SUBS field contains zero. An additional CCT entry then follows to give *both* locants involved, in the LOCT and SUBS fields, with the TIPE and SIZE fields unused and set to zero. This method represents a further instance of the CCT philosophy whereby common situations are represented in a few fields, with default assumptions where necessary. Relatively rarer or more complex situations use progressively more space and are introduced by specific "flag values" in designated fields of the simpler formats.

CONVERSION TO ATOM-BY-ATOM TABLES

In contrast with the CCT, most other connection table schemes are fully explicit in listing all the (non-hydrogen) atoms and bonds in an organic compound, with connectivity information for each atom.[8] Such connection tables are much used in chemical information systems for structure storage and retrieval.[8,9] Another use is as the starting point for the display of the molecule, in which case the coordinates of each atom may also be held in the table.[10] However, there are other situations in which a fully explicit connection table is not necessary and alternative concise schemes and notations have

been proposed.[11] Thus, it is the use for which a particular connection table is required that determines its complexity.[8]

Software for the expansion of the CCT into a full atom-by-atom connection table was therefore desirable, in particular to aid the eventual display of structure diagrams but also to permit the interfacing of the nomenclature translation software to a variety of chemical information systems that take connection table input.

Atom Table. A linked-list data structure has been implemented to store an atom table containing information about the non-hydrogen atoms within a molecule in the locant order resulting from the sequential processing of the CCT.

The topology of an organic molecule is predictable from the TIPE field of the parent structure in the CCT, namely, aliphatic chain, aromatic or alicyclic ring system, or individual atom. Procedures have been written to expand each of these parent structures. Thus, for aliphatic chains the SIZE field gives the length of the chain, enabling entry of the appropriate number of carbon atoms into the atom table. Display coordinates for each atom can also be generated, if required, from the positions of the preceding atoms according to the context inferred from the CCT. Initially, it is assumed that all bonds are single bonds.

For aromatic ring systems, each ring fusion, detected when a further ring segment entry has to be processed, requires an amendment to connectivity information for two atoms already in the atom table. The LOCT field for the additional RSE identifies the first of these, relative to the first atom of the whole ring system (i.e., the current parent). New atoms are inserted for the additional ring segment, taking account of the orientation of the fusion position, until the second fusion atom is reached, as indicated through the SUBS field. From the information given in the CCT, it is not possible to decipher directly the double-bond coordination in an aromatic system, so all the bonds are identified as aromatic.

Alicyclic ring system entries are similarly processed by using the SIZE field for overall size and LOCT and SUBS fields to detect any spiro attachments or bridges, as described previously.

The nature of an individual atom parent is determined from its SIZE field, which gives the atomic number. Coordinates can be computed by assumptions of standard bond lengths and angles.

The sequential processing of the CCT means that substituents and/or modifications to a parent structure are dealt with later, requiring changes to connectivities, bond, and atom types for atoms already inserted in the atom table. Before terminating, each parent structure procedure makes calls to a further, general procedure to process any substituents and modifications on that parent structure as indicated in the SUBS field of the parent entry. Parameters passed at each call include the locant of the substituent or modification, known from the LOCT field of the corresponding CCT entry and the pointer to the first atom of the current parent in the partially constructed atom table.

The substituents themselves are handled by a general substituents and modifications procedure with calls of each appropriate parent structure procedure to enter further atoms in the table, according to the TIPE field of the substituent CCT entry. These procedures are therefore mutually recursive through the general substituent procedure, which enables the substituents-on-substituents situation to be handled correctly.

Modifications to bonds or atoms (i.e., heteroatoms) and special entry functional groups, whose structure is known and cannot be substituted or modified, are dealt with by further separate procedures.

Standard Molecular Data (SMD). The SMD format has resulted from the Computer Assisted Synthesis Planning (CASP) project run by a consortium of seven German and Swiss chemical companies.[12] It was developed to provide a common interface for the exchange of molecular data between various chemically orientated systems rather than to provide a means of permanent storage.

An SMD file is a sequential ASCII text file containing one or more SMD structures. Each SMD structure contains all the relevant information for a specific compound or reaction and is divided into information (or main) blocks. Each block contains the relevant data for a particular property associated with the compound or reaction. Blocks are designated for structure name, connection table (CT), Cartesian coordinates (CO) and lables (LB), among others. It is the responsibility of the computer program, taking an SMD file as input, to select only those information blocks that are necessary for its operation.

Hierarchic structuring of certain information blocks is provided by the use of subblocks (in the CT, CO, and LB blocks) and superatoms. A superatom can be used, for example, to represent a fragment that occurs more than once within a molecule, thus eliminating the need for repetition. Data records are used to provide explicit information and are located at the lowest level in the information hierarchy.

We have developed software to produce the SMD file format directly from the CCT. The atom table described previously was designed for the display of structures by use of our own character graphics technique (see next section). Hence, it contains information specific to that task and is not easily transformed to the SMD format.

Once a valid CCT is produced by nomenclature translation, the SMD procedures can be called to create an SMD file containing the following blocks:

DTCR	creation date and time of the structure file
CT	connection table of the structure
CO	corresponding coordinates
FORM	empirical formula
NAME	name of compound

It is the CT block that is generated from the CCT and is described here. At present, the CO block contains character-based coordinates extracted from the atom table, although, in time, these will be replaced with display coordinates computed by other procedures.

Two linked-list data structures, namely, the atom and bond lists, were designed to allow construction and modification of the atom and bond entries during the sequential processing of the CCT. Each atom and bond entry consists of records that contain a separate field for every SMD CT field within that entry, plus pointers to the previous and next records to allow bottom-up and top-down data structure manipulation, respectively. In addition, an atom record contains two extra fields, atom position and locant position, which contain the atom number and its corresponding locant number and are only used in the construction of the CT block. As with the atom table, the TIPE field of the parent structure in the CCT results in an appropriate procedure being executed to handle the relevant parent.

For an aliphatic chain, the SIZE field of the first CCT entry is interpreted as the length of the chain. The numbers of atom and bond entries are equal to the chain length and one less than this, respectively.

With an aromatic ring system, the SIZE field of the first CCT entry is interpreted as the number of rings. The first CCT ring segment entry (RSE) is examined and the content of the SIZE field used to construct the atom and bond entries for a ring of that size. This initial ring is handled differently from any further rings on the parent since it has the same number of atom and bond entries, which are equal to its size.

A procedure to deal with fused rings is executed for each of the additional rings on the parent in turn. The LOCT field of a fused ring's RSE contains the fusion position to the previous ring system. The atom entry that contains this first fusion position is located, and the implied hydrogen field is set to zero for this entry and the following one. The linked-list of atoms is then broken between these two entries, the extra entries are inserted for the new ring and the links restored. The way in which this is done depends upon whether the aromatic ring system is linear or nonlinear.

The bond entries for the new ring are then added to the end of the bond list, after slight modification to the last entry. Once the last fused ring has been handled in this way, the fusion bonds are added to the bond list by searching through the atom list for those entries with zero implied hydrogens. Finally, the atom position and locant position fields in the atom list entries are updated, resulting in all fusion positions having a zero in the latter field.

For alicyclic ring systems, the SIZE field of the first CCT entry again gives the number of rings. The SIZE field of the next CCT RSE is used to construct atom and bond list entries for a ring of that size, the number of implied hydrogens per atom being two. The TIPE field of the third and subsequent CCT entries indicates the nature of additional rings as spiro or bridged. A spiro ring is indicated by a TIPE 3 RSE, whereas bridges are represented by TIPE 2 chain ring segments (CRS). For spiro compounds, the atom and bond list entries are derived from the remaining RSEs as a whole, rather than ring by ring as in aromatic systems. For bridged systems, the number of extra carbon atoms to be added to the atom list is obtained by summing the number of atoms in each of the bridges, as given by the SIZE fields in each CCT CRS. Extra bonds are added to the bond list to describe the connectivity between the atoms within a bridge and with their corresponding bridgeheads.

Steroids are handled differently, and this will be presented in the next paper of this series.[7]

As with the atom table, each parent structure procedure makes a call to another procedure to process any substituents and/or modifications on the particular parent before terminating. Each substituent or modification is handled in turn, and a relevant procedure is called to add to and/or modify the existing atom and bond lists. Complex substituents are dealt with by recursive calls to the appropriate procedure.

The molecular fragments, known as SEFGs, are represented in SMD format by a novel use of superatom entries. We assign a unique superatom symbol to each SEFG comprising a lower case s, e, f, or g plus a single digit. Thus, the 32 SEFGs in Table I are represented by the symbols

s1,s2,—,s9, e0,e1,—,e9, f0,f1,—,f9, g0,g1,g2

Alongside each superatom symbol in the atom list is given a four-character subblock name, which identifies the later explicit description of the superatom structure. The superatom subblock follows the atom list and should normally contain the resolved connection table of the superatom. However, for SEFGs, the subblock is in fact empty; that is, it comprises only a heading record. The reason for this is that SEFGs are represented as text strings on the structure diagram, rather than by their full molecular structure. Consequently, the SEFG subblock name, with the extension string field if necessary, can itself be the relevant fragment string, e.g., COOH, thus eliminating the need for an LB block entry to represent the label of the superatom.

Once the CCT has been fully processed, the atom and bond CT lists are traversed in a top-down manner with the relevant entry details being output to a file in SMD format. This file can then be picked up by any other program requiring SMD input.

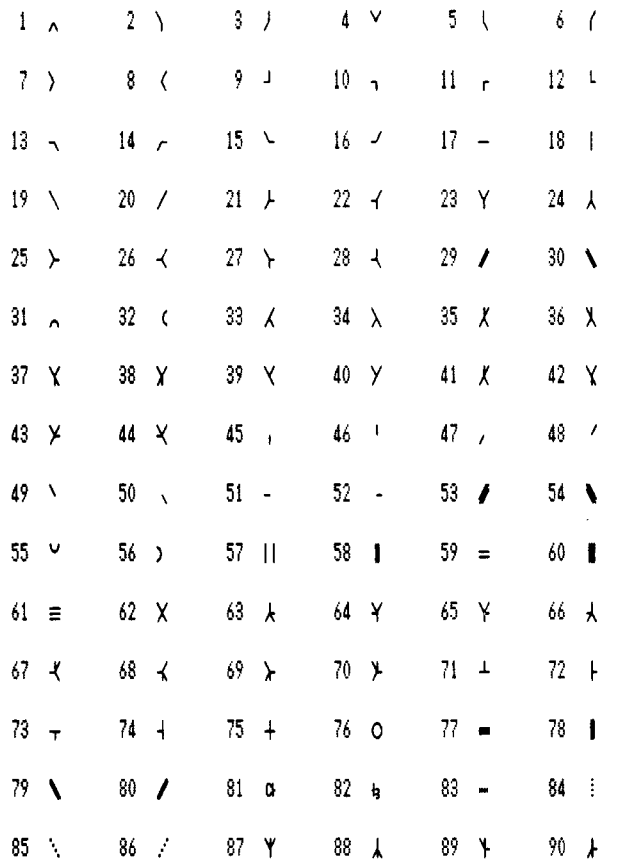

Figure 1. Graphic character set used for the display of structure diagrams.

STRUCTURE DIAGRAM DISPLAY

To confirm that a name corresponds to an expected structure, the display must be clear and unambiguous but need not have the sophistication provided by many current commercially available interactive molecular graphics programs. The two-dimensional unscaled structure diagrams used in organic chemistry textbooks are adequate, and we originally developed our own character graphics software to produce such displays. However, now that the IBM PC has become a chemical industry standard, there are many software packages available for the display, storage, manipulation, and searching of chemical structures.[13] As Warr[14] points out, the future for these various products lies in integration. However, the full integration of software processes in the PC environment remains constrained at present by inadequate systems support for multiple operation of such relatively large packages. These operational problems are discussed in the next section. The development of the CCT-to-SMD conversion software provides a standard data transfer medium that allows the nomenclature translator to provide structural data input to other chemically oriented software packages. This approach has been demonstrated for alternative molecular graphics packages to our own.

Character Graphics. For ring sizes from 4 to 8 carbon atoms and with aliphatic chains drawn as zigzag lines, short straight single or multiple lines of a few different lengths in a limited number of orientations are all that is needed to display the carbon skeleton of a conventional structure diagram with hydrogens assumed. A simple way of providing this on a wide range of microcomputers and printers is by use of a special chemical graphics character set or font. Other atoms and SEFGs can be shown explicitly by atomic symbols using the normal ASCII character set.

A minimal set of some 36 graphic characters initially used in this project for simple rings and chains was published[15] in 1983. This set has been considerably extended into a font of

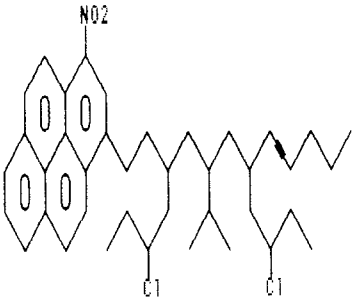

Figure 2. Structure diagram drawn by character graphics.

some 90 characters, shown in Figure 1, that permits a wide range of complex structure diagrams to be displayed. On the IBM PC, video displays are produced by two fundamently different modes, called text and graphics by IBM. Graphics mode is mainly used for complex drawings, but it can be used to display characters as well.[16] In graphics mode the extended ASCII character set (codes 128–255) is located by the PC via the 1FH interrupt vector. By creation of an array of bytes containing the definition of our chemical graphics character set and placement of the start address of this array in the 1FH interrupt vector, the extended set is replaced by the chemical one.

Entries in the atom table are associated by software with corresponding characters from either the graphic or normal character set. The diagram is constructed first in a screen array, an internal representation of the screen. The screen coordinates for each character are computed and stored in the atom table, from knowledge of the nature and position of the preceding character drawn and the context (i.e., chain, ring, atom).

The coordinates of the first character to be drawn are chosen to allow the diagram to expand in all directions. If, however, the structure diagram attempts to pass a boundary of the screen array, then the starting position is recalculated to allow for the greatest expansion in this direction and the diagram reentered into the screen array. A maximum of three reentries is allowed before the user is told that the structure diagram is too large for the screen.

The screen resolution is limited to 80 × 25 characters, and overlapping of characters cannot be allowed. The zigzag aliphatic chain substituents cause most problems with overlap of existing screen characters. Hence, a table of alternative character codes is used by the character graphics software to attempt to redraw the offending substituent in different directions or orientations. Alternatives are tried from the table until the current extent of the structure can be drawn without character overlap and without exceeding the screen size or until there are no more alternatives.

This redrawing is quite successful in enabling the display of structure diagrams that otherwise could not be represented, but there are cases where it is awkward to draw structures without overlap. Figure 2 is the structure diagram finally drawn by the character graphics software after the name 3-[3,7-bis(2-chlorobutyl)-5-isopropyldodec-8-ynyl]-1-nitropyrene has been parsed and processed semantically. Initially, its parent rings and main chain are constructed with the chlorobutyl substituent on locant 3 then drawn down and to the right of its attachment atom. However, this blocks some of the character positions required next for the isopropyl group on locant 5. In this case, the software finds that the chlorobutyl group can be redrawn to the left, as seen in Figure 2, which resolves the conflict.

It is hard to resolve problems of character overlap involving ring systems as substituents, since their rigid internal structure makes reorientation difficult within the partially constructed 2D display. SEFGs with several characters can be redrawn by reversing the order of the characters, e.g., from -COOH to HOOC-.

MOLIDEA. MOLIDEA[17] is a program for the IBM PC that automatically calculates atomic Cartesian coordinates from a molecular description and displays the 3D structure on the color graphics screen. The molecular structure can be displayed either as a wire-frame or a space-filling model, which can subsequently be rotated around the x, y, and z axes or around a selected bond. This package is intended for molecular modeling applications rather than structure diagram display. As such, it complements the other packages we are using.

The main advantage of this system is that coordinates do not need to be provided by the nomenclature translator. The molecular description required by MOLIDEA has been derived from the SMD CT block produced by the CCT-to-SMD conversion. Through collaboration with CompuDrug Ltd. a two-process system has been developed that can repeatedly read in a name and produce a MOLIDEA display of the corresponding structure, the molecular description being passed from the nomenclature translation process to the MOLIDEA process in a file. All the display facilities of the MOLIDEA system are available for manipulation of such displays.

PCMODEL and MOLGRAF. Both of these packages[18,19] allow the drawing and manipulation of 3D structures in graphics form on IBM PCs and compatibles. They are similar to MOLIDEA except that they require either Cartesian or X-ray coordinates. Neither package actually reads an SMD file, but their corresponding connection table inputs are very similar to SMD file format and an interface to perform the necessary conversion has been implemented.

PSIDOM. PSIDOM[20] (Professional Structure Image Database on Microcomputers) is a package for IBM PCs and compatibles providing connection table based input, storage, retrieval, and display of chemical structures. With assistance from Hampden Data Services Ltd., we developed another two-process system that translates names into SMD format and drops this into a file for reading by the PSIDOM package. All the facilities of PSIDOM are available once the name has been translated, including display of the corresponding 2D structure diagram, storage within a PSIDOM database, and modification of the structure followed by restorage in SMD format if required.

SYSTEMS CONSIDERATIONS

The nomenclature translation software forms a large program (14 000 lines of source code) which is structured as several modules for TurboPascal version 5.0 on an IBM PC compatible. Problems exist in linking this program with a variety of graphics, or other, packages to produce one multiprocess software system. These result from the size of the programs concerned, the use of different compilers for each package, limitations imposed by the MSDOS operating system used on IBM PCs and compatibles, and being unable to change the point of entry to packages for which only executable code is available.

It is possible under MSDOS to combine several processes by loading the first, dropping output into a predetermined file, halting and loading, or reloading, the next process, and so on. However, reloading the nomenclature translation software, in particular, is a lengthy process since the fragment dictionary and parse tables have to be copied back into main memory each time. This problem has been overcome with the DESQview multitasking extension to MSDOS.[21]

DESQview is essentially a resource manager for a PC. The nomenclature translation and other required software packages are loaded into memory. If there is insufficient memory to load a new process, DESQview has the capability to switch a previous process out to either expanded memory, a disk, or

RAM disk cache, by memory paging. Once the nomenclature translation software has successfully processed a particular name, a desired display package can be chosen by the user via a key which activates an operating system script, or macro, set up by the nomenclature translation program. This ensures that the requested process is (re)activated. The user can similarly return to the nomenclature translation software when appropriate.

The use of expanded memory or disk cache greatly reduces the transfer time between selected processes. However, each process usually still has to be restarted from the first initialization phase of that process, requiring the user repeatedly to negotiate menu structures, for example, before activating the required package feature. There is typically no means of selecting a particular display and maintaining that choice between activations of the process. Thus, it is not generally possible to construct a composite system based on the integration of separate proprietary packages. This will remain the case until such packages permit multiple entry and exit points for use in conjunction with multitasking operating systems.

DISCUSSION AND CONCLUSION

The concise connection table was originally devised to allow representation of structural fragments within a dictionary of systematic nomenclature morphemes. The CCT has been expanded in line with increasing scope of the Hull nomenclature translator, and the implicit hierarchic structure of the CCT has proved to be easily enhanced to support a variety of unforeseen extensions. The use of the CCT remains an important pivotal technique in the conversion of molecular structural information from systematic nomenclatural representations into alternative connection tables, for storage, retrieval, and graphical display.

A range of techniques and software systems exist for the graphical display of molecular structures, with a variety of data interface mechanisms. The CCT as output from the translator is potentially a further interface, but the need for a commonly agreed standard for communicating structural representations between different software modules is apparent. The Standard Molecular Data (SMD) proposal is welcome as a potential answer, and conversion of structures from CCT to SMD representations now gives a route from nomenclature into an increasing number of other software packages. In particular, a variety of alternative structure display modules may be entered, affording a choice of display styles for graphic output.

Problems have been encountered in the systems linkage between different proprietary software modules in the PC environment. The recent availability of resource manager operating systems for PCs now enables the prototyping of a multiprocess software system using sequences of processes of disparate origin, without any requirement to access protected source code. This is particularly necessary to enable the nomenclature translator to interface with other chemically oriented software packages, but further developments in systems support facilities will be necessary before an acceptable user interaction can be achieved.

ACKNOWLEDGMENT

We are pleased to acknowledge the funding provided by the U.K. Laboratory of the Government Chemist for this project. We are grateful to CompuDrug Ltd., Budapest, Hungary, and Hampden Data Services, Oxford, England, for their assistance in linking their software products with our nomenclature translator.

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.* **1989**, *29*, 101–106.
(2) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 2. Development of a Formal Grammar. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 106–112.
(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.* **1989**, *29*, 112–118.
(4) Rayner, J. D. A Concise Connection Table Based on Systematic Nomenclatural Terms. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 108–111.
(5) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. In *The Awards of Science and Other Essays*; ISI Press: Philadelphia, 1985; p 453.
(6) Goebels, L. The Role of Beilstein Today. Presented at the Conference on Chemical Nomenclature into the Next Millenium—Has It a Role?, London, Nov 12/13, 1987.
(7) Cooke-Fox, D. I.; Kirby, G. H.; Lord, M. R.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 5. Steroids. *J. Chem. Inf. Comput. Sci.* (following paper in this issue).
(8) Ash, J. E. Connection Tables and Their Role in a System. In *Chemical Information Systems*; Ash, J. E., Hyde, E., Eds.; Ellis Horwood: Chichester, U.K., 1974; Chapter 11.
(9) Ash, J. E.; Chubb, P. A.; Ward, S. E.; Welford, S. M.; Willett, P. *Communication, Storage and Retrieval of Chemical Information*; Ellis Horwood: Chichester, U.K., 1985; p 141.
(10) Dittmar, P. G.; Mockus, J.; Couvreur, K. M. An Algorithmic Computer Graphics Program for Generating Chemical Structure Diagrams. *J. Chem. Inf. Comput. Sci.* **1977**, *17*, 186–192.
(11) Barnard, J. M.; Jochum, C. J.; Welford, S. M. ROSDAL: A Universal Structure/Substructure Representation for PC–Host Communication. In *Chemical Structure Information Systems. Interfaces, Communications and Standards*; Warr, W. A., Ed.; ACS Symposium Series 400; American Chemical Society: Washington, DC, 1989; pp 76–81.
(12) Bebak, H., et al. The Standard Molecular Data Format (SMD Format) as an Integration Tool in Computer Chemistry. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 1–5.
(13) Meyer, D. E.; Warr, W. A.; Love, R. A. *Chemical Structure Software for Personal Computers*; ACS Professional Reference Book; American Chemical Society: Washington, DC, 1988.
(14) Warr, W. A. *Graphics for Chemical Structures. Integration with Text and Data*; ACS Symposium Series 341; American Chemical Society: Washington DC, 1987; p ix.
(15) Rayner, J. D.; Milward, S.; Kirby, G. H. A Character Set for Molecular Structure Display. *J. Mol. Graphics* **1983**, *1*, 107–110.
(16) Norton, P. *The Programmers Guide to the IBM PC*; Microsoft Press: Washington, DC, 1985; p 68.
(17) Lopata, S.; Gabanyi, Z.; Bencze, A. MOLIDEA Version 2.0, 1988, available from CompuDrug Ltd., Furst Sandor Utca 5, H-1136 Budapest, Hungary.
(18) Henkel, J. G.; Clarke, F. H. *PCMODEL Molecular Graphics on the IBM PC Microcomputer, Enhanced Version 2.0*; Academic Press: London, 1986.
(19) Barlow, R. B.; O'Donnell, C. H. *MOLGRAF Molecular Graphics for Microcomputers*; Elsevier-BIOSOFT: Cambridge, U.K., 1986.
(20) Hampden Data Services Ltd., PSIDOM Release 5.0, Oxford, U.K., 1987.
(21) Quarterdeck Office Systems, DESQview Version 2, Santa Monica, CA, 1987.

Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 5. Steroid Nomenclature

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

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

Received October 12, 1989

The grammar-based approach to the computer recognition and translation of IUPAC systematic organic chemical nomenclature, described in previous papers in this series, has been applied to semisystematic steroid names. Steroid nomenclature is based on trivially named steroid parents, which may be modified and substituted by following the systematic rules of IUPAC organic nomenclature, and a special numbering scheme. Grammar rules are presented for the stigmastane and lanostane series and for the stereochemical and structural modifications applied to steroids. Semantic processing of acceptable steroid names produces a concise connection table (CCT), and definitions of the extensions to this are given for some of the stereochemical and structural modifications allowed by the grammar. The production from steroid names of Standard Molecular Data (SMD) format and character graphics displays of structure diagrams in the preferred orientation are reported.

INTRODUCTION

Steroids constitute a large and very important class of compounds based on a common tetracyclic ring skeleton. A specialized nomenclature has developed for naming these compounds.[1] Rules developed specifically for naming steroids have been used in other nomenclatures to name diturpenes and triturpenes and for several of the alkaloids.

The principles followed in the naming of steroids are (1) clearly defined stem names with the stereochemistry implied within, (2) systematic application of the rules of organic nomenclature on these stem names, and (3) systematic application of the skeletal modification rules of steroid nomenclature. The use of trivial names for the stem, with systematic application of the rules of organic nomenclature, is particularly common in naming steroids. The skeleton of steroids may be named more systematically as a derivative of cyclopentanoperhydrophenanthrene but would then require the stereochemistry to be explicitly stated for each chiral center. Furthermore, as noted by Garfield,[2] "it is ridiculous to call phenanthrene systematic when it could properly be called benzonaphthalene".

The grammar-based technique for the recognition and translation of chemical nomenclature to structure diagrams, described in previous papers in this series,[3,4] has been applied to steroids. The purpose of this exercise was to demonstrate the application of the technique to a particular class of industrially important compounds where names are based on trivial nomenclature and a specialized numbering.

The systematic numbering of the stigmastane/lanostane steroid nucleus[1] is shown in Figure 1. The numbering is retained even if some of the lower numbered atoms, such as the methyl groups at positions 10 and 13, are missing. The concise connection table (CCT) entries produced in the semantic phase of nomenclature translation have to take into account this steroid specific numbering. The CCT was developed by Rayner[5] to describe structural fragments represented by the parts of a systematic name. It has since been extended to cover additional features including stereochemistry and the steroid nucleus.[6]

The only other study of nomenclature translation applied to steroids, reported by Stillwell,[7] used a small dictionary of steroid nucleus names and common substituent terms. By means of ad hoc processing methods, a tabular representation of molecular geometry was formed for each input name, and diagrammatic output was then produced. Stillwell made no use of formal syntactic methods and concentrated on the stereochemical aspects of steroid structures in his output.

SYNTAX ANALYSIS

To demonstrate the feasibility of applying the nomenclature translator to steroids, it was necessary to recognize a subset of the stem names. These are the stigmastane and lanostane series:

gonane	cholane	lanostane	cucurbitane
estrane	cholestane	tirucallane	protostane
androstane	ergostane	euphane	
pregnane	stigmastane	dammarane	

Only substitutive nomenclature is handled, as this is the system most often applied to steroids.

Derivatives. The above stems may be modified by the inclusion of double and triple bonds in the skeleton. Double bonds that are placed between numerically nonadjacent bridgeheads require both the start and end locants of the bond to be explicitly stated, for example, gon-5(10)-ene. All of the substituents and principal groups that are known to the full grammar, and reported in paper 2,[3] can be applied to steroids. Most of these substituents can be designated either suffixes or prefixes. A few substituents, such as the halogens, can only be named as prefixes, for example, 2-chloro-5α-stigmast-3-ol.

Stereochemical Modifications. All of the principal stereochemical modifications for steroids are handled by the grammar. These include *ent*, *retro*, (±)-*rel*,*R*/*S*, $\alpha/\beta/\xi$ bonds, and *rac*.

This is the first class of compounds for which stereochemistry was introduced in the grammar. The asymmetric center can be represented by the $\alpha/\beta/\xi$ bond or the *R*/*S* system depending on whether it is attached to the tetracyclic ring skeleton or to a side chain, respectively. These notations have now been applied also to all the other classes of compounds currently handled by the full grammar.

Structural Modifications. Ring contractions and expansions are indicated by the prefixes nor and homo. Implementing this feature in the grammar resulted in lettered and ring locants being introduced for the first time, as in C(14a)-homo-5α-androstane. Dinor and dihomo are allowed when loss or insertion of two methylene groups is required.

Fission of a ring, with an additional hydrogen atom being created at each of the two new terminal nodes, is indicated by the prefix seco, for example, 2,3-seco-5β-cholestane. Additional ring formation, signified by cyclo, is allowed. Bond migration, indicated by *abeo*, has not been implemented.

Finally, the occurrence of hetero atoms in the steroid ring system is indicated by the "oxa-aza" replacement system of nomenclature.

0095-2338/90/1630-0128$02.50/0

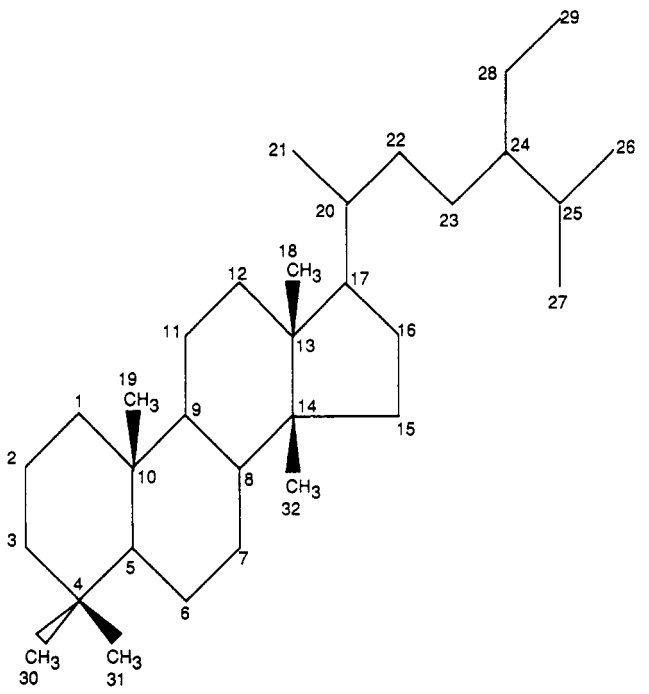

Figure 1. Systematic numbering within the steroid nucleus.

The newly created grammar rules for steroids are given in Appendix, in the format explained in paper 2,[3] together with restatement of earlier rules that needed modification. Steroids have been introduced into the grammar, at the highest level, by the new rule alternative

org-name = steroid-name,

Thus, the nonterminal symbol, "steroid-name", can be expanded, in a top-down fashion, to describe any steroid currently handled by the grammar. In addition, a new rule, defining a nonterminal symbol called "parent-part", has been introduced to allow steroid parents to have prefixed substituents. This modification has led to all those rules for certain classes of compounds, such as acids, alcohols, aldehydes, and ketones, which used to comprise an "aliph-part" and a unique class ending (e.g., ol for alcohol), now comprising a parent-part and the unique ending. This parent-part can either be aliph-part, as before, or the new "steroid-part". This methodology shows the ease with which a whole new class of compounds can be introduced, with only a few minor changes to the existing grammar.

The Greek symbols α, β, and ξ are input, via the keyboard, as alpha, beta, and xi. When this method is used, there needs to be an explicit hyphen between the locant and the modification. This is not the case when a Greek symbol is used, for example, 5α-stigmastane would be input as 5-alpha-stigmastane. Hence, the rules describing the locant and its stereo bond modification are

locant = num-mod number;

num-mod = config-mark hyphen, ..., $;

with the new terminal symbol "config-mark" representing alpha, beta, and xi.

SEMANTIC PROCESSING AND CCT REPRESENTATION

The production of the concise connection table (CCT) in the semantic phase of the name translation has been described previously.[4] In the formation of a CCT representation for a steroid, the output from the semantic phase must include the special numbering scheme and stereochemical and structural modifications. Stereochemistry had not previously been handled by the CCT or by the chemical character graphics technique.[6]

LOCT	TIPE	SIZE	SUBS
1	5	17	0
6	6	6	5
13	1	1	2
10	1	1	2
17	1	2	0
20	1	3	0
24	1	2	0
25	1	1	0
24	1	1	0
28	1	1	0
4	1	0	1
4	1	0	1
14	2	1	1
20	2	1	2
17	2	1	1
5	2	1	0
8	2	1	2
9	2	1	1

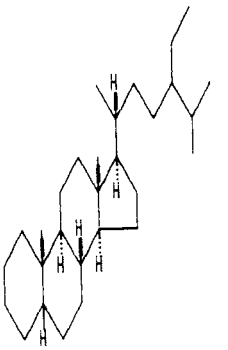

Figure 2. CCT for 5ξ-stigmastane and structure diagram drawn by character graphics.

Steroid CCT Definitions. A new TIPE, 5, was introduced for the steroid CCT main entry. The SIZE field contains a count of the number of CCT steroid segment entries (SSE), which follow the main entry to describe the particular parent. This count includes the steroid nucleus entry, describing the tetracyclic ring skeleton typically by the four ring-cue values 6 6 6 5. The SUBS field of the header entry contains the number of substituents or modifications on the parent, to give the desired overall structure.

The SSEs following the nucleus entry are arranged in the CCT so that, in the drawing phase, the entries in the atom table are placed in locant sequence. In all the segment entries, the LOCT field contains a positive integer that gives the position of attachment of the structure described by the CCT entry to the partially constructed steroid structure.

The TIPE field of an SSE can take one of two values. A TIPE value of 1 indicates that the entry is an aliphatic chain or dummy entry. If the SIZE field is 0 when the TIPE is 1, the entry is a dummy CCT entry and the SUBS field contains a count of the number of dummy entries that need to be added to the atom table to retain the nonsystematic numbering. This is necessary to find the position of attachment of, for example, a 30-chloro group to the lanostane series of parents which do not have the ethyl group (C-28 and C-29) at position 24.

If the SIZE field is not 0, the entry is an aliphatic chain whose length is given by the SIZE field. The SUBS field then has one of the values 0, 1, or 2 to indicate the stereochemistry of the bond connecting the first atom of the chain to the atom being substituted within the steroid nucleus. If the SUBS field is 1, the bond is an α bond and is below the plane of the steroid. If the SUBS field is 2, the bond is a β bond and is above the plane of the steroid, while a SUBS value of 0 indicates that the bond is a ξ bond and may be either above or below the plane.

A TIPE value of 2 indicates that the entry is an atom entry, in which the SIZE field contains the atomic number of the atom substituent and the SUBS field indicates the bond stereochemistry as described above.

The use of the SUBS field for describing bond stereochemistry is specific to the context of the steroid nucleus and contrasts with the use of ordinary bond modification entries in the case of substituents on the nucleus or elsewhere (see later under CCT extensions for steroid modifications, and Table II of ref 6).

The CCT for 5ξ-stigmastane constructed by the semantic processing and the corresponding structure diagram, drawn by character graphics, are shown in Figure 2. Following the

steroid header entry and steroid nucleus entry, the first two segment entries represent methyl groups (aliphatic chains of size 1) at positions 13 and 10 in the β orientation. The next six steroid segment entries, up to locant 28, describe the carbon chain attached at position 17 of the nucleus.

Not all steroid parents have the full complement of carbon atoms in this chain or all the methyl groups attached to the nucleus. Comparison of Figure 1 with 5ξ-stigmastane in Figure 2 shows that the latter parent lacks the two methyl groups at position 4, numbered 30 and 31. Hence, in the 5ξ-stigmastane CCT there are two dummy entries at locant 4, identified by being chains of size 0, with SUBS field value of 1 indicating the number of carbon atoms missing from the steroid numbering system. Although 5ξ-stigmastane lacks a methyl group at position 14, a dummy entry is not required because there is an explicit α hydrogen atom locant there.

The remaining six steroid segment entries in the example CCT are atom entries for hydrogen (SIZE = 1) that need to be explicit to indicate the stereochemistry at the bridgehead of the nucleus (positions 5, 8, 9, and 14) and at the asymmetric centers at positions 17 and 20. The latter two entries result in the structure diagrams for those steroid parents that have asymmetric centers at locants 17 and/or 20 (i.e., all but androstane, estrane, and gonane) displaying explicit hydrogen atoms at these positions. In fact, the presence of these explicit hydrogen atoms is not necessary for the display of those parents in the stigmastane series, because for all these parents the carbon chain attached at position 17 is assumed to be β oriented and the C21 methyl group at position 20 is assumed to be α oriented.

However, with the lanostane series this is not the case, and the above modification can indicate clearly that the carbon chain attached at position 17 is α oriented for tirucallane, euphane, and protostane and, second, that the C21 methyl group at position 20 is β oriented for tirucallane. Without this distinction, the parent structures for tirucallane and euphane would appear the same.

Although the syntactic recognition of the *R*/*S* terminology is handled, semantic processing is extremely difficult and has not yet been implemented. The rules for determining whether a substructure should be labeled *R* or *S* are very complex, involving detailed attention to the structure of the parent molecule in each individual case. Beyond that, there are potential problems in structure display since two-dimensional representations are ambiguous and existing display packages typically do not provide a sufficient variety of bond symbols to make the stereochemistry explicit.

CCT Extensions for Steroid Modifications. As all the CCT extensions for steroids are modification entries, they all have a LOCT field value of zero and a nonzero value in the TIPE field.

Additional ring formation, cyclo, has a TIPE field value of 3 to indicate a bond modification:

LOCT	TIPE	SIZE	SUBS
0	3	3	0
locant 1	config 1	config 2	locant 2

The 3 in the SIZE field represents the cyclo bond, with the start and end locants and their corresponding stereochemical configurations being given in the next CCT entry, as indicated above by locant 1, locant 2, config 1, and config 2.

The ring fission entry, seco, is also a bond modification entry depicted by a SIZE field value of 4 with the start and end locants given in the following CCT entry.

LOCT	TIPE	SIZE	SUBS
0	3	4	0
locant 1	0	0	locant 2

The elimination of methyl groups, the shortening of side chains (nor), and ring contraction and expansion modifications (nor, homo) are all depicted as hetero atom modifications (TIPE = 2):

LOCT	TIPE	SIZE	SUBS	
0	2	0	locant	nor for chains
0	2	0	ring	nor for ring contraction
0	2	6	ring	homo for simple ring expansion
0	2	6	locant	homo for ring expansion in
ring 1	0	0	ring 2	bridgehead

A zero in the SIZE field indicates that the carbon atom will not be replaced by anything. Hence, in the case of elimination of a methyl group and shortening of side chains the SUBS field contains the locant of the carbon atom to be removed. With ring contraction, the SUBS field holds the ring concerned. Rings are represented by 100 for A, 200 for B, 300 for C, and 400 for D, to distinguish them from locants. A 6 in the SIZE field (atomic number of carbon) implies an expansion. The SUBS field gives the ring concerned. However, describing a ring expansion, created by formal insertion of a methylene group between directly linked bridgeheads, requires an additional following CCT entry. This gives the ring(s) affected by the insertion, as shown above, with ring 2 in the SUBS field being zero if only one ring is affected.

Any hetero atoms occurring in the steroid ring system are represented by a hetero atom modification (TIPE = 2) with the SIZE field containing either the atomic number of oxygen or the atomic number of nitrogen, depending on whether an oxa or aza modification is present, respectively.

LOCT	TIPE	SIZE	SUBS
0	2	atomic number	locant

The stereochemistry of substituents, rather than that within the steroid nucleus, is represented by modification entries using the bond codes 0, 1, or 2 (Table II of ref 6):

LOCT	TIPE	SIZE	SUBS	
0	3	0	0	xi bond
0	3	1	0	alpha bond
0	3	2	0	beta bond

Thus, the CCT for 3α-chloro-5β-gonane is represented by

1	5	17	1
	⟨the 17 SSEs⟩		
3	2	17	1
0	3	1	0

The *ent* stereochemical modification is dealt with by replacing all the α bond CCT entries by β bond entries, and vice versa, in the SSEs and any appropriate substituent entries.

CCT Generation. The CCT is constructed from appropriate entries as determined by reference to a dictionary of name fragments or morphemes.[4] To conserve space in this fragment dictionary, not all the structure of each steroid stem is stored, merely the modifications that need to be made to a predefined default steroid. The CCT for this default steroid is created by the semantic rules and then modified by the CCT entries in the fragment dictionary for the stem being processed. The structure chosen to be the default steroid is 5ξ-stigmastane, the only parent that has the complete side chain on carbon 17 (C-20–C-29). The use of ξ at position 5 is introduced as it is the only ring junction that is not implied implicitly by the stem used. Hence, there is no semantic information in the fragment dictionary for stigmastane other than the steroid nucleus entry 6 6 6 5. The CCT for 5ξ-stigmastane contains entries for all those positions on the ring system which may be different in any of the other steroid parents. This method requires the presence of two dummy entries both at locant 4, because all the lanostane series have two methyl groups at that position. Thus, to translate the default parent CCT to any other parent CCT only requires modification of the existing entries, rather than addition of new ones or deletion of old ones. In the fragment dictionary, only the entries that differ from

the default one for a particular parent are stored following the steroid nucleus entry. During the semantic processing, the steroid nucleus entry is expanded to the full CCT for the 5ξ-stigmastane parent structure (Figure 2), modified first by those entries for the particular steroid parent and, second, by any additional substituents or modifications on that parent, such as unsaturations, appearing explicitly in the name. The latter stage always involves the addition of new CCT entries, but may result in the deletion of a steroid segment entry if, for example, a substituent is replacing a parent group or atom.

CCT to SMD Conversion for Steroids. Steroid CCTs are converted to Standard Molecular Data (SMD) format[8] by the following method.

First, a procedure is executed, which constructs the partial atom and bond linked lists for the first 17 locant positions of the 5ξ-stigmastane skeleton (the default structure). At this stage the carbon atom at locant 17 only has two bond entries making up the five-membered ring and no implied hydrogens.

The complete parent structure is then built by investigating in turn each of the CCT steroid segment entries (SSE) following the nucleus segment and using the information to add to or modify the existing data structures. For each SSE the TIPE and SIZE fields are used to determine the entry type, which can be chain, dummy, or atom. The atom and bond lists are then constructed by using a combination of the entry type and locant for the appropriate SSE.

DISPLAY OF STEROID MOLECULAR STRUCTURES

The character graphics technique[9] has proved adequate for simple derivatives of those steroid parents that can be handled (see Figure 2). To allow the stereochemistry at the bridgeheads of a steroid skeleton to be explicitly shown, a number of new chemical characters have been implemented.[6] α bonds are represented by a dashed line, β bonds by a solid line, and ξ bonds by the normal bond character.

Many steroid features that can be handled by the grammar cannot be drawn at present by character graphics. For example, 3α,5-cyclo-5α-cholestan-6β-ol has a bond connecting atoms 3 and 5 in the steroid ring system. The graphic characters required for this bond are not present in the font since it is not possible to allow for all the variations of such characters in a fond of 128.

The software to display steroids by means of the character graphics technique has been written so that the preferred orientation of the steroid nucleus is displayed with the A ring to the lower left and the D ring on the upper right. Use of molecular graphics packages that generate their own coordinates from our SMD CT block[6] does not necessarily ensure the correct orientation. This problem has been recognized by Shelley.[10]

DISCUSSION AND CONCLUSIONS

It has been shown possible to recognize some trivally named steroid parents and to modify the steroid nucleus with a grammar derived from the systematic rules of IUPAC organic nomenclature. While changes to the IUPAC recommendations on the nomenclature of steroids are imminent, these are not expected to result in the need for major changes to the grammar presented here or to our current software.

Difficulties do exist, however. The steroid parents cardanolide and bufanolide behave differently to those currently handled, and the necessary additional rules for these are expected to be rather more complex. Particularly, their behavior with functional group derivatives is problematic, and these names will therefore not be included in the foreseeable future. The same applies to spirostan and furostan, although the ending "-an" is similar to the present "-ane" ending, making their implementation more straightforward.

Problems arise not in the construction of a grammar to recognize the nomenclatural forms that are given in the IUPAC steroid rules but rather in generating a numeric representation of the steroid structure, and the many amended forms, and in checking that the CCT produced is structurally valid. In the semantic tree,[4] the steroid segment entries are held under the parent entry of the semantic tree node in the same way as aromatic ring segment entries. They are therefore not particularly easy to access and check for the validity of the parent. The CCT has been shown to be extendable in handling some of these many forms that are implemented in the grammar. However, some of these extensions do not strictly conform with the original definitions of the LOCT, TIPE, SIZE, and SUBS fields.[5]

The exercise reported here shows that it is reasonable to provide translators for subsets of IUPAC nomenclature for specialist classes of compounds, which may involve semisystematic nomenclature. However, such translators rely on a system for general nomenclature to handle substituents, principal groups, and bond modification. Where, as here, the class is built around one structural skeleton with its own numbering scheme, the semantic processing can use this by providing the necessary modifications to the basic skeleton. The display of stereochemistry has been accomplished with the character graphics technique, but general coverage of ring formation, fission, contraction, and expansion is beyond its capabilities.

ACKNOWLEDGMENT

This work was supported by the U.K. Laboratory of the Government Chemist, and we are particularly grateful to E. W. Godly and I. Cohen of its Chemical Nomenclature Advisory Service for their help and advice.

APPENDIX: CURRENT STEROID GRAMMAR FOR THE IUPAC SYSTEMATIC ORGANIC CHEMICAL NOMENCLATURE

TERMINALS

```
...
steroid-suffix      = e ;
...
steroid-mark        = gon , estr , androst , pregn , chol ,
                      cholest , ergost , stigmast , lanost ,
                      tirucall , euph , dammar ,
                      cucurbit , protost ;
...
config-mark         = alpha , beta , xi ;
seco-mark           = seco ;
nor-mark            = nor ;
ring-locant         = A , B , C , D ;
let-locant          = a , b , c , d , e , f , g , h , i , j , k ,
                      l , m , n , o , p , q , r , s , t , u , v ,
                      w , x , y , z ;
homo-mark           = homo ;
h-mark              = H ;
h-ket               = ' ) ' ;
oxa-aza-mark        = oxa , aza ;
stereo-hyphen       = '-' ;
ent-mark            = ent ;
retro-mark          = retro ;
rac-mark            = rac ;
rs-mark             = R , S ;
rel-mark            = rel ;
sign-mark           = '+' , '-' ;
...
```

RULES

```
name                = org-name ;
```

```
org-name        = aliph-name ,
                  arom-name ,
                  acid-name ,
                  conj-name ,
                  alcohol-name ,
                  ketone-name ,
                  aldehyde-name ,
                  radico-name ,
                  steroid-name ,
                  special-case ;
...
parent-part     = aliph-part ,
                  steroid-part ;
...
acid-name       = acid-mark acid-space acid-parent ;
acid-parent     = oic-part parent-part ,
                  dioic-part parent-part ,
                  carb-suffix carb-root ,
                  unsub-aliphacid ,
                  sub-aliphacid ;
...
carb-root       = aliph-suffix parent-part ;
...
alcohol-name    = ol-part alcohol-parent ;
alcohol-parent  = parent-part ,
                  alc-en-mark arom-parent sub-part ,
                  trivalcohol-part sub-part ;
...
ketone-name     = one-part ketone-parent ;
ketone-parent   = parent-part ;
...
aldehyde-name   = al-part parent-part ,
                  dial-part parent-part ,
                  carbald-suffix carb-root ,
                  glyoxal ,
                  aldehyde-mark triv-alds ;
...
steroid-name    = steroid-suffix steroid-part;
steroid-part    = steroid-parent sub-part
                  stereo-mod ;
steroid-parent  = sat-seq steroid-stem ndprefix-part ;
steroid-stem    = steroic-root loc-part ;
steroid-root    = steroid-mark ;
ndprefix-part   = nondet-prefix ndprefix-part ,
                  $ ;
nondet-prefix   = ring-form ,
                  ring-break ,
                  size-change ,
                  hetero-mod ;
ring-form       = cyclo-mark loc-part ;
ring-break      = seco-mark loc-part ;
size-change     = rg-contractn ,
                  rg-expansion ;
rg-contractn    = nor-mark gm-part hyphen
                  rg-loc-part ;
rg-loc-part     = num-loc-part ,
                  rg-loc-seq ;
num-loc-part    = number num-loc-seq hy-part ;
num-loc-seq     = comma number num-loc-seq ,
                  $ ;
rg-loc-seq      = ring-locant hy-part ,
                  ket let-locant number bra
                  ring-locant secnd-rg-loc hy-part ;
secnd-rg-loc    = ring-locant ,
                  $ ;
rg-expansion    = homo-mark gm-part hyphen
                  rg-loc-part ;
hetero-mod      = oxa-aza-mark loc-part ;
stereo-mod      = stereo-hyphen stereo-desct,
                  $ ;
stereo-desct    = rs-mod ,
                  ent-mod ,
                  retro-mod ,
                  rac-mod ,
                  rel-mod ;
rs-mod          = rs-loc-part ;
rs-loc-part     = ket rs-locant rs-loc-seq bra ;
rs-locant       = rs-mark number ;
rs-loc-seq      = comma rs-locant rs-loc-seq ,
                  $ ;
ent-mod         = ent-mark ;
retro-mod       = retro-mark ;
rac-mod         = rac-mark ;
rel-mod         = rel-mark sign-part ;
sign-part       = hyphen ket sign-mark bra ,
                  $ ;
...
locant          = num-mod number ;
num-mod         = ket number bra ,
                  config-mark hyphen ,
                  h-ket h-mark bra config-mark
                  hyphen ,
                  $ ;
...
ROOTSYMBOL      name
```

REFERENCES AND NOTES

(1) IUPAC/IUB. Definitive Rules for Nomenclature of Steroids (1971). *Pure Appl. Chem.* **1972**, *31*, 283–322.
(2) Garfield, E. An Algorithm for Translating Chemical Names to Molecular Formulas. In "*The Awards of Science and Other Essays*; ISI Press: Philadelphia, 1985; pp 463–464.
(3) Cooke-Fox, D. I.; Kirby, G. H.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Nomenclature. 2. Development of a Formal Grammar. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 106–112.
(4) 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.* **1989**, *29*, 112–118.
(5) Rayner, J. D. A Concise Connection Table Based on Systematic Nomenclatural Terms. *J. Chem. Inf. Comput. Sci.* **1985**, *25*, 108–111.
(6) Cooke-Fox, D. I.; Kirby, G. H.; Lord, M. R.; Rayner, J. D. Computer Translation of IUPAC Systematic Organic Chemical Nomenclature. 4. Concise Connection Tables to Structure Diagrams. *J. Chem. Inf. Comput. Sci.* (preceding paper in this issue).
(7) Stillwell, R. N. Computer Translation of Systematic Chemical Nomenclature to Structural Formulas—Steroids. *J. Chem. Doc.* **1973**, *13*, 107–109.
(8) Bebak, H., et al. The Standard Molecular Data Format (SMD Format) as an Integration Tool in Computer Chemistry. *J. Chem. Inf. Comput. Sci.* **1989**, *29*, 1–5.
(9) Rayner, J. D.; Milward, S.; Kirby, G. H. A Character Set for Molecular Structure Display. *J. Mol. Graphics* **1983**, *1*, 107–110.
(10) Shelley, C. A. Heuristic Approach for Displaying Chemical Structures. *J. Chem. Inf. Comput. Sci.* **1983**, *23*, 61–65.

Journal of
Chemical
Information and
Computer Sciences

Supporting information for *J. Chem. Inf. Comput. Sci.*, **1989**, 29(2), 106 – 112, DOI: 10.1021/ci00062a010

COOKE-FOX
106-112

APPENDIX

The Current Grammar For The IUPAC Systematic Organic Chemical Nomenclature

```
TERMINALS
a-mark          = a ;
e-mark          = e ;

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

val1            = hen ;
val2            = do ;
val3to9         = tri ,
                  tetra ,
                  penta ,
                  hexa ,
                  hepta ,
                  octa ,
                  nona ;

val4to9         = tetra ,
                  penta ,
                  hexa ,
                  hepta ,
                  octa ,
                  nona ;

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

val10to12       = dec ,
                  undec ,
                  dodec ;
val20to22       = icos ,
                  eicos ,
                  henicos ,
                  docos ;

val30           = triacont ;


dec-mark        = dec ;
cos-mark        = cos ;
cont-mark       = cont ;
ct-mark         = ct ;
li-mark         = li ;

gmult-mark1     = a ;
gmult-mark2     = a ;
```

```
an-mark        = an ;
en-mark        = en ;
yn-mark        = yn ;
aliph-suffix   = e ;
arom-suffix    = ene ;
conjring-pf    = ene ;
alc-en-mark    = en ;
rad-mark       = yl ;
conj-radmark   = yl ;
cyclo-mark     = cyclo ;
bicyclo-mark   = bicyclo ;

benz-root      = benz ;
mbenz-root     = tolu ,
                 styr ,
                 cum ;
dbenz-root     = cym ,
                 xyl ;
tbenz-root     = mesityl ;

ompbenz-root   = benz ;
msubbenz-root  = tolu ,
                 styr ,
                 cum ;
dsubbenz-root  = cym ,
                 xyl ;

trivarom-root  = naphthal ,
                 anthrac ,
                 phenanthr ;

arom-root      = triphenyl ,
                 chrys ,
                 naphthac ,
                 pyr ,
                 pic ,
                 peryl ,
                 coron ,
                 trinaphthyl ,
                 pyranthr ,
                 oval ,
                 pental ,
                 ind ,
                 acenaphthyl ,
                 fluor ,
                 fluoranth ,
                 acephenanthryl ,
                 aceanthryl ,
                 rubic ;

multphen-root  = pentaph ,
                 hexaph ,
                 heptaph ;

multphen-rad   = pentaphenyl ,
                 hexaphenyl ,
                 heptaphenyl ;
```

```
arom-mark      = ene ;
linarom-mark   = ac ;
laromrad-mark  = acenyl;
aromrad-mark   = enyl ;
omp-prefix     = o ,
                 ortho ,
                 m ,
                 meta ,
                 p ,
                 para ;
di-mark        = di ;
omp-hyp        = '-' ;


special-case   = ethylene ,
                 allene ,
                 isoprene ,
                 acetylene ,
                 mesitylene ,
                 isobutane ,
                 isopentane ,
                 neopentane ,
                 isohexane ,
                 isobutene ,
                 fluoroform ,
                 chloroform ,
                 bromoform ,
                 iodoform ,
                 phosgene ,
                 thiogene ;

triv-subA      = isopropyl ,
                 isobutyl ,
                 sec-butyl ,
                 tert-butyl ;
triv-subB      = isopentyl ,
                 neopentyl ,
                 tert-pentyl ,
                 isohexyl ,
                 isopropenyl ;

triv-subC      = vin ,
                 all ;

triv-subD      = naphthyl ,
                 anthryl ,
                 phenanthryl ;

triv-subE      = benz ,
                 benzhydr ,
                 cinnam ,
                 pheneth ,
                 styr ,
                 trit ;

phen-rad       = phenyl ;
```

```
mesityl-rad    = mesityl ;
tolyl-rad      = tolyl ,
                 cumenyl ;
xylyl-rad      = xylyl ;


o-sub-root     = fluor ,
                 chlor ,
                 brom ,
                 iod ,
                 dihydroxyiod ,
                 difluoroiod ,
                 dichloroiod ,
                 dibromoiod ,
                 diaz ,
                 azid ,
                 nitros ,
                 nitr ,
                 aci-nitr ,
                 ox ;

o-mark         = o ;
conj-omark     = o ;

pref-subroot   = chlorosyl ,
                 iodosyl ,
                 chloryl ,
                 iodyl ,
                 perchloryl ,
                 hydroxy ,
                 carboxy ,
                 formyl ;

a-sub-root     = az ;

acid-mark      = acid ;
conjacid-mark  = acid ;
oic-mark       = oic ;
conj-oicmark   = oic ;
dioic-mark     = dioic ;
carboxylic-mark = carboxylic ;

trivacid1      = isobutyric ,
                 isovaleric ,
                 pivalic ,
                 lauric ,
                 myristic ,
                 palmitic ,
                 stearic ,
                 pimelic ,
                 suberic ,
                 azelaic ,
                 sebacic ,
                 citraconic ,
                 mesaconic ;

trivacid2      = formic ,
                 acetic ,
```

```
                  propionic ,
                  butyric ,
                  valeric ;

trivacid3      = oxalic ,
                  malonic ,
                  succinic ,
                  glutaric ,
                  adipic ,
                  acrylic ,
                  propiolic ,
                  methacrylic ,
                  crotonic ,
                  isocrotonic ,
                  oleic ,
                  elaidic ,
                  maleic ,
                  fumaric ;


trivacid4      = phthalic ,
                  isophthalic ,
                  terephthalic ,
                  toluic ;


benz-acid      = benzoic ;

naph-acid      = naphthoic ;


conj-trivacid = formic ,
                  acetic ,
                  propionic ,
                  butyric ,
                  valeric ;

ol-mark        = ol ;
conjol-mark    = ol ;

triv-alcohol   = phen ,
                  naphth ,
                  anthr ,
                  phenanthr ;

triv-alcoxy    = methoxy ,
                  ethoxy ,
                  propoxy ,
                  butoxy ,
                  phenoxy ;

unsub-alcoxy   = isopropoxy ,
                  isobutoxy ,
                  sec-butoxy ,
                  tert-butoxy ;

oxy-mark       = oxy ;
```

```
monoradico-mark   = alcohol ,
                    thiol ,
                    selenol ,
                    hydroperoxide ,
                    cyanide ,
                    isocyanide ,
                    azide ,
                    fluoride ,
                    chloride ,
                    bromide ,
                    iodide ;

diradico-mark = ketone ,
                thione ,
                selone ,
                ether ,
                sulphide ,
                sulphoxide ,
                sulphone ,
                selinide ,
                selenoxide ,
                selenone ;

one-mark      = one ;
oxo-mark      = oxo ;

al-mark       = al ;
dial-mark     = dial ;
carbaldehyde-mark = carbaldehyde ;
aldehyde-mark = aldehyde ;

trivald1      = isobutyr ,
                isovaler ,
                pival ,
                laur ,
                myrist ,
                palmit ,
                stear ,
                pimel ,
                suber ,
                azela ,
                sebac ,
                citracon ,
                mesacon ;

trivald2      = form ,
                acet ,
                propion ,
                butyr ,
                valer ;

glyoxal       = glyoxal ;

trivald3      = malon ,
                succin ,
                glutar ,
                adip ,
```

```
                  acryl ,
                  propiol ,
                  methacryl ,
                  croton ,
                  isocroton ,
                  ole ,
                  elaid ,
                  male ,
                  fumar ;

trivald4      = phthal ,
                isophthal ,
                terephthal ,
                tolu ;

benz-ald      = benz ;

naph-ald      = naphth ;

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

e-mult        = bis ,
                tris ,
                tetrakis ;
emult-mark    = akis ;

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

gk-loc        = alpha ,
                beta ,
                gamma ,
                delta ,
                epsilon ,
                zeta ,
                eta ,
                theta ,
                iota ,
                kappa ,
                lambda ,
                mu ,
                nu ,
```

```
                  xi,
                  omicron ,
                  pi ,
                  rho ,
                  sigma ,
                  tau ,
                  upsilon ,
                  phi ,
                  chi ,
                  psi ,
                  omega ;

config-mark     = alpha ,
                  beta ,
                  xi ;


space           = ' ' ;
comma           = ',' ;
hyphen          = '-' ;
arom-hyphen     = '-' ;
bra             = '(' ;
ket             = ')' ;
substbenz-ket   = ')' ;
encarom-ket     = ')' ;
sqbra           = '[' ;
sqket           = ']' ;
dot             = '.' ;


RULES
name            = org-name ;

org-name        = aliph-name ,
                  arom-name ,
                  acid-name ,
                  conj-name ,
                  alcohol-name ,
                  ketone-name ,
                  aldehyde-name ,
                  radico-name ,
                  special-case ;

aliph-name      = aliph-suffix  aliph-part ;

aliph-part      = aliph-parent  sub-part ;

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

chain-parent    = sat-seq  aliph-stem ;

ring-parent     = monocy-parent ,
                  bicy-parent ;

monocy-parent = sat-seq aliph-stem cyclo-mark ;
```

```
bicy-parent   = sat-seq  aliph-stem  sqket  bridge3  dot
                   bridge2  dot  bridge1  sqbra
                   bicyclo-mark ;

bridge1       = number ;

bridge2       = number ;

bridge3       = number ;

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

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

en-ext        = a-mark ,
                $ ;

an-sat        = an-mark ;

arom-name     = arom-suffix  arom-part ;

arom-part     = arom-parent  sub-part ,
                subst-trivarom  sub-part ,
                unsubst-trivarom ;

arom-parent   = benz-root ,
                trivarom-root ,
                arom-root ,
                multphen-root ,
                linarom-mark mult-value ;

subst-trivarom = mbenz-root ,
                 dbenz-root  arom-hyphen  locant  comma
                    locant  hy-part,
                 tbenz-root ;

unsubst-trivarom  = benz-stem disub-part arom-prefix ,
                    msubbenz-stem omp-sub2 arom-prefix ,
                    dsubbenz-root arom-prefix ;

benz-stem     = ompbenz-root;

msubbenz-stem = msubbenz-root;

disub-part    = omp-sub di-mark ,
```

```
                omp-sub2 omp-sub1 ;

omp-sub1      = omp-sub ;

omp-sub2      = omp-sub ;

omp-sub       = subs ;

arom-prefix   = omp-hyp  omp-prefix ;

acid-name     = acid-mark acid-space acid-parent ;

acid-parent   = oic-part  aliph-part ,
                dioic-part  aliph-part ,
                carb-suffix  carb-root ,
                unsub-aliphacid ,
                sub-aliphacid ,
                unsub-aromacid ,
                sub-aromacid ;

oic-part      = oic-mark ;

dioic-part    = dioic-mark  aliph-suffix ;

carb-suffix   = carboxylic-mark  gm-part  loc-part ;

carb-root     = aliph-suffix  aliph-part ,
                arom-suffix  carbarom-part ;

carbarom-part = carbarom-root  sub-part ;

carbarom-root = arom-parent ;

unsub-aliphacid = trivacid1 ;

sub-aliphacid  = sub-aliphacid-part  sub-part ;

sub-aliphacid-part = trivacid2 ,
                     trivacid3 ;

unsub-aromacid = trivacid4 ;

sub-aromacid  = sub-aromacid-part  sub-part ;

sub-aromacid-part = benz-acid ,
                    naph-acid  loc-part ;

acid-space    = acid-space  space ,
                space ;

conj-name     = conj-parent  conj-subs ;

conj-parent   = chain-part  gm-part  ring-part loc-part ;

chain-part    = conjacid-mark  acid-space  conj-acid ,
                conj-alcohol ;

conj-acid     = conj-trivacid ,
```

```
                conj-oicpart  an-mark  conj-aliphstem;

conj-oicpart  = conj-oicmark;

conj-aliphstem = aliph-stem;

ring-part     = conjring-pf  conj-root;

conj-root     = arom-parent ;

conj-subs     = conjsub-part  conj-subs ,
                $ ;

conjsub-part  = substit ,
                conj-substit gm-part gkloc-part ;

conj-substit  = conj-radmark  rad-sub ,
                conj-omark o-sub-root ;

conj-alcohol  = conjol-part an-mark  conj-aliphstem ;

conjol-part   = conjol-mark ;

alcohol-name  = ol-part alcohol-parent ;

alcohol-parent = aliph-part ,
                 alc-en-mark  arom-parent  sub-part ,
                 trivalcohol-part sub-part ;

trivalcohol-part = triv-alcohol ;

ol-part       = ol-mark  gm-part  loc-part  e-part ;

ketone-name   = one-part ketone-parent ;

ketone-parent = aliph-part ;

one-part      = one-mark  gm-part  loc-part  e-part ;

aldehyde-name = al-part  aliph-part ,
                dial-part  aliph-part ,
                carbald-suffix  carb-root ,
                glyoxal ,
                aldehyde-mark  triv-alds ;

triv-alds     = unsub-aliphaldehyde ,
                sub-aliphaldehyde ,
                unsub-aromaldehyde ,
                sub-aromaldehyde ;

al-part       = al-mark ;

dial-part     = dial-mark  aliph-suffix ;

carbald-suffix = carbaldehyde-mark  gm-part  loc-part ;

unsub-aliphaldehyde = trivald1 ;
```

```
sub-aliphaldehyde = sub-aliphald-part  sub-part ;

sub-aliphald-part = trivald-part ;

trivald-part  = trivald2 ,
                trivald3 ;

unsub-aromaldehyde = trivald4 ;

sub-aromaldehyde  = sub-aromald-part  sub-part ;

sub-aromald-part = benz-ald ,
                   naph-ald  loc-part ;

radico-name    = monoradico-suffix space  radico-part ,
                  radsuffixandparent secondradical ;

radsuffixandparent = diradico-suffix  space radico-part ;

radico-part    = radico-parent  sub-part ;

radico-parent = subst-sub ;

secondradical = space radico-sub sub-part ,
                di-mark ;

radico-sub     = subst-sub ;

monoradico-suffix = monoradico-mark ;

diradico-suffix = diradico-mark ;

e-part         = $ ,
                 e-mark ;

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

em-part        = e-mult ,
                 emult-mark mult-value ,
                 $ ;

mult-value     = gmult-cont ,
                 mult10to99 ;

mult10to99     = val10to12 ,
                 dec-mark  val3to9 ,
                 val20to22 ,
                 cos-mark  val3to9 ,
                 val30  mult1to9 ,
                 mult40to90   mult1to9  ;

mult40to90     = cont-mark  val4to9 ;

mult1to9       = $ ,
                 val1 ,
                 val2 ,
```

```
                val3to9 ;

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

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

locant          = num-mod  number ;

num-mod         = ket  number  bra ,
                  config-mark hyphen ,
                  $ ;

hy-part         = hyphen ,
                  $ ;

gkloc-part      = hyphen  gk-locant  gkloc-seq  hy-part ,
                  $ ;

gkloc-seq       = comma  gk-locant  gkloc-seq ,
                  $ ;

gk-locant       = gk-loc ;

sub-part        = substit  sub-part ,
                  $ ;

substit         = complex-sub,
                  simple-sub ;

complex-sub     = ket encs-part bra em-part loc-part ;

encs-part       = subpa-head  sub-part ,
                  sub-alcoxy  sub-part ;

subpa-head      = radical  substit ;

radical         = subst-sub ,
                  num-arom ,
                  nonum-arom ;

sub-alcoxy      = alcoxy-sub  substit ;

alcoxy-sub      = alcoxy-part ;

simple-sub      = subs  gm-part  loc-part ;

subs            = pref-sub ,
                  subst-sub ,
                  unsubst-sub ,
                  nonum-arom ,
                  alcoxy-part ;

pref-sub        = o-mark  o-sub-root ,
                  pref-subroot ,
                  a-mark a-sub-root ;
```

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

rad-sub        = sat-seq aliph-stem ,
                 aliph-stem ,
                 sat-seq aliph-stem cyclo-mark ,
                 aliph-stem cyclo-mark ,
                 triv-subC ;

unsubst-sub    = triv-subA ,
                 triv-subB ,
                 triv-subE ,
                 subst-benz ,
                 enc-arom ,
                 unsub-alcoxy ;

subst-benz     = mesityl-rad ,
                 tolyl-rad  arom-prefix ,
                 substbenz-ket  enc-substbenz  bra ;

enc-substbenz  = tolyl-rad  arom-hyphen  locant ,
                 xylyl-rad  arom-hyphen  locant  comma
                    locant ;

enc-arom       = encarom-ket  arom-rad  arom-hyphen
                    locant  bra ;

num-arom       = arom-rad  arom-hyphen locant hyphen ;

nonum-arom     = arom-rad  ;

arom-rad       = arom-rad1 ,
                 arom-rad2 ;

arom-rad1      = multphen-rad ;

arom-rad2      = triv-subD ,
                 aromrad-mark  arom-root ,
                 laromrad-mark mult-value ;

alcoxy-part    = oxy-part  alc-part ,
                 triv-alcoxy ;

oxy-part       = oxy-mark ;

alc-part       = subst-sub ,
                 num-arom ,
                 nonum-arom ;
```

ROOTSYMBOL name