International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 5 3067 - 3069

Translation of Organic Compound to 2D Graphical Representation using SDT

Ms. Snehal H. Chaflekar
Department of Information Technology
Priyadarshini Bhagwati College of Engineering
Nagpur, Maharashtra, India
Assistant Professor
snehalchaflekar@gmail.com

Ms. Mukta D. Khirwadkar
Department of Computer Technology
Yeshwantrao Chavan College of Engineering
Nagpur, Maharashtra, India
khirwadkar.mukta@gmail.com

***Abstract*—** IUPAC (The International Union of Pure and Applied Chemistry) customary is employed to explain structure and characteristic of chemical compound. This paper describes translation of IUPAC (International Union of Pure and Applied Chemistry) name into Two-dimensional structure of substance that consists of graphical entities. OpenGL graphical language is wont to generate graphical structure of IUPAC name. Chemical names square measure a typical manner of act chemical structure data. Basic graphical entities square measure wont to generate 2nd graphical structure of IUPAC name from Intermediate Graphical Language. computer file is generated on analyzing IUPAC name. computer file is OpenGL graphical functions to show graphical entities. This translation is achieved victimisation Syntax – Directed Translation theme by associating linguistics action. This offers internet 2nd graphical illustration of IUPAC name. This paper proposes a strategy for achieving this translation

***Index Terms*—** *Graphical Language, IUPAC, Syntax – Directed Translation, OpenGL, Organic Compound, Two-Dimension Structure*

*****

I. INTRODUCTION

The International Union of Pure and Applied Chemistry (IUPAC) standardized nomenclature for organic compound which has been widely accepted among the chemists. This nomenclature provides the organic chemists with basis for identification of numerous compounds. A strict nomenclature system is obviously required to eliminate ambiguities in characterizing a chemical formula. The IUPAC system satisfies this requirement. Syntax-directed translation can be advantageously used to transform formulas in IUPAC nomenclature into their standard two-dimensional representation. The purpose of this project is to describe how this translation can be accomplished and generate 2D graphical representation from this translation. Basic graphical entities are used to generate 2D graphical structure of IUPAC name from Intermediate Graphical Language. Electronic representations of chemical structures are one of the informatics underpinnings for any organization operating in the domain of chemistry or biology and enable the creation of a structure/substructure searchable database of chemical structures and associated data and knowledge. There is an enormous wealth in the form of chemical names and a means by which to convert those alphanumeric text descriptors into a more rich chemical structure representation has long been the mission of a large group of investigators. With increasing understanding of chemistry and the graphical representation of chemical structures came the need for an agreed upon language of communication between scientists, despite the limitations and challenges associated with chemical names graphical chemical structure representations. By providing software package tools for the conversion of differing chemical nomenclatures into universally recognized chemical structures, chemists will a lot of simply review the chemical structure of interest and also the information will be migrated to info technologies. The conversion of chemical names into chemical structures will be diagrammatic as 2 crossed schemes – that of utilizing a look-up wordbook and of victimization syntax analysis. a mixture of those 2 approaches is unquestionably required for the analysis of chemical names old within the planet. For the conversion of systematic names a more powerful and flexible approach must be based on the parsing of the chemical names and the application of syntax analysis.

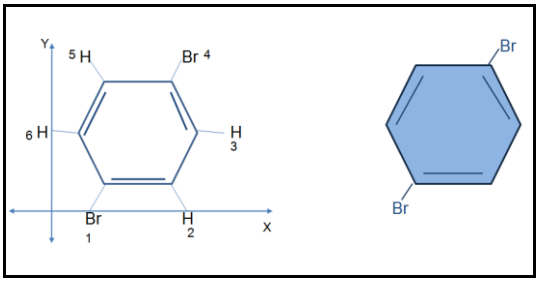

Fig. 1 *Two-Dimensional Structure of 1, 4-DIBROMO_CYCLOHEXANE*

In this project, 2D graphical structure of IUPAC name is generated. Grammar for detection of IUPAC name is generated. Intermediate Graphical Language is used that describe basic graphical entities. So, input is Intermediate Graphical Language and output is 2D graphical structure. In above figure, structure of 1, 4-DIBROMO_CYCLOHEXANE contain hexane ring with element bromine at position 1 and 4 on hexane at Cartesian co-ordinate.

 ISSN: 2321-8169
3067 - 3069

II. SEMANTIC TRANSLATION

According to IUPAC naming rules, one can define compound names such as:

(1) 3-CHLORO-3-AMINO-2-PENT-4-ENONE
(2) 2,4-DIBROMO-CYCLO HEXANE
(3) 3-HEPTYNE

The above strings (implicitly) contain the following information necessary for the translation:

(a) the length of the carbon chain (e.g. PENT indicating a carbon chain of length 5 in (1), HEX meaning a chain length of 6 in (2));
(b) one or more unit prefixes, indicating the number of the carbon to which any functional group is attached, followed by the functional group itself (e.g. a chlorine atom and amino group are attached to carbon 3 in (1));
(c) the carbon number and type of unsaturation, which composes the primary suffix (e.g. a double bond at carbon 4 in (1); a triple bond at carbon 3 in (3));
(d) (d) a secondary suffix once more providing points of attachment of purposeful teams (e.g. carbon a pair of could be a carbonyl in (1)); singular or multiple functionality (e.g. 2 bromine atoms are present in (2)).

The translation process involves the execution of semantic actions once the elements of a rule are recognized while parsing. Text file is OpenGL graphical functions to display graphical entities. Lexical analyzer is used to analyze text file to generate tokens and syntax analyzer is used to parse graphical function according to grammar rule. This gives net 2D graphical representation of IUPAC name. OpenGL functions for generating various graphical entities have been identified. For example, consider the following IUPAC Name 2-bromo-1-chloro-4-methyl pentane,

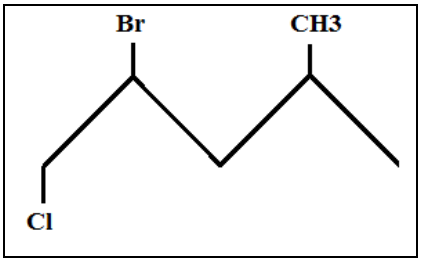

Fig.1. Structure of 2-bromo-1-chloro-4-methyl pentane

To generate above structure we require OpenGL function for line and text. Input is intermediate graphical text file which contain OpenGL functions for structure. On parsing document victimization Jflex and CUP tool it offers variety of carbon needed for structure, that variety of vertex is generated in OpenGL.

TABLE I. Carbon number with their functionality

Carbon No.	Function on Carbon
1	Cl
2	Br
4	CH3

Transition functions are implemented to represent line with associated angle and length according to structure specified by IUPAC name. First vertex can be generated with the reference of another with associated angle and length. Also transition functions to generate text on vertex are used. This gives net two-dimensional structure of organic compound.

III. PROPOSED METHODOLOGY

A. Syntax Directed Translation

A Syntax-directed translation is a context-free grammar in which attribute associated with the grammar symbol. Semantic action is associated with each attribute to transform into basic graphical entities and which can be further transform into two-dimensional representation. When syntax-directed translation is carried out every attribute is translated into basic graphical entities with associated co-ordinate where it should display. This net result is displayed on screen gives 2D graphical representation. A Syntax-directed translation is a context-free grammar in which attribute associated with the grammar symbol. Semantic action is associated with each attribute to transform into basic graphical entities and which can be further transform into two-dimensional representation. Lexical analyzer is used to analyze text file to generate tokens and syntax analyzer is used to parse graphical function according to grammar rule. This gives net 2D graphical representation of IUPAC name. JFlex is a lexical analyzer generator (also known as scanner generator) for Java, written in Java. It is also a rewrite of the very useful tool Jlex. It can be used together with ANTLR. JFlex has built-in support for the CUP parser generator. JFlex is a lexical analyser generator for Java from lexical specification. CUP is a system for generating LALR parsers from simple specifications. CUP is used to generate syntax analyzer. Using CUP involves making a straightforward specification supported the synchronic linguistics that a programme is required, at the side of construction of a scanner capable of breaking characters up into meaningful tokens (such as keywords, numbers, and special symbols).

B. OPENGL Graphical Language

OpenGL is software interface to graphics hardware. OpenGL may be a cross-platform, language-independent, industrial normal API for manufacturing 3D (and 2D) camera work.

OpenGL is intended as a efficient, hardware-independent interface to be enforced on many alternative hardware platforms.

With OpenGL, you want to build up your required model from atiny low set of geometric primitives - points, lines, and polygons. Graphics cards that claim OpenGL-compliance create use of the hardware acceleration once attainable to hurry up the graphics rendering method. Rendering, is that the method by that a pc creates pictures from models. These models, or objects, ar made up of geometric primitives - points, lines, and polygons - that ar like by their vertices. JOGL, or Java Bindings for OpenGL, permits Java programs to access the OpenGL API for graphics programming. The graphics code in JOGL programs can look nearly similar to that found in C or C++ OpenGL programs, because the API is mechanically generated from C header files. JOGL uses Java Native Interface (JNI) to access OpenGL JOGL uses Java Native Interface (JNI) to access OpenGL. This is one among the best strengths of JOGL, because it is sort of straightforward to port OpenGL programs written in C or C++ to JOGL; learning JOGL is actually learning OpenGL. it\'s merely a wrapper library for Java application to use OpenGL API. JOGL is ASCII text file and presently maintained by \"JogAmp\" (Java on Graphics, Audio, Media and Processing). The setting given by the OpenGL bindings permits for quicker graphics programming and development by increasing interactivity. once graphics commands square measure issued to the J system they\'re directly understood and dead. The technologist is thus allowed to experiment with translation values or red-green-blue-alpha values (or the other graphics parameter) and understand the results of his or her changes comparatively quickly. In alternative words, the graphics state updates because the technologist problems commands within the J atmosphere. As a result, the technologist directly sees the result of his or her actions while not having to compile, link, and check the applying changes pro re nata once operating within the C or C++ development environments. Even with the convenience of build scripts that automatise the build method, building the C or C++ viable to guage the results of added OpenGL commands is time overwhelming and still needs the technologist to run the freshly engineered viable.

As for the choice of the OpenGL graphics programming libraries, OpenGL offers a strong set of utilities and commands appropriate to a large vary of 2 and 3 dimensional graphical programming endeavors. OpenGL has conjointly evolved as a typical amongst 3D graphics programming languages and has been ported to and in depth type of platforms and operative systems. In any case, the creation of J bindings for OpenGL definitely doesn't exclude or forestall the creation of J bindings for alternative graphics programming libraries within the future. OpenGL may be a specification enforced within the C language, although it will use alternative programming languages. it's designed on the idea of a state machine, although newer OpenGL versions have remodeled it into way more of associate object-based system. As an API, OpenGL depends on no specific language feature, and might be created owed from nearly any programing language with the right bindings.

In general, Direct3D is meant to virtualize 3D hardware interfaces. Direct3D frees the sport computer programmer from accommodating the graphics hardware. OpenGL, on the opposite hand, is meant to be a 3D hardware-accelerated rendering system which will be emulated in computer code. These 2 Apis ar essentially designed underneath 2 separate modes of thought.

As such, there ar useful variations in however the 2 Apis work. Direct3D expects the appliance to manage hardware resources; the developer should manage hardware resources severally. However, the implementation is less complicated, and developers have the flexibleness to allot resources within the most effective approach doable for his or her application.

In laptop computer code, Direct3D may be a proprietary API designed by Microsoft Corporation that gives a consistent API for hardware 3D acceleration on the Windows platform. Direct3D is enforced, like OpenGL on a similar platform, within the show driver. OpenGL is AN open customary Application Programming Interface that gives variety of functions for the rendering of 2nd and 3D graphics. AN implementation is on the market on latest operational systems.

References

[1] Antony J. Williams, Andrey Yerin, “Automated Identification and Conversion of Chemical Names to Structure Searchable Information”, Advanced Chemistry Development Inc., Moscow Department, 2007

[2] Christopher Southan and Andras Stracz , “Extracting and connecting chemical structures from text sources using hemicalize”, Southan and Stracz Journal of Cheminformatics, 2013

[3] “Converting Chemical Names to Structures with Name=Struct”, CambridgeSoft

[4] Daniel Mark Lowe , “Extraction of chemical structures and Ricardo Veguilla , “Introduction to OpenGL and JOGL”, @ http://www.cl.cam.ac.uk/research/dtg/

[5] Gene Davis, “Learning Java Bindings for OpenGL (JOGL)”, Woods Cross City, Utah 84087, 2004

[6] Andrew W. Appel, “Modern Compiler Implementation in Java”

[7] Dave Shreiner, Mason Woo, Jackie Neider, “The Red Book OpenGL Programming Guide “, Addison-Wesley, sixth edition

[8] Mark Segal, Kurt Akeley, “The OpenGLR Graphics System” , Version 2.0 ,October 22, 2004