Chemlogic:

A Logic Programming Computer Chemistry System

Second Edition

Nicholas Paun

* Abstract

Chemlogic is a logic program for computer chemistry that performs stoichiometric calculations, balances and completes equations, and converts between formulas and names. The program has applications in education, particularly as a study tool. Features are implemented using a chemical information database, linear equation solver, and grammatical rules. Guidance is provided for resolving errors in user input. Chemlogic is available on Android and the Web.

Background

In high school Chemistry, students learn to write formulas and names for chemical compounds and to write and balance chemical equations. These concepts are simple, but implementing a program to do this was an interesting task. Algorithms were researched, adapted to chemistry problems and implemented in Prolog to create a program that could be useful in education.

Design and Implementation

User input is parsed into a form that can be easily manipulated and transformed. A parser recognizes a formal grammar describing valid user input. In Prolog, parsers are imple-

mented using DCGs (Definite Clause Grammars), which provide a simplified syntax for creating logical clauses that process a grammar using difference lists, an efficient representation.[4]

Internal representations must be created for the parsed input. Chemlogic uses a pseudo-Abstract Syntax Tree to record the structure of an equation, as well as lists containing useful information (e.g. the elements contained in an equation).

The reaction type analysis module identifies common types of chemical reactions, predicts whether a reaction will take place, and completes chemical equations for common reactions, given the reactants. These features are implemented by matching the Abstract Syntax Tree generated by parsing an equation, against a pattern representing the structure of a certain chemical reaction type. In Prolog, this matching is performed by syntactic unification, which is the process of combining two structures by replacing variables with constant terms, so as to make the two structures equal.

Once the reaction type is determined, a prediction can be made as to whether the reaction will occur by comparing the relative positions in the reactivity series of the elements involved. The reaction type and the reaction prediction can then be displayed by the user interfaces, as additional information for the entered chemical equation.

To complete a chemical equation, the complete formulas of the product compounds must be calculated, given the arrangements of elements and ions determined by reaction pattern. This process is performed using rules specific to the type of compound being formed by the reaction. For example, in the case of an ionic compound, the charge of the cation becomes the subscript of the anion, and vice-versa, then the formula is reduced to lowest terms. The complete formulas are then substituted into the equation structure, to produce a complete chemical equation. This structure can then be used as input by other modules, such as the chemical equation balancer.

Balancing of chemical equations is usually performed by inspection.[3] This process cannot easily be used in a program because it is unsystematic. Instead, Chemlogic uses an efficient algorithm, representing a chemical equation as a system of linear equations. One linear equation is created for every element in a chemical equation, with the number of occurrences of the element in each formula representing a coefficient, multiplied by an unknown (the chemical equation coefficient).[5]

These systems are commonly solved by converting them to a matrix and applying Gaussian elimination.[2] In Chemlogic, a matrix is produced from structures created by the parser and lookup tables. The matrix is converted into a system of linear equations, which is then provided to the built-in CLP(q) facility, which can solve constraints over rational numbers.[1]

Stoichiometry is an important new feature of Chemlogic. The program can be used to perform a variety of complex calculations involving quantities of chemicals in a reaction. The process begins by parsing a chemical equation that includes quantities for some compounds. Whenever a quantity is parsed, the number of significant figures in the value is determined and recorded in the quantity structure. A list of queries for quantities to be determined is also parsed.

The Abstract Syntax Tree for the given chemical equation is then traversed to determine the number of known quantities. If more than one quantity is known, the limiting reactant must be determined and used to calculate the results. This is performed by converting each known reactant quantity to moles, so that the values can be compared, then dividing by the corresponding coefficient of the reactant in the equation. The limiting reactant always has the lowest value for this ratio. If only one quantity is known, it must, logically, be used to perform all the calculations.

Once the correct quantity and compound to use as input for the calculation is determined, the list of queries is traversed, and the input quantity is converted to each requested output

quantity. This conversion is performed according to the same method that is typically used when performing stoichiometric calculations manually: the input quantity is converted to moles, then moles of the input compound are converted to moles of the requested compound and, finally, moles of the requested compound are converted to the output unit specified by the user. Conversions between units are performed by simple rules, which take into account the maximum number of significant figures that can be yielded by the calculation. When the results of the stoichiometric calculation is displayed to the user, all values are rounded to the correct number of significant figures.

Syntax errors cannot simply cause a program to fail — clear identification and explanation of an error is necessary. When a predicate that must succeed for a given input to be valid fails, a syntax error exception is thrown, containing a code name for the error and the remaining unparsed input (within the tail). The exception aborts the execution of the program and is then passed to the error handling module. It first attempts to localize the error by highlighting only the problematic part within the tail. Different rules are used depending on the type of the first character (e.g. an invalid letter suggests a chemistry mistake, while an invalid symbol suggests a typo). The combination of an error code and character type is used to find the correct error message to provide to the user.

The Android App was developed to make Chemlogic more readily available as a study tool. The App consists of a user-interface, implemented in Java, using the Android APIs, and a package consisting of the cross-compiled code of Chemlogic and its dependencies. The user interface communicates with Chemlogic through a UNIX pipe. Upon receiving the solution from Chemlogic, the interface renders the formatting of the result and displays it.

The Chemlogic package contains a copy of the Chemlogic command-line interface, compiled as a stand-alone application with an embedded Prolog interpreter in its binary, some

necessary Linux libraries and an initialization script. When loaded, the App executes the initialization script, which starts Chemlogic by running the dynamic linker to locate and link Chemlogic with the provided libraries.

Discussion

Performance was analyzed in Chemlogic by counting inferences (provided by `time/1`) used by different algorithms for various problem sizes. Algorithms were compared on their fixed inferences (intercept), inferences per item (slope) and to ensure that their complexities were not exponential.

The time taken by the algorithms used in Chemlogic could not be analyzed because the difference between the performance of algorithms on typical problem sizes was immeasurably small.

Further research and development — The parsers currently implemented in Chemlogic process user input directly, as character lists, without a tokenization process. Using new features available in version 7 of SWI-Prolog, a simple and fast procedure could be implemented to separate user input into space-delimited parts, [8] significantly reducing the number of tests and operations required to perform parsing.

Developing a feature to automatically generate and mark random chemistry problems for student review and test creation, would be an important extension to Chemlogic. With the addition of reaction type analysis, the preliminary development for this feature has been completed. The reaction type, equation, formula and name grammars in Chemlogic could be extended to produce random valid structures, in addition to simply recognizing and converting them.

Conclusions

Prolog was chosen as the language for Chemlogic because its features make it well suited to writing programs of this type in a simple and efficient way. The built-in Definite Clause Grammars syntax allows a programmer to implement advanced parsers using a very simple syntax, and strong support for metaprogramming allows syntax and code to be simplified in powerful ways. CLP(q), a linear equation solving library, is also included in SWI-Prolog. Using a logic programming language, such as Prolog, enables the programmer to describe the results, instead of the process.[7]

Each module of Chemlogic is designed to transform the standardized Abstract Syntax Trees generated by the parsers, allowing for modules to be composed together, each adding a piece of functionality as structures are passed from predicate to predicate. In this way, existing modules can make use of new functionality, new features can be implemented based on existing calculations and structures, and modules can be connected to as part of integrated user interfaces. Most of the changes required to existing code to enable the development of new features, consisted of modifying parsers and other procedures to expose more information to other modules of the program.

Chemlogic was successfully extended to implement new features, based upon the existing well-designed and modular structure, and can now perform stoichiometric calculations, convert chemical quantities between units and complete chemical reactions.

Earlier Work

The original version of Chemlogic was an experimental program I developed in 2012, to simplify my Science homework.

Based on these experiments, the project was extended in 2013 and 2014. The program was rewritten in Prolog, new algorithms were studied and adapted and many new features were implemented. The current modular design and structure of Chemlogic was implemented in Version 1, which supported balancing chemical equations, converting chemical names to formulas, and vice-versa. This version was presented at the 2014 CWSF in Windsor, where it received a Bronze Excellence Award.

In 2014 and 2015, Chemlogic was further developed to significantly expand the project, by continuing to improve existing features, adding a new user interface and adding support for higher levels of high school Chemistry. Version 2 of Chemlogic introduced the Android user interface and support for stoichiometric calculations, chemical unit conversions and reaction type analysis.

Acknowledgments

I would like to thank the many people who gave advice and helped with the project.

I am particularly grateful for the valuable assistance provided by Dr. Peter Tchir, now retired, my Physics, Chemistry and Computer Science teacher. His help and advice, especially with algorithms, and his support for my Computer Science projects helped make this program possible.

I would also like to thank Mr. Jason Peil for his assistance in designing the display, and Mr. Greg Osadchuk for his input and assistance regarding the visual presentation.

The technical and financial assistance of Selkirk College is greatly appreciated. I would like to thank Mr. David Feldman, School Chair of University Arts and Sciences, and Mr.

Ian Parfitt, Coordinator of the Selkirk Geospatial Resarch Centre, for their help in printing the display and their support for the project.

Obtaining Chemlogic / Contact

Nicholas Paun <np@icebergsystems.ca>

Chemlogic is open-source software. A copy of the program and additional information is available at http://icebergsys.ca/chemlogic

References

[1] C. Holzbaur. OEFAI clp(q,r) Manual Rev. 1.3.2. 1995.

[2] Nayuki Minase. Chemical equation balancer (JavaScript), 2013. URL: http://nayuki.eigenstate.org/page/chemical-equation-balancer-javascript.

[3] L. Sandner. *BC Science 10*. McGraw-Hill Ryerson, 2008. URL: http://books.google.ca/books?id=vEjRtgAACAAJ.

[4] Markus Triska. DCG Primer. URL: http://www.logic.at/prolog/dcg.html.

[5] Mark E. Tuckerman. Methods of balancing chemical equations. 2011. URL: http://www.nyu.edu/classes/tuckerman/adv.chem/lectures/lecture_2/node3.html.

[6] Jan Wielemaker, Zhisheng Huang, and Lourens van der Meij. SWI-Prolog and the Web. *Theory and Practice of Logic Programming*, 8(3):363–392, 2008.

[7] Jan Wielemaker, Tom Schrijvers, Markus Triska, and Torbjörn Lager. SWI-Prolog. *Theory and Practice of Logic Programming*, 12(1-2):67–96, 2012.

[8] Jan Wielemaker, et al. The string type and its double quoted syntax, 2014. URL: http://www.swi-prolog.org/pldoc/man?section=strings.