An Open-Source Java Platform for Automated Reaction Mapping

John D. Crabtree,*,† Dinesh P. Mehta,‡ and Tina M. Kouri‡

Department of Computer Information Systems, University of North Alabama, Florence, Alabama 35632 and Department of Mathematical and Computer Sciences, Colorado School of Mines, Golden, Colorado 80401

Received February 8, 2010

This article presents software applications that have been built upon a modular, open-source, reaction mapping library that can be used in both cheminformatics and bioinformatics research. We first describe the theoretical underpinnings and modular architecture of the core software library. We then describe two applications that have been built upon that core. The first is a generic reaction viewer and mapper, and the second classifies reactions according to rules that can be modified by end users with little or no programming skills.

INTRODUCTION

Software tools that can be used to evaluate combustion reactions are lacking in the chemical process industry. Due to this deficit, most researchers rely upon trial and error methods when constructing and comparing the suites of reactions that often number in the thousands. Each reaction will typically have kinetic and thermodynamic parameters that add to the complexity. Thermodynamic and reaction rate analysis of such combinatorially complex problems require automated systems. The manual classification of thousands of reactions is also tedious, if not impossible. Before either rate analysis or classification can take place, the reactions must first be mapped.

Chemical kineticists routinely generate large suites of reactions, called mechanisms, to represent some overall transformation of interest. Classification of the reactions in a mechanism is especially useful during the generation and reduction process. The mechanism generation algorithms are designed to systematically create all theoretically likely reactions which results in very large and unorganized mechanisms.[1−5] A mechanism reduction algorithm is typically used to reduce the number of reactions prior to running simulations. Mechanism reduction is a computationally expensive task which may take days to complete.[6−12] Prior to running a reduction algorithm, it is useful for the kineticist to sort the reactions, based on each reaction's classification, to verify that all of the important reactions and the reaction classes are included. After running the reduction algorithm, it is useful for the kineticist to sort the reactions, based on the each reaction's classification, to verify that no important reactions were discarded.

When working with bioinformatic databases, reaction mapping is also a prerequisite for the analysis of biological pathways, enzymatic reaction data, tracer experiments, and the consistency checking of pathway databases.[13] Large biological molecules often have very complex structures that can be nearly impossible to validate manually. Reaction mapping has also proven useful in identifying data anomalies in such databases.[14]

Our core software library contains a variety of reaction mapping algorithms that, due to their unique theoretical basis, are capable of optimally and efficiently mapping any reaction that can be represented as a set of chemical graphs. This library has been used as the foundation of novel software tools that have been successfully used to analyze combustion reactions and locate data anomalies. While the algorithms themselves have been described in detail elsewhere,[15] we believe that researchers in other areas will be able to make use of these algorithms and open-source tools in order to analyze chemical reactions in new ways.

REACTION MAPPING

Research into the process of identifying the bond changes that occur during a chemical reaction (i.e., reaction mapping) has a rich history that has been described by McGregor and Willett.[16] Most of the published algorithms involve finding the largest substructure common to similar compounds on both sides of the reaction called the maximum common subgraph (MCS).[17,18] Once the MCS has been identified, bonds that are not part of the MCS are assumed to have been broken or formed during the reaction. MCS is a problem that is in a class of NP-hard problems that include optimization problems for which no efficient algorithm has ever been found. Consequently, practical implementations generally use heuristics (which are not guaranteed to always find the MCS) or exhaustive searches which are time consuming and not guaranteed to finish.[13] Even if the MCS is found, there is no guarantee that the mapping produced will be optimal. In addition, MCS will find a mapping that minimizes the number of reaction sites, which may not minimize the number of bonds broken or formed.[13]

Most of the algorithms that do not use MCS limit the types of reactions that can be processed. One of Akutsu's non-MCS algorithms can be used on reactions that involve two reactants and two products.[13] Researchers at the Technical University of Catalonia have recently described their collection of algorithms that map reactions that fit into one of four classifications (combination, decomposition, displacement, and exchange reactions).[19,20] While the authors mention that most biochemical reactions can be classified into

* Corresponding author. E-mail: jcrabtree@una.edu.
† University of North Alabama.
‡ Colorado School of Mines.

10.1021/ci100061d © 2010 American Chemical Society
Published on Web 08/17/2010

$$R_1 + \cdots + R_g \Leftrightarrow P_1 + \cdots + P_h$$

Figure 1. Chemical equation modeled as a set of chemical graphs.

one of these four groups, no solution is presented for reactions with incompatible forms.

Elsewhere we have proven that the reaction mapping problem for general graphs belongs to the complexity class called NP-complete.[15] NP-complete problems include decision problems (i.e., problems that have a yes/no answer) for which no efficient algorithm has ever been found. Chemical graphs, in which each vertex denotes an atom and each edge corresponds to a bond, are a subset of general graphs. We have formulated the reaction mapping problem as an optimization problem where we attempt to minimize the number of bonds broken or formed (i.e., find a mapping of minimum cost). While this approach to solving the mapping problem is not new, we believe we are the first to provide a precise mathematical formulation of the problem and prove it to be NP-complete.[15] We have found, through empirical results, that minimum cost mappings are often chemically correct mappings as well.[15]

Instead of using the MCS problem as the basis of our algorithms, we use the arguably simpler graph isomorphism (GI) problem. If two graphs are isomorphic, then they are equivalent graphs and, in the case of chemical graphs, represent the same chemical compound. The GI problem is one whose complexity class is unknown. Some experts label GI and problems like it as isomorphism complete, meaning that a given problem is as provably hard as isomorphism.[21] We believe that our use of GI makes our algorithms unique in their application to any reaction regardless of classification or form and without the use of heuristics found in MCS-based approaches.

A reaction can be formulated as a set of chemical graphs representing reactants (R) and products (P), as shown in Figure 1.

Our algorithms encode the process of discovering a mapping of reactants (R) to products (P) such that the number of bonds broken or formed is minimized. Bonds and combinations of bonds are broken in turn until each chemical graph (R) has a corresponding isomorphic graph (P) on the other side of the equation.

If each chemical graph can be given a unique label (i.e., a name), then two chemical graphs can be identified as isomorphic if their names are the same. The first algorithm used to create unique names for chemical compounds is described by Morgan.[22]

Our library has been designed to use any canonical naming algorithm that implements our NamingStrategy Java interface as shown in Figure 2. We have included an implementation of Morgan's algorithm in our library. However, Morgan's algorithm is not guaranteed to work on graphs that are highly regular and may show oscillatory behavior in some cases.[23] Therefore, we have included two other naming algorithms. Faulon's algorithm produces a molecular descriptor called a signature which is based on extended valence sequences,[24] and Nauty is based on a method of finding the automorphism group for a graph.[25]

We have implemented a version of Faulon's algorithm in Java that implements NamingStrategy. The integration of the Nauty 2.2 package, which is written in C, was performed

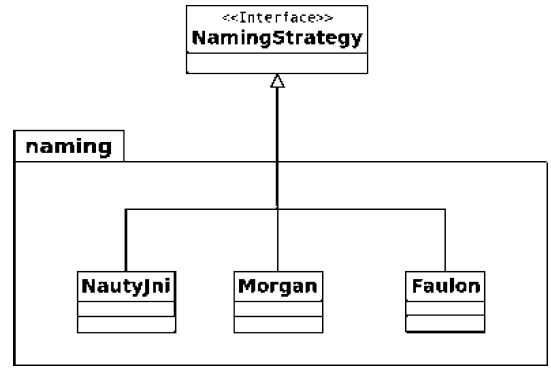

Figure 2. NamingStrategy interface and its implementing classes.

using the Java native interface (JNI). The JNI enables one to use a platform-specific library written in a different language within a Java environment. We have implemented JNI modules that interface with C libraries built for Microsoft Windows and Mac OS X.

Our five algorithms (Exhaustive, FewestBondsFirst, FBF-Symbolized, ConstructiveCountVector, and CutSuccessive-Largest) perform automated reaction mapping in a novel and robust manner. As mentioned above, previous approaches have focused on specific reaction classes (e.g., combination, decomposition, displacement, etc.),[19] particular reaction forms (e.g., two reactants forming two products, etc.),[13] or heuristic maximum common subgraph algorithms.[26] Each of our five approaches is capable of mapping any valid chemical reaction without rules or heuristics and regardless of class or form. In practice, four of the five algorithms perform this mapping in an efficient manner. The fifth algorithm, Exhaustive, is primarily used to evaluate the completeness and accuracy of the other four. Four of the five algorithms produce a mathematically optimal mapping. The fifth, CutSuccessiveLargest (CSL), produces an optimal mapping most of the time but is guaranteed to be efficient (a claim not shared with the other four).

The pseudocode for CSL follows:

(1) While the equation is not fully mapped:
  (a) Find the largest compound.
  (b) Find the best bond to remove. For each bond in the compound:
    (i) Remove the bond.
    (ii) Compare the resulting new equation, saving the removed bond that results in the most matching compounds as best.
  (c) Remove the best bond.
  (d) Match compounds.

The Exhaustive algorithm searches for patterns of broken bonds in the direction of increasing bond number. In other words, all combinations of breaking a single bond are considered until all structures can be matched or we run out of combinations. Internally, the combinations are modeled as a bit pattern where a one represents a broken bond and a zero represents no change. Then two broken bonds are considered and so on.

The pseudocode for Exhaustive follows:

(1) Create a bit pattern containing one bit for each bond in the reaction.
(2) Initialize the bit pattern to all zeroes.
(3) While there is a least one bit in the pattern set to zero:

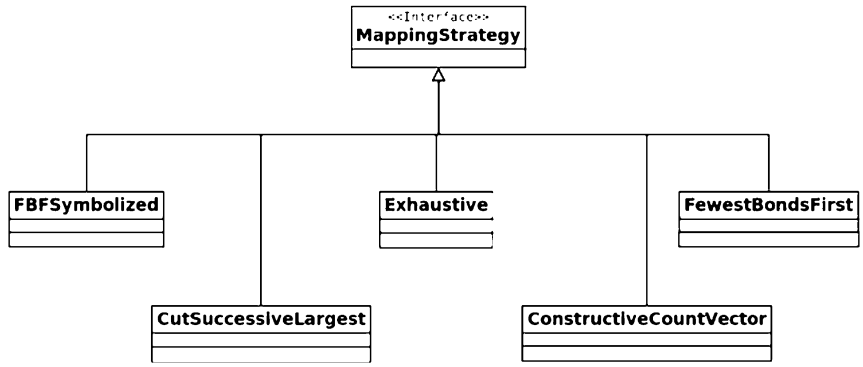

Figure 3. MappingStrategy interface and its implementing classes.

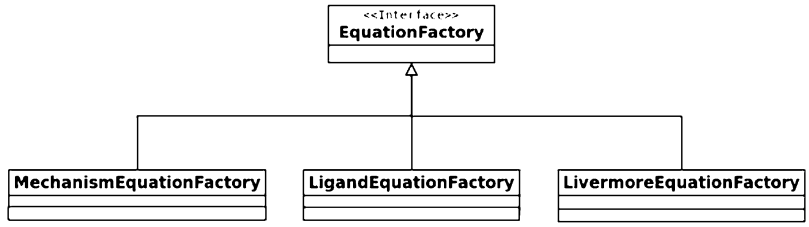

Figure 4. EquationFactory interface and its implementing classes.

```
public void testGRI317() throws Exception {
   Registry.setFilename("docs/molecules.xml.gri");
   EquationFactory theFactory = new MechanismEquationFactory();

   Equation e = theFactory.createEquation("317");
   MappingStrategy ms = new ConstructiveCountVector();
   ms.setEquation(e);
   ms.map();

   assertEquals("wrong number of total bonds cut", 2,
            ms.getNumberOfBondsBrokenOrFormed());
}
```

Figure 5. A simple unit test involving the GRI 3.0 database.

(a) Create a new equation by breaking each bond represented by a 1 in the bit pattern.
(b) If the equation can be completely matched, then save it as a possible solution.
(c) Increment the bit pattern by one.

As mentioned above, this algorithm is primarily used to check the correctness of our other algorithms, since it takes a brute-force approach in exhaustively searching the solution space for all possible mappings.

FewestBondsFirst and FBFSymbolized are progressive improvements toward our most efficient optimal mapping algorithm: ConstructiveCountVector (CCV). These three all use the bit pattern approach described above but use more efficient methods of searching the solution space. The best algorithm in this family (CCV) uses an advanced combinatorial algorithm that finds all bounded compositions. In practice, CSL can be used to obtain an initial mapping (efficiency is guaranteed) to produce an answer that can be verified by CCV, which is guaranteed to produce a minimum cost mapping. By using these two together, one may be able to efficiently find an optimal mapping. We have found that when these algorithms do not complete in a reasonable amount of time (on the order of seconds for combustion reactions and minutes for complex biomolecules), our chemical graphs have contained errors.

Each of the Java classes that encode a mapping algorithm implement the MappingStrategy interface as shown in Figure 3. This allows the software designer to separate the specific algorithms from their usage in larger applications.

DATA MANAGEMENT

We have tested our algorithms against three different data sources and are currently working with a fourth. Each database represents equations and compounds using different file and data formats. Therefore, we have isolated the generic data interaction from the specifics using the factory method design pattern.[27] The classes that encapsulate the specifics of the Gas Research Institute's 3.0 database of combustion reactions are contained in the MechanismEquationFactory and its helper classes. The LigandEquationFactory knows how to create objects from the KEGG LIGAND v20 database of molecules important to life, and the LivermoreEquationFactory deals with the n-C7 database of combustion data from the Lawrence Livermore National Laboratory, as shown in Figure 4. We are currently conducting related research using the Colorado School of Mines oxidation and pyrolysis reaction database, which was developed by chemical engineers using a subset of published pyrolysis and oxidation mechanisms.[28,29] The Colorado School of Mines oxidation and pyrolysis reaction database is formatted similar to the

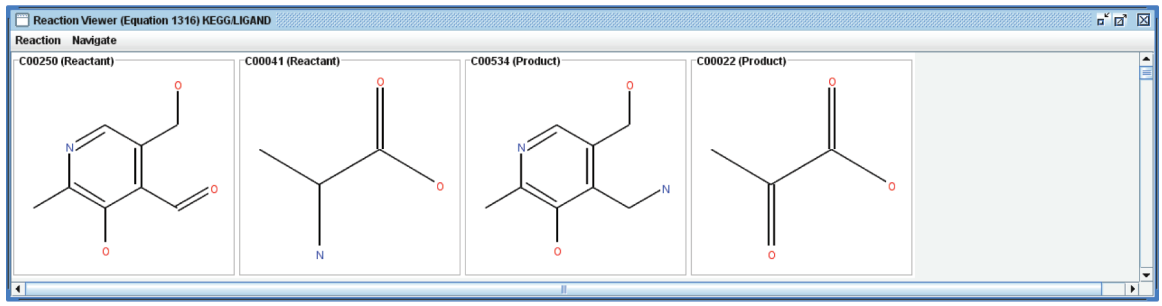

Figure 6. Equation 1316 from the KEGG LIGAND database.

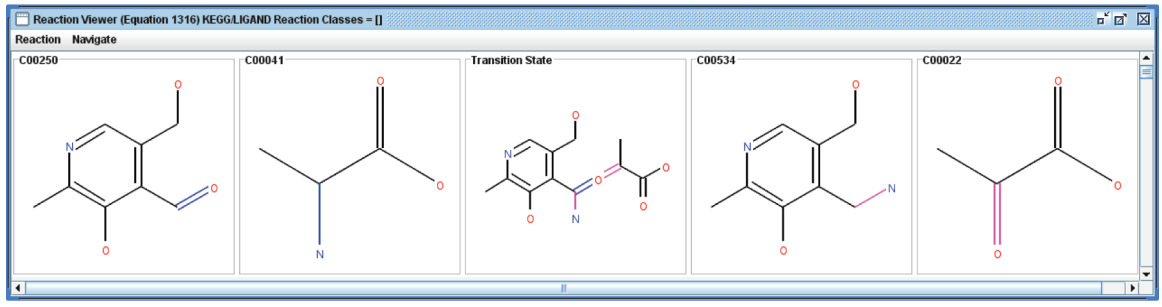

Figure 7. Mapped equation showing the transition state.

KEGG LIGAND v20 database and uses the LigandEquationFactory.

The core library includes unit tests that make use of the JUnit testing framework.[30] A suite of 34 tests has been included in the open-source distribution. In addition to ensuring the correct operation of the library, these tests also provide examples of how the core library and the factory classes can be used. The test shown in Figure 5 uses the MechanismEquationFactory and its supporting classes to construct equation 317 from the XML files that contain reaction and compound structure information.

VIEWER

The first application built upon the core library is used to map equations and display the results. The application was built with the Java Swing GUI library which was designed for maximum portability. This application will run unaltered on any platform that supports Java and the Swing extensions. The application can be configured at runtime to use any of our mapping algorithms, and menu items allow the user to choose reactions from any of the three databases presented above. Figure 6 shows one such reaction.

The viewer application uses code from the Chemistry Development Kit[31] (CDK) to display graphical representations of the compounds involved in the reactions. The CDK source code was altered to enable the color-coded display of the bonds that were broken or formed during the reaction as shown in Figure 7. In addition, we created a "transition state" molecule that shows the transitory combination of reactant molecules that occurs at a potential energy maximum. The CDK source code was also modified to support the display of the transition state.

The menu options available in the viewer enable the user to navigate through all of the equations in the database and, once an equation has been mapped, display all of the mappings of minimum cost in turn.

MECHA

Our second application incorporates the functionality of the viewer as well as the ability to view individual compounds in three dimensions using the Jmol library.[32] This application was built with the Standard Widget Toolkit (SWT) which uses native I/O libraries to provide better GUI performance and appearance at the expense of portability. This combustion research application, dubbed the Mechanism Analyzer, is used to analyze any given collection of reactions that model combustion. Reactions can be mapped and then classified using the Jess rule engine library,[33] as shown in Figure 8. This application uses packages that are not free or do not have open terms and/or licenses. Therefore, this application is not available for download and is presented here as an example of the kinds of systems that could be built using the core library.

The application can be configured via the menu to access any external database of reactions. These reactions are shown in a list, as shown in Figure 8. The user may expand a row that represents a single reaction to reveal its component species. Each compound may be selected and, with a right mouse click, viewed with Jmol as shown in Figure 8. The Jmol three-dimensional molecular display may be rotated with the mouse to view the compound from any angle. The center columns in Figure 8 display kinetic and thermodynamic parameters related to each reaction.

The user may initiate reaction mapping and/or classification via menu selections. Reactions must be mapped before they are classified. One may choose to have the system automatically classify reactions as they are mapped. If the algorithms encounter ambiguous mappings (i.e., there is more than one mapping of minimum cost), the expert user may investigate all potential min-cost mappings with the viewer to determine the chemically correct mapping. In testing, we have found that approximately 90% of the reactions have a unique min-cost mapping. In the GRI 3.0 reaction database, there are 376 reactions. Only six of those reactions have

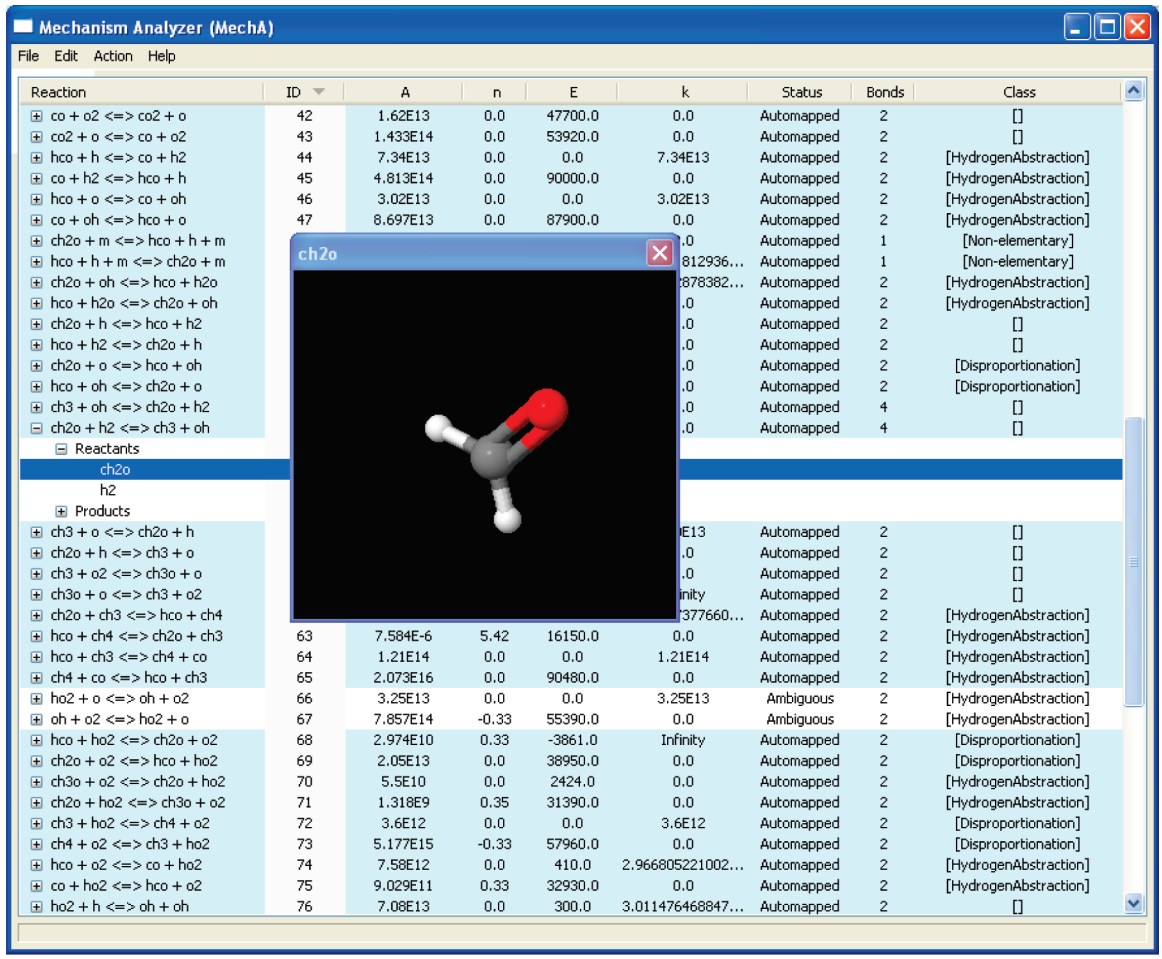

Figure 8. GRI reactions after mapping and classification.

ambiguous min-cost mappings (1.8%). We have also tested this using the Colorado School of Mines oxidation and pyrolysis reaction database. There are 3544 reactions, and only 376 of those reactions have ambiguous min-cost mappings (10.6%).

If a given reaction is mapped and classified and all of the optional min-cost mappings produce the same reaction classification, then the reaction is automatically classified. In other words, if the researcher is primarily concerned with classification, this will likely increase the automation rate above 90%. The user may also manually choose a mapping and/or classification. The user may highlight any of the collapsed rows in Figure 8, perform a right mouse click, and select the viewer which will display the reaction in a similar fashion, as shown in Figure 7. Alternatively, one may expand the reaction and view the compounds as described above.

The dialogue window shown in Figure 9 can be used to edit the rules that control reaction classification. The user may edit, create, or delete rules. These rules affect the behavior of the application but do not force the code to be recompiled, enabling an expert user without programming skills to modify the behavior of the classification engine at will. Since our algorithms contain no special rules or other hard-coded assumptions, we can easily allow the end-user to apply these classifications to the system without requiring internal changes. The user may also sort the list of classified reactions by their classification or any other column in the table by clicking on the column's label.

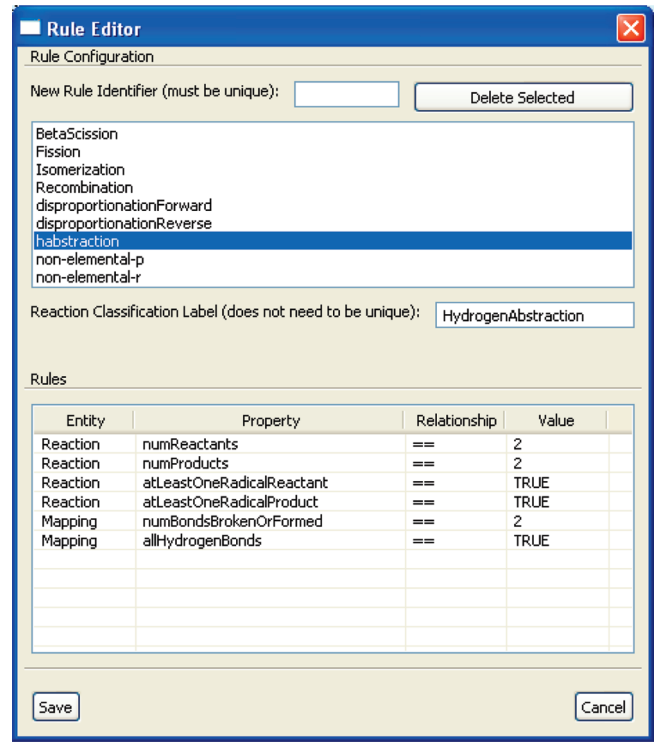

Figure 9. Rule engine editor.

The highlighted example in Figure 9 shows that if the reaction, before it was mapped, had two reactants and two products, at least one of each that was a radical, it will be

classified as "H-abstraction" if exactly two hydrogen bonds are broken or formed during mapping. The rules are applied to the mapped reaction using Jess, a Java-based rule engine.[33] If all of the rules evaluate to "true", then the reaction is classified with the given Reaction Classification Label.

We have been working with chemical engineers to devise rules based on their expert knowledge for combustion mechanisms. We provide rules to classify beta scission, fission, isomerization, recombination, radical addition, disproportionation, hydrogen abstraction, electronic transition, and complex reactions. The rules, which can be changed by the user, are used to classify the reactions after they have been mapped, and therefore, they do not affect the use of the algorithms in other areas, such as biomolecular research.

CONCLUSION AND FUTURE RESEARCH

We have introduced our free, open-source library of reaction mapping algorithms which is available at armsrc.sourceforge.net. Taken together, these unique algorithms are capable of mapping any reaction optimally and efficiently due to their unique foundation based on the graph isomorphism (GI) problem. They impose no limits on reaction form or classification and use no hard-coded assumptions or heuristics.

We define optimal mappings as those that break or form the fewest number of bonds. Our research has found that our algorithms produce optimal unambiguous mappings 90% of time when used on combustion mechanisms. These mappings are usually the chemically correct mapping. We are currently conducting research to quantify the chemical accuracy of these mappings, and the preliminary results look promising.

We have also demonstrated two of our applications that are based on this library. A version of the viewer is available for download at armsrc.sourceforge.net. MECHA is the first application of its kind in the area of combustion research and the first to use a rule engine to classify reactions. Due to its use of proprietary software, MECHA is not available for public download.

Currently, our algorithms do not deal with unbalanced reactions, but this may be a very interesting topic for future research, and we plan to investigate alternatives for dealing with this problem soon. One promising approach is presented by Félix and Valiente.[19,20]

REFERENCES AND NOTES

(1) Muharam, Y.; Warnatz, J. Kinetic Modelling of the Oxidation of Large Aliphatic Hydrocarbons Using an Automatic Mechanism Generation. *Phys. Chem. Chem. Phys.* **2007**, *9*, 4218–4229.

(2) Matheu, D.; Grenda, J. A Systematically Generated, Pressure-Dependent Mechanism for High-Conversion Ethane Pyrolysis. 1. Pathways to the Minor Products. *J. Phys. Chem.* **2005**, *109*, 5332–5342.

(3) Buda, F.; Bounaceur, R.; Warth, V.; Glaude, P.; Fournet, R.; Battin-Leclerc, F. Progress Toward a Unified Detailed Kinetic Model for the Autoignition of Alkanes from C4 to C10 Between 600 and 1200 K. *Combust. Flame* **2005**, *142*, 170–186.

(4) Nemeth, A.; Vidoczy, T.; Heberger, K.; Kuti, Z.; Wagner, J. MECHGEN: Computer Aided Generation and Reduction of Reaction Mechanisms. *J. Chem. Inf. Comput. Sci.* **2002**, *42*, 208–214.

(5) Cartensen, H.; Dean, A. Rate Constant Rules for the Automated Generation of Gas-Phase Reaction Mechanisms. *J. Phys. Chem.* **2009**, *113*, 367–380.

(6) Kovacs, T.; Zsely, I.; Kramarics, A.; Turanyi, T. Kinetic Analysis of Mechanisms of Complex Pyrolytic Reactions. *J. Anal. Appl. Pyrolysis* **2007**, *79*, 252–258.

(7) Shi, Y.; Ge, H.; Brakora, J.; Reitz, R. Automatic Chemistry Mechanism Reduction of Hydrocarbon Fuels for HCCI Engines Based on DRGEP and PCA Methods with Error Control. *Energy Fuels* **2010**, *24*, 1646–1654.

(8) Sun, W.; Chen, Z.; Gou, X.; Yiguang, J. A Path Flux Analysis Method for the Reduction of Detailed Chemical Kinetic Mechanisms. *Combust. Flame* **2010**, *157*, 1298–1307.

(9) Nagy, T.; Turanyi, T. Reduction of Very Large Reaction Mechanisms Using Methods Based on Simulation Error Minimization. *Combust. Flame* **2009**, *156*, 417–428.

(10) Liang, L.; Stevens, J.; Raman, S.; Farrell, J. The Use of Dynamic Adaptive Chemistry in Combustion Simulation of Gasoline Surrogate Fuels. *Combust. Flame* **2009**, *156*, 1493–1502.

(11) Pepiot-Desjardins, P.; Pitsch, H. An Efficient Error-propagation-based Reduction Method for Large Chemical Kinetic Mechanisms. *Combust. Flame* **2008**, *154*, 67–81.

(12) Straube, R.; Flockerzi, D.; Muller, S.; Hauser, J. Reduction of Chemical Reaction Networks Using Quasi-Integrals. *J. Phys. Chem.* **2005**, *109*, 441–450.

(13) Akutsu, T. Efficient Extraction of Mapping Rules of Atoms from Enzymatic Reaction Data. *J. Comput. Biol.* **2004**, *11*, 449–462.

(14) Goto, S.; Nishioka, T.; Kanehisa, M. Ligand: Chemical Database for Enzyme Reactions. *Bioinformatics* **1998**, *14*, 591–599.

(15) Crabtree, J.; Mehta, D. Automated Reaction Mapping. *J. Exp. Algorithmics* **2009**, *13*, 1.151.29.

(16) McGregor, J.; Willett, P. Use of a Maximum Common Subgraph Algorithm in the Automatic Identification of Ostensible Bond Changes Occurring in Chemical Reactions. *J. Chem. Inf. Comput. Sci.* **1981**, *21*, 137–140.

(17) Arita, M. In Silico Atomic Tracing by Substrate-Product Relationships in Escherichia Coli Intermediary Metabolism. *Genome Res.* **2003**, *13*, 2455–2466.

(18) Masaaki, K.; Yasushi, O.; Masahiro, H.; Susumu, G.; Minoru, K. Computational Assignment of the EC Numbers for Genomic-Scale Analysis of Enzymatic Reactions. *J. Am. Chem. Soc.* **2004**, *126*, 16487–16498.

(19) Felix, L.; Valiente, G. Efficient Validation of Metabolic Pathway Databases. In Proceedings of *Sixth International Symposium on Computational Biology and Genome Informatics*; World Scientific Press, Inc.: Salt Lake City, UT, 2005.

(20) Felix, L.; Valiente, G. Validation of Metabolic Pathway Databases Based on Chemical Substructure Search. *Biomol. Eng.* **2007**, *24*, 327–335.

(21) Pemmaraju, S.; Skiena, S. Algorithmic Graph Theory. In *Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, 1st ed.; Cambridge University Press: New York, 2003.

(22) Morgan, H. L. The Generation of a Unique Machine Description for Chemical Structures - A Technique Developed at Chemical Abstracts Service. *J. Chem. Doc.* **1965**, *5*, 107–113.

(23) *Representation of Chemical Compounds. Chemoinformatics: A Textbook*; 1st ed.; Gasteiger, J., Engel, T., Eds.; Wiley: Weinheim, Germany, 2003.

(24) Faulon, J.-L.; Collins, M. J.; Carr, R. D. The Signature Molecular Descriptor. 4. Canonizing Molecules Using Extended Valence Sequences. *J. Chem. Inf. Model.* **2004**, *44*, 427–436.

(25) McKay, B. Practical Graph Isomorphism. *Congr. Numer.* **1981**, *30*, 45–87.

(26) Hattori, M.; Okuno, Y.; Goto, S.; Kanehisa, M. Development of a Chemical Structure Comparison Method for Integrated Analysis of Chemical and Genomic Information in the Metabolic Pathways. *J. Am. Chem. Soc.* **2003**, *125*, 11853–11865.

(27) Gamma, E.; Helm, R.; Johnson, R.; Vissides, J. Creational Patterns. Design Patterns: Elements of Reusable Object-Oriented Software; 1st ed.; Addison-Wesley Professional: Reading, MA, 1994.

(28) Randolf, K.; Dean, A. Hydrocarbon Fuel Effects in Solid-oxide Fuel Cell Operation: An Experimental and Modeling Study of n-hexane Pyrolysis. *Phys. Chem. Chem. Phys.* **2007**, *9*, 4245–4258.

(29) Naik, C.; Dean, A. Detailed Kinetic Modeling of Ethane. *Combust. Flame* **2006**, *145*, 16–37.

(30) Cheon, Y.; Leavens, G. A Simple and Practical Approach to Unit Testing: The JML and JUnit Way. In *ECOOP 2002 - Object-Oriented Programming*; Springer-Verlag: Berlin, 2002.

(31) Steinbeck, C.; Han, Y.; Kuhn, S.; Horlacher, O.; Luttmann, E.; Willighagen, E. The Chemistry Development Kit (cdk): An Open-source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Model.* **2003**, *43*, 493–500.

(32) Herraez, A. Biomolecules in the Computer: Jmol to the Rescue. *Biochem. Mol. Biol. Educ.* **2006**, *34*, 255–261.

(33) Friedman-Hill, E. *Writing Rules in Jess. Jess in Action: Java Rule-based Systems*, 1st ed.; Manning Publications Co.: Greenwich, CT, 2003.

CI100061D