Building Robust Chemical Reaction Mechanisms: Next Generation of Automatic Model Construction Software

by

Jing Song

B.S., Tsinghua University, 1995
M.S., Tsinghua University, 1998
M.S.C.E.P., Massachusetts Institute of Technology, 2000

SUBMITTED TO THE DEPARTMENT OF CHEMICAL ENGINEERING IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY IN CHEMICAL ENGINEERING

AT THE

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

FEBRUARY, 2004

© Massachusetts Institute of Technology 2004. All rights reserved.

Jing Song
Department of Chemical Engineering
December 2003

William H. Green, Jr.
Associate Professor of Chemical Engineering
Thesis Supervisor

George Stephanopoulos
Arthur D. Little Professor of Chemical Engineering
Thesis Supervisor

Daniel Blankschtein
Professor of Chemical Engineering
Chairman, Committee for Graduate Students
Abstract

Building proper reaction mechanisms is crucial to model the system dynamic properties for many industrial processes with complex chemical reaction phenomena. Because of the complexity of a reaction mechanism, computer-aided reaction mechanism generation software appeared in recent years to help people efficiently build large reaction mechanisms. However, the limitations of those programs, such as not being able to model different types of reaction systems and to provide sufficiently precise thermodynamic and kinetic parameters impede their broad usage in real reaction system modeling.

Targeting the drawbacks in current first-generation reaction modeling software, this thesis presents the second-generation of reaction mechanism construction software, Reaction Mechanism Generator, (RMG). In RMG, a new reaction template method is proposed to help quickly and flexibly define different types of reaction families, so that users can easily characterize any reaction system of interest without modifying the software. Furthermore, this work also presents new functional group tree approaches to construct hierarchical structured thermodynamics and kinetics databases for managing a large number of parameters, so that people are able to quickly and precisely identify better kinetics for different reactions in the same reaction family and to easily extend and update the databases with the latest research results. This new data-model dramatically improves the interface between the chemistry and computer-science, removing many of the ambiguities that have plagued the field of chemical kinetics for many years, and greatly facilitating the maintenance and documentation of both the software and the databases that provide the key inputs to any chemical kinetic model.

The author applied object-oriented technology and unified modeling language in system analysis, architecture design, and implementation of RMG. Therefore it is designed and developed into a robust software with good architecture and detailed documentation, so that this software can be easily maintained, reused, and extended.

RMG is successfully applied to generate a reaction mechanism for n-butane low temperature oxidation, which includes a complex autoignition process. The model generated by RMG caught the fundamental phenomena of autoignition, and the predicted ignition delay time and many major products' yields are in very good agreement with experimental data. This is the first time that model generation software automatically generated such a complicated reaction mechanism without human interference, and provided precise predictions on ignition delay and major products yields consistent with experimental data.
Dedication

This thesis is dedicated to my parents, Song Wenbin and Guo Yuanzhi, and my husband, Yi Li. Their love and support always provide me great power and strength to explore new areas.
Acknowledgement

RMG is a large interdisciplinary project, requiring cooperation from people with many different backgrounds. The author has received great helps from others; without them, RMG would not be successfully developed so quickly.

Firstly, I want to thank my two advisors, Prof. William H. Green and Prof. George Stephanopoulos for their great help, advices, and encouragements, which drive me to explore and learn new things every day. I also want to extend my thanks to my Ph.D. thesis committee, for their time, interest, and suggestions on my work.

I have to thank Dr. Sumathy Raman, who has given me great help, advice, and suggestions on designing the RMG system. She has an in-depth understanding on chemical reaction kinetics, and initiated many important ideas of RMG, such as introducing functional group concept into reaction modeling and building hierarchy tree structured kinetics databases. She did a great amount of work at the first stage to define the documentation of reaction families for RMG, and constructed almost all the reaction family trees for their kinetics databases. She also did many quantum calculations on some important reaction families to enrich our kinetics database. I’d like to thank her for her hard work and great help on implementing all those RMG fundamental databases and libraries.

There are many other people in Green Group contributing to RMG project greatly. Joanna Yu built the hierarchy tree structured thermodynamics database for RMG thermodynamics property estimator, and she also provided ideas on several important algorithms in thermal properties estimation. Catherina D. Wijaya helped Dr. Sumathy Raman in building some hierarchical structured kinetics databases, and she also collected kinetics rate rules from literature and assigned them properly into our RMG kinetics database. I truly thank them for their hard work on building useful databases for RMG.

I’d also like to thank Dr. Dave Matheu and Dr. Preeti Aghalayam, at the beginning design stage of RMG, for providing helpful documentations on defining reaction family. Dr. Dave Matheu also explained carefully to me the pressure dependence algorithm, developed in his PhD thesis, and helped me implement this
algorithm in RMG. I also appreciate that he gave his modified version of CHEMDIS and THERFIT for RMG pressure dependent kinetics calculations.

Dr. Pisi Lu had very constructive and helpful discussion with me about how to apply RMG into Computational Fluid Dynamics calculation. I really appreciate his experience in industry and useful suggestions. Bryan Wong helped me calculate several pressure dependence kinetics rates by hand, and Paul Yelvington provided me a revised version of a reverse kinetic fitting program, originally written by S. Raman, and helped me search for important software tools through the internet. I’d like to thank them and all other Green Group members for their help, interests and suggestions on my thesis.

Finally, I’d like to present my thanks to my husband, Yi Li, who has been giving me great help, supports, and encouragements on completing my thesis. With a computer science major, he also provided me much useful information on software engineering and software development, and we had many useful discussions on some fundamental design issues of RMG.

This work was financially supported by U.S. Department of Energy through grant DE-FG02-98ER14914 and through an NSF CAREER award to Prof. William H. Green (CTS-9875335).
Table of Contents

Chapter 1 Introduction

1.1 Computer-aided chemical reaction modeling .............................. 10
1.2 Review of first-generation reaction modeling software .................. 12
1.3 Problems in first-generation reaction modeling software ............... 16
1.4 RMG: second-generation of reaction modeling software ............... 23
1.5 Chapter overview .................................................................. 27

Chapter 2 Object-Oriented Technology .............................................. 32

2.1 Object-oriented software development ...................................... 32
2.2 Challenges in developing large software .................................. 35
2.3 Software development life cycle .............................................. 36
2.4 Unified Modeling Language (UML) ........................................ 40
2.5 Conclusions and Discussions ................................................ 46

Chapter 3 Fundamental Data Structures ........................................... 47

3.1 Graph ............................................................................... 48
3.2 Tree ............................................................................ 53
3.3 Design chemUtil package in RMG ....................................... 55
3.4 Conclusions and discussions ................................................ 62

Chapter 4 Fundamental Chemical Objects in RMG ......................... 63

4.1 ChemGraph and Species .................................................... 63
4.2 FunctionalGroup ................................................................ 68
4.3 Design of chem Package .................................................... 73
4.4 Conclusions and Discussions ................................................ 79

Chapter 5 Group Additivity Thermodynamic Properties Estimator .......... 80
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1 Roles of thermodynamic properties in reaction system</td>
<td>80</td>
</tr>
<tr>
<td>5.2 Overview of thermodynamic properties</td>
<td>81</td>
</tr>
<tr>
<td>5.3 Group additivity method</td>
<td>84</td>
</tr>
<tr>
<td>5.4 Improvements of group additivity method in RMG</td>
<td>89</td>
</tr>
<tr>
<td>5.5 Design highlights for RMG-GATP</td>
<td>112</td>
</tr>
<tr>
<td>5.6 Conclusions and discussions</td>
<td>117</td>
</tr>
<tr>
<td>Chapter 6 Generating Elementary Reactions by Reaction Templates</td>
<td>121</td>
</tr>
<tr>
<td>6.1 Overview of chemical elementary reaction</td>
<td>121</td>
</tr>
<tr>
<td>6.2 Reaction Family</td>
<td>122</td>
</tr>
<tr>
<td>6.3 Modeling reaction families by reactions templates</td>
<td>125</td>
</tr>
<tr>
<td>6.4 Obtaining elementary reaction parameters</td>
<td>131</td>
</tr>
<tr>
<td>6.5 Design of rxn package in RMG</td>
<td>146</td>
</tr>
<tr>
<td>6.6 Conclusions and discussions</td>
<td>156</td>
</tr>
<tr>
<td>Chapter 7 Automatic Reaction Mechanism Generation</td>
<td>176</td>
</tr>
<tr>
<td>7.1 Introduction of a reaction mechanism</td>
<td>176</td>
</tr>
<tr>
<td>7.2 Algorithm for reaction mechanism generation</td>
<td>177</td>
</tr>
<tr>
<td>7.3 RMG approach for reaction mechanism generation</td>
<td>183</td>
</tr>
<tr>
<td>7.4 Design of rxnSys package in RMG</td>
<td>193</td>
</tr>
<tr>
<td>7.5 Conclusions and discussions</td>
<td>207</td>
</tr>
<tr>
<td>Chapter 8 Generating n-Butane Low-Temperature Oxidation Mechanism</td>
<td>215</td>
</tr>
<tr>
<td>8.1 Introduction</td>
<td>215</td>
</tr>
<tr>
<td>8.2 Generate n-butane low-temperature oxidation</td>
<td>215</td>
</tr>
<tr>
<td>8.3 Results</td>
<td>217</td>
</tr>
<tr>
<td>8.4 Conclusions</td>
<td>259</td>
</tr>
</tbody>
</table>
Chapter 9  Valid Parameter Range Analysis ......................................................... 261
9.1 Motivations .............................................................................................. 261
9.2 Methodology for valid parameter range analysis ..................................... 264
9.3 Relation of model, flexibility and tolerance ............................................. 283
9.4 Generalization of valid range analysis algorithm ...................................... 290
9.5 Conclusions and discussions .................................................................. 292

Chapter 10 Conclusions and Recommendations................................. 294
10.1 Conclusions ............................................................................................ 294
10.2 Recommendations on future work .......................................................... 300
10.3 Final Conclusion ..................................................................................... 306
Chapter 1  Introduction

Chemical reaction phenomena happen in many important industrial processes, such as pyrolysis, combustion, polymerization, etc. People have long been working on modeling those important reaction processes into detailed reaction mechanisms to reveal and predict the system dynamic properties [Dente et al., 2003; Curran et al., 2002 and 1998; Donskoi et al., 2000; Glaude et al., 2000; Daugaut et al., 1999, 1996, 1994, and 1991; Warth et al., 1998; Barbe et al., 1995; Brown et al., 1995; Tan et al., 1995; Wilk et al., 1995; Kojima, 1994; Wilk et al., 1990; Ranzi et al., 1994; Chakir et al., 1992, 1991, and 1990; Hoffman et al., 1991].

Understanding the fundamental principles involved in those reaction processes is necessary for different purposes. As a means of understanding the underlying phenomenology of the reaction process in terms of elementary reactions and participating species, reaction kinetic mechanisms have long been employed in the chemical industry for interpolative tasks such as process optimization and control, and, as the models improve, they are increasingly used for extrapolation, e.g. to help guide R&D on new processes. Reaction kinetic models have also been employed in public policy decision-making, e.g. the Clean Air Act regulations. Some famous reaction mechanisms in pyrolysis and oxidation systems include the SPYRO mechanism for steam cracking of ethane and naphtha, the GRI [Smith, et al.] and Leeds [Hughes, et al., 2001a, 2001b, 2001c] mechanisms for natural gas combustion and NOx formation, and the LLNL [Curran et al., 2002 and 1998] mechanisms for heptane and octane combustion.

1.1  Computer-aided chemical reaction modeling

Although reaction models have long been used in different areas in different ways, constructing reaction models has never been an effortless job. Even in a seemingly simple reaction process, hundreds or thousands of species and reactions might be involved to form a very complex reaction network; therefore, to enumerate all the species and reactions as well as to manage this complex reaction system are mostly beyond human being's capability. Computers, as powerful computation tools, have been more and more used in helping people fulfill their goals in reaction modeling.
There are two major stages in the history of constructing chemical reaction mechanisms. Before computers became such a valuable quantitative analysis assistant in recent decades, people studied reaction phenomena mainly by experimental means, from which the dynamics of important species were measured, and the kinetics of individual reaction pathways were investigated. Schematic reaction mechanisms composed with a small number of chief species and reactions were then manually constructed by chemistry experts to help understand the underlying phenomenology. People, of course, saw many limitations of this modeling procedure. First of all, people could only catch the dynamic profiles of a limited number of intermediate species, and the uncertainties involved in the measurements could be quite large. Meanwhile, kinetics derived from such measurements might have large error bars. Secondly, the schematic reaction mechanism built manually might miss important intermediate species and reaction paths; it might also use misleading kinetics data; as a result, it is mostly considered as a qualitative description instead of a quantitative model. Finally and most importantly, the procedure demanded too much manpower and time. For instance, the famous GRI mechanism for methane oxidation required several years’ effort by a rather large group of collaborators to reach its present good shape.

After the development of the first electronic computer ENIAC in 1940’s, people have witnessed a blooming development of computer technology in both hardware and software in the second half of twentieth century. Computers are now involved in almost every aspect of our lives. Chemical reaction modeling, which is very computationally demanding, has been a proper application case for computer technology. Not like the first stage when computer programs were just some small tools for fitting kinetics data, in the second stage, much larger and more powerful computer software had been developed to fulfill much bigger goals. For example, people began to apply basic laws in quantum mechanics in molecular modeling, and one of the most famous software, Gaussian, allows people to predict the energies, optimize the molecular structures, and estimate important properties of molecular systems in a computational way. By combining the results from the quantum mechanical calculations with statistical mechanics and Transition State Theory [Eyring, 1935; Evans and Polanyi, 1935], software was developed for calculating thermal data for complicated molecules and kinetics for
elementary reactions. Secondly, software for automatically generating complex reaction mechanisms has been successfully developed by several groups [Chinnick et al., 1988; Hillewaert et al., 1988; Chevalier et al., 1990; Froment, 1991; DiMaio and Lignola, 1992; Quann and Jaffe, 1992; Broadbelt et al., 1994, 1995, 1996; Blurock, 1995; Ranzi et al., 1995; Prickett and Mavrovouniotis, 1997; Prickett and Mavrovouniotis, 1997a, 1997b, 1997c; Warth et al., 2000; Battin-Leclerc et al., 2000]. Such software has been applied in modeling reaction processes like pyrolysis [Susnow et al., 1997; Prickett and Mavrovouniotis, 1997;], combustion [Warth et al., 2000 and 1998; Glaude et al., 1998; Zeigarnik et al., 1997; Come et al., 1996; Ranzi et al., 1995], and etc., and the agreements between the computer models’ prediction and experimental measurements are generally satisfying. Finally, people also applied computer tool in reaction mechanism analysis, such as model reduction [Bhattacharjee et al., 2003 and 2001; Law et al., 2003; Vora and Daoutidis, 2001; Petzold et al., 1999 and 1997; Massias et al., 1999; Edwards et al., 1998; Genyuan et al., 1994], sensitivity analysis [Turanyi, 1997 and 1990; Campolongo and Saltelli, 1997; Milford et al., 1992; Rabitz et al., 1987, 1984, and 1983; Larter and Clarke, 1985; Kramer et al., 1984; Larter et al., 1984; Hwang, 1983; Edelson and Allara, 1980], uncertainty analysis [Balakrishnan et al., 2002; Iepapertritou and Androulakis, 1999; McRae et al., 1997; Phenix et al., 1998], etc., to help analyze properties of the reaction models. Experimental studies, certainly, are not retiring in this stage; however, it is more and more combined with quantum chemical predictions and with automatic mechanism generation methods to provide necessary verification of theoretical modeling and to improve understanding of many not-well-understood phenomena in chemical kinetics [Goos et al., 2001a, 2001b, and 2000; Glaude et al., 2000; Fotache et al., 1995; Dagaut et al., 1995, 1994, 1991; Barbe et al., 1995; Hoffman et al., 1991].

1.2 Review of first-generation reaction modeling software

In this section, a detailed review on first-generation reaction mechanism construction software is provided for the reader to understand the strengths and weaknesses of the current available model generation software.
1.2.1 Basic features of reaction model generation software

As guided by Tomlin [Tomlin, Turanyi, and Pilling, 1997], reaction mechanism generation software should have the following four basic features:

1. A unique and unambiguous representation of chemical species;
2. Rules for manipulation and management of molecular and reaction information;
3. Estimation methods to arrive at thermo chemical and kinetic data;
4. Rules to prune unimportant reactions in the mechanism by means of thermodynamic and thermo kinetic criteria.

Most of the contributions, in what we would like to call first-generation codes, have been in the area of identifying a unique representation for molecules, devising protocols to perform chemical reactions and designing easy-to-handle, effective and meaningful termination criteria to prevent combinatorial explosion during mechanism generation. Thus, although the significant prior works undoubtedly provide valuable basic tools needed for mechanism generation, there is still room for further improvement with respect to chemistry viz., definition and classification of reaction types, estimation procedures for thermo-kinetic data etc.

1.2.1.1 Representation of Molecules and Reactions

In the literature, different representations were developed based on graph theory, substitution matrices and Boolean algebra to internally represent a molecule and to generate product species.

Quann and Jaffe [1992], in their “structure-oriented lumping” (SOL) method, employed a 22-element vector representation, where each element stands for a structural building block. Product generation rules in SOL add and delete elements from the vector, and, in the case of multiple products, iteratively generate the product vectors with the appropriate stoichiometry. The vector, however, does not represent an exact molecular structure as it is unaffected by rearrangement of the building blocks.
Yoneda [1979] represented the structure of each species as a square matrix and his reaction generator GRACE searched for elementary reactions using combinatorial algorithm by rearranging the matrix elements. Since no reaction types were defined, it would probably lead to many meaningless reactions. Froment [1991] used Boolean matrices to represent hydrocarbons, which are then manipulated to form products. However, for certain reactions Boolean algebra entails raising the matrix to the fourth power and thus becomes costly for computational resources. Also isomorphism was not considered, while a library was maintained to ensure that each intermediate and product is produced only once.

Chevalier et al. [1990] used logical programming in LISP to generate the oxidation reaction mechanism involving aliphatic hydrocarbons. Their philosophy is similar to that of Froment and they employed a limited number of reaction types. Acyclic species were canonically represented as a tree rooted at a unique point and it thus allowed for identification of equivalent molecules. Blurock [1990 and 1995] developed the X-window based program, REACTION for the manipulation, management and generation of molecular and reaction information. The required input is a set of reactions represented as a set of connection tables usually MOL files. Using graph difference algorithm, Blurock derives the atom-atom correspondences (Reaction patterns) between the reactions and products. The pattern of the reaction is matched with the same pattern within a reactant molecule. The application of reaction pattern on a molecule results in a specific reaction.

DiMaio and Lignola [1992] employed the bond and electron (BE) matrix representation of Ugi et al. [1979 and 1994] for molecules and constructed oxidation reaction mechanisms through the addition of a reaction matrix to BE matrices. They adapted the notion of a graph invariant to handle isomorphism and used the calculations of the permutation-invariant eigenvalues of the BE matrix to reveal equivalency. However, this isomorphism rule is questionable as it was shown that matrices with the same eigenvalues are not necessarily equal [Foulds, 1992].

Broadbelt et al. [1994, 1995, and 1996] followed and improved the isomorphism rule in DiMaio and Lignola’s approach for model generation. They also used BE matrices
to represent reactants and reactions. The reaction matrices contained only the atoms involved in the reaction. The off-diagonal elements represent the formation or breaking of bonds, while the diagonal elements contain information concerning the loss or gain of free valence electrons. A graph invariant unique string code was derived for both non-cyclic and cyclic species by constructing the structurally explicit decomposition tree. A lexicographical comparison of the unique string code allowed unambiguous determination of species uniqueness.

Prickett and Mavrovouniotis [1997a, 1997b, and 1997c] developed a syntax computer language for describing generic reaction types with additional flexibility to define any type of reaction system in the course of model generation and also to adjust/prune the level of detail within a specific reaction network. Representation of molecules in this work takes two forms: an object or graph oriented representation for internal use in reaction and a line notation comprising of an ordered list of symbols representing atoms, bonds and charges for the input and output. This 1-D line notation differs from the input representation of EXGAS [Warth et al., 2000], with respect to the nested-list format and the default assumptions about bond-order and description of chemical bonds and suffixes. The internal representation employed in EXGAS is a canonical treelike structure for both acyclic and cyclic molecules and radicals.

Very recently, Lengyel et al.[Lengyel et al., 2003] have developed a mechanism generation program based on the SMILES [Weininger, 1986] string representation of molecules, using the commercial package distributed by Daylight Inc.

1.2.1.2 Mechanism Halt

The commonly employed criteria for mechanism halt are the upper bound of an atom counts in the product species and the order in which a product species appear in a reaction mechanism (rank). The original version of NetGen employs [Broadbelt et al., 1995] a combination of both criteria (species rank-based) for the rational halt of mechanism generation. There exists also a second school of thought [Ranzi et al., 1995, 1997, 2001] in automated algorithms, which aims for a compact mechanism rather than the comprehensive one. The success of this approach stresses the importance of chemistry in automated algorithms. However, it starts with a validated C_0-C_n (n \leq 4)
reaction base and comprehensively generates only the primary mechanism using an algorithm and incorporates a secondary mechanism, which is based on the lumping of both species and reactions. In strict sense it is not an automated approach but pruned and tuned by the user using one's knowledge of chemistry and kinetics. All the termination approaches have the potential for missing important species and including many insignificant species. The most significant contribution in this field is the introduction of the rate-based termination algorithm [Susnow et al., 1997]. The rate-based algorithm uses the rate and concentration information when constructing the reaction scheme and the generated mechanism depends on the rate estimates. As shown by Susnow et al [1997], both the size of the reaction mechanism generated and the calculated concentration vs. time profiles using this approach rely heavily on the thermochemical and rate constant estimates employed. Even with the best termination criterion, the accuracy of the generated kinetic model is limited by the accuracy of the employed rate constants.

1.2.1.3  Thermochemical and Kinetic Parameter Estimation

Benson’s concept of group additivity [Benson, 1976] provides the backbone for thermochemical estimation and is widely employed as a database or module (THERM [Ritter and Bozzelli, 1991], RADICALC [Bozzelli and Ritter, 1993], THERGAS [Muller et al., 1995], CHETAH [Downey et al., 1994]) in all model generation codes. Usually, the needed kinetic parameters are calculated using Quantitative Structure-Reactivity Relationships (QSRR) described by Nigam and Klein [Nigam and Klein, 1993], and Sumathi and Green [Sumathi and Green, 2002a]. Elementary reactions were categorized into reaction families for which entropy of activation or the A factor is taken as a constant and differences in reactivity within a reaction family because of substituents are correlated in terms of their effect on activation energy. An Evans-Polanyi relationship was used to correlate activation energy with the heat of reaction.

1.3  Problems in first-generation reaction modeling software

Although much automatic reaction mechanism generation software has been written and applied in modeling reaction processes, none of them is widely distributed and used for modeling real reaction systems, and people still feel quite unsatisfied with many existing problems in most of those computer programs. Complaints generally arise
from two groups: chemists who are using the software and software developers who are
developing and maintaining the software. If one is both a chemist and a software
developer, he surely experiences these problems from both sides.

1.3.1 Chemists' concerns

Chemists are the final users of reaction modeling software. Their comments and
level of satisfaction are the most important issue for evaluating the software. Most
concerns from chemist users lie in two aspects: the software unambiguousness/usability
and the chemistry flexibility/extendibility.

1.3.1.1 Software unambiguousness and usability

Since there are so many factors, like reaction conditions, user's tolerance on
omitting unimportant species and reactions, reaction types included in reacting system,
etc, affecting the mechanism generation, users have to make a lot of choices carefully
before running the software. As chemist users, instead of getting confused with all those
ambiguous tags and flags, they basically require a clear definition and documentation of
all the possible choices, so that they understand the exact meanings of commands they
send to the software. Similarly, after the software generates a reaction model, the output
of the model and its corresponding thermal and kinetic data should be straightforward to
chemists to understand and utilize. The key for realizing unambiguousness is to maintain
documentation definitively illustrating all the details in the user interface. The meaning
of each tag or flag inputted and outputted should be uniquely defined and updated, so that,
by reading such documents, software developers are able to write the software the same
way as chemists desire it to be, while the chemists are able to use the software the same
way as the software developers designed it to work. Most present reaction model
generation software ignores this aspect, which could potentially cause a lot of
misunderstandings or even fatal errors.

Another important issue is the software usability. Like users of any software,
chemists users look for an easy way to use the software, and the Graphical User Interface,
GUI, could be a good solution to it. In software wrapped with a GUI, both input and
output are dealt with by interacting graphic windows, and online help documents are
always available to help the user understand his selections and the results the software
provides to him. Of course, people can also stay with non-graphic user interface like ASCII-format input/output files, and in such case, the file formatting issue could be critical for users. In much current reaction model generation software, little attention was paid to this aspect, which probably hindered the adoption of this software.

1.3.1.2 Chemistry flexibility and extendibility

Chemistry flexibility and extendibility are the two top issues chemists complained about in first-generation reaction model generation software. Everyday, papers about new research results in reaction engineering are published to bring out more and more information to chemical society. They could be about new theories and algorithms on calculating molecule properties; they could be about discovery of new elementary reaction paths; they could be about new experimental data for species thermal property; they could be about new quantum calculation results for individual reaction. Chemists always want to investigate the value of such information and update their knowledge databases accordingly. Therefore, if reaction mechanism construction software couldn’t accommodate enough chemical information and allow it to be efficiently updated, the software would soon become out-of-date and, therefore, be abandoned.

Unfortunately, most reaction model generation software before this work is not able to handle chemistry flexibility and extendibility very well. Many chemical details were pre-defined and fixed inside the software. There was very little flexibility for chemists, who can do little in revising and improving models and data embedded in the software if they are not experienced computer programmers. For example, in most software, the reaction types included are hard-coded in the software, i.e., the possible reaction types, such as H abstraction reactions, beta scission, cyclo-addition, etc., are fixedly defined inside the software, and there is no flexible way for users to add or modify the definitions. If a chemist without any programming experience wants to change the present reaction type information or to add any new type of reactions for his special reaction system, either he has to explain his idea carefully to the software developer and ask him or her to add a bunch of new codes, or he has to pick up an introductory programming book to learn the computer language. Therefore, most of the time, chemists feel very uncomfortable using such software, since they more or less find
themselves bound to some limited chemical functions locked in the existing software, and they can do very little to improve it.

Another very important issue in reaction model generation is how to handle a large number of thermodynamic and kinetic parameters. As we know, a reaction mechanism includes many parameters: each species has its specific set of thermodynamic parameters, and every individual reaction has its kinetic parameters. If we just simply put all available data into one huge database without a good structure, managing and searching such database could introduce many problems for reaction mechanism generation. For the first-generation reaction modeling software, in order to simplify the database-managing problem, people generally used approximate parameters. For example, although there exists hundreds of available kinetics parameters for different types of H abstraction reactions, many software just used one or a couple of raw numbers for all types of H abstraction reactions. This could cause a big problem for a complex reaction system, since, for example, it is not reasonable that reaction \( \text{OH} + \text{CH}_4 \rightarrow \text{H}_2\text{O} + \text{CH}_3 \) and \( \text{CH}_3 + \text{C}_6\text{H}_6 \rightarrow \text{C}_4\text{H}_5 + \text{C}_6\text{H}_5 \) will have the same kinetic rate rule. Readers might be wondering why people could not put all the available reaction kinetics rates into the database. The reason lies in two aspects. Firstly, for the first-generation reaction modeling software, the detection of the reactants and its reacting sites are hard-coded, and therefore, it is very difficult to identify the different subgroups of one big type of reactions. For this H abstraction reaction, for example, it is quite easy to identify the reactants candidates' structure, which is only a single bond with a hydrogen atom and a radical site, but the rate depends on more specific information, such as what atom is bonded to the hydrogen atom, and what is the radical type, etc. Programming such detailed chemistry information into the software directly is very hard and time-consuming. Furthermore, since such procedure would typically have a big block of nested (if, then) statements, it is very difficult to debug this kind of structure, or to modify it without introducing new bugs. Therefore, people try to avoid subcategorizing one type of reaction into many subgroups with different kinetics in their software. Hence, they neglect a large number of the good kinetic rates available in literature. Obviously, chemists are not satisfied with such results, since they work very hard to estimate good parameters for each reaction system, and if the reaction modeling software can not use
their results and can never update its thermodynamic and kinetic parameters in time according to the latest research, they will be disappointed with such software.

In terms of the requirements from chemists, the users of reaction modeling software, the most important drawbacks of the first-generation software are: (1) the software is not implemented in a user-friendly way, (2) the present software does not include enough chemistry details to make a good modeling and prediction of real reaction systems, and (3) users are not able to define and modify the chemistry in a flexible and extendable way.

1.3.2 Software developers’ concerns

The last section shows that current reaction model generation software is far from chemistry perfection, and, therefore, maintaining and modifying present software to have more and better chemical functions are the jobs software developers face every day. On the other hand, the software discussed here is not a simple one with just several thousand lines that can be managed easily; instead, it is a large program with almost a hundred thousand lines. Therefore, the software developers are facing large-scale software needing frequent updates and modifications.

Assume that you are responsible for maintaining such software, and one day you are asked to fix a bug or add new functionality to the software package. How can you do that job? You should first figure out the functionality of the corresponding part in the software, and then make the modifications. However, if there is no design documentation for the software architecture, no detailed explanations on how objects were implemented and related to each other, how can you understand the code first? The only way left is to figure out the software functionality by reading through thousands of lines of source codes. Fixing the bug and adding the new function maybe only take you a hour, while trying to understand the code first could eat up to weeks or even months of your time. Moreover, it is not so rare that even if you work very hard and carefully, you still have quite a lot of misunderstandings of the source code and, as a result, you cannot really fix the bug or correctly add the new functionality.
The key point the author tries to address here is that the communication among software developers should never move down to the source code level. Unfortunately, this is what happens for much existing software.

1.3.2.1 Software architecture

Software architecture serves as a detailed blueprint for software development, based on which people can easily grasp the highlights of the system, clearly exchange design and development ideas, reasonably distribute work, and quickly implement units. Just as building a big house without a blueprint is completely impossible, developing large software without a well designed and thoroughly investigated architecture would be almost guaranteed to fail. Even if a problematic-structure program were, by any chance, written successfully without an architecture design, the corresponding jobs of testing, debugging and modifying it could be big trouble. Furthermore, even if a blueprint was drawn, but was not sufficiently detailed and well-structured, programmers would still pay much extra efforts and time to fix the potential problems later. Many times, software development projects are canceled in the middle of or even after completing coding primarily because of the bad architecture.

During the development of first-generation reaction modeling software, people paid most of their attention to modeling the chemistry details, such as teaching the computer to recognize chemical species, making rules to generate different types of reactions, etc. The main point was to realize and demonstrate the idea. The issues of analyzing software composition and designing a reasonable architecture for it held a lower position or were even completely omitted from the software developers’ to-do list. At the beginning stage, this seemed okay; but once the initial difficulties of modeling chemical details in a reacting system were gradually overcome, people found the size and complexity of software had already jumped to a very high level, and managing it without an architecture blueprint was impossible.

1.3.2.2 Reusability and extendibility

Software reusability and extendibility have been discussed in many software engineering textbooks. In a simple word, reusability means that fundamental software components, once they are created, should be easily reused for many different
applications; extendibility means those component should be easily extended to have more functionalities. Those two issues have been discussed frequently in recent years when object-oriented programming, OOP, has become a major trend in software development. The major strength of OOP is that it provides a fundamental programming methodology to implement the software reusability and extendibility. For other types of programming methods, such as procedural programming approach, although it is possible to implement reusability and extensibility, the efforts demanded are fairly large and the procedures are very difficult and complex.

For the first-generation reaction modeling software, considering the computation efficiency, people mostly chose to use procedural languages, like C and Fortran, for programming. Since those procedural programming languages are not that powerful in implementing software reusability and extendibility, anytime people need to add new feature or modify the present procedures, they should spend a lot of time reading the existing source code to figure out how the program was implemented, where and how to correctly insert the new parts, and how those modifications will affect different parts of the original system. Furthermore, when people add a new procedure, even if it is very similar to one of the existing procedures, it is often not quite safe to reuse the existing code directly. To avoid conflicting with the present program, people usually have to rewrite many procedures with similar functionalities, causing the system to be cumbersome.

The author and some other previous researchers from our group had such experience when they were working on our first-version model generation software, ExxonMobil Mechanism Generator, XMG [Grenda et al., 2000 and 1998]. Before RMG, this group used XMG, a 60,000-line reaction model generation program written in C, for generating and studying kinetics models. The original XMG can generate kinetics models for alkane pyrolysis at fixed reaction conditions, using only the high-pressure limit rate rules. To improve it, the author aimed to add some uncertainty and sensitivity analysis procedures and use them as the supplementary information to guide the model generation procedure; D. Matheu tried to add an internal loop inside the original model generation iteration to account for the pressure dependent networks; Dr. P. Aghalayam tried to enlarge the original pyrolysis-only reaction family set in XMG into an oxidation
reaction family set. All three of us working on XMG had the same experience: it is too difficult to understand XMG’s structure and to make any changes. It took each of us several months to understand only part of XMG and another several months to modify and debug the program, even if our algorithms were straightforward and clearly defined.

In summary, although the first generation reaction construction software has successfully implemented many important fundamental features and can be used for modeling some reaction systems, such software can be hardly considered as a powerful modeling tool that can be conveniently used and controlled by chemistry users, and it also can hardly be maintained and extended by software developers for further improvements.

1.4 RMG: second-generation of reaction modeling software

Targeting the problems existing in first-generation reaction model generation software, the author developed our Reaction Model Generator, RMG, to implement chemical flexibility and extendibility and software reusability and extendibility.

As introduced earlier, there are four basic features of model generation software: (1) a unique representation of chemical structure (2) the rules for manipulation and management of reaction information, and (3) estimation methods for thermodynamic and kinetic data, and (4) termination rule to avoid generating unimportant species. For aspects (1) and (4), people have put a lot of efforts into them, and have generally agreed on how to implement them. The major contributions of RMG are in aspects (2) and (3). In RMG, we provide users a flexible and extendable way to define reaction types and manipulate reaction generation, and also propose a new method of systematically managing a large number of thermodynamic and kinetic data so that users can easily construct and update their database.

On the other hand, since the difficulties of developing large-scale software are fully understood, the author proposes to use object-oriented technology, one of the most advanced software development methodologies occurring in the latest 50 years, in developing RMG. More importantly, the author also realized the importance of the software architecture design issue, and used unified modeling language, UML [Booch et
al., 1999; Rumbaugh et al., 1999; Fowler and Scott, 2000], the most advanced and popular complex system modeling language, in the system analysis and software architecture design for RMG.

1.4.1 Chemistry flexibility and extendibility

In order to implement chemistry flexibility, we proposed using a functional group representation for modeling reaction families and hierarchy trees for constructing the thermodynamics and kinetics databases.

1.4.1.1 Categorizing chemical structure into functional groups

Functional group is a common chemical concept. In Morrison’s Organic Chemistry [Morrison, 1992], it is defined as: “The atom or group of atoms that defines the structure of a particular family of organic compounds and, at the same time, determines their properties is called the functional group.” It is obvious that functional groups are the keys to categorize chemical species into groups with similar reactive functionalities, so that people can study the reactive feature of a group and apply the results to all the species in the same group. Therefore, functional group builds a bridge between chemical species structure and chemical reactions, and it is, therefore, a crucial concept in generating reactions from chemical structure and categorizing the reactions by its reactants and products structure.

Before this work, people have not introduced generic functional group objects into any automatic model generation programs, and any categorizations of chemical species were pre-defined in the software. In this work, the author and Dr. S. Raman designed functional group objects in our new model generation software, and developed unique graph representation for functional groups. With these new functional groups objects introduced, users can conveniently define functional groups to categorize chemical structures into groups with similar reactivities. The author also developed a matching algorithm for quickly identifying the sub-graph relations between any chemical species and functional group, so that RMG can quickly recognize a chemical species’ reactivity. This is the key to RMG’s ability to handle user-defined reaction families, and for constructing its thermodynamics and kinetics database.
1.4.1.2 Modeling reaction families by functional groups

Reaction family represents a special type of reactions. Important reaction families in pyrolysis and oxidation system include H abstraction, H migration, radical addition, cyclo-addition, disproportionation, etc. As introduced above, all those reaction families are pre-defined in first generation reaction modeling software, which has been proved to be a non-flexible way to describe chemistry details. Furthermore, such programming approaches make it very difficult to later introduce new reaction rules and to modify the present ones.

In RMG, with the functional groups definition, the author proposed a new way to define the reaction family. In this new method, the reactants of any reaction family are defined by functional groups, and the reacting actions like breaking bond, losing radical, etc, are defined by standard graph mutations. Since RMG now offers users a way to flexibly define the functional groups and reacting action outside of RMG, the whole reaction families can be easily defined and modified without changing the program. This new feature makes it possible to clarify the interface between the chemistry details and the software details, and, therefore, the flexibility and extensibility with respect to the chemistry are easily implemented.

Using this data-model based reaction family definition, we successfully built 34 reaction families, accounting for all the important reaction types in pyrolysis and oxidation for C/H/O systems, which is the richest reaction family set among all the automatic reaction model generation software.

1.4.1.3 Building hierarchy tree structured database

The author has discussed the importance of including a large number of thermodynamic and kinetic parameters into reaction modeling software. In designing RMG, the author and Dr. S. Raman also proposed a new methodology to manage the thermodynamics and kinetics database: the hierarchy tree method. In this method, all the databases are constructed into hierarchical trees, instead of the traditional unordered lists. Hierarchical tree, as a data structure, is very powerful at storing and quickly searching hierarchically ordered data. In RMG, we took advantage of the tree data structure, and
constructed our databases into hierarchical structure, so that to build, search, and extend such database is convenient, efficient and quick.

Using this hierarchy tree method, we successfully built the tree-structured thermodynamics group database and kinetics rate rules databases for all reaction families. We also collected about 100 thermodynamic group values and 1000 kinetics rate rules from literature, calculation, and experiments to fill our database. Those libraries are the richest ones that have ever been used in any automatic reaction model generation software.

1.4.2 Software architecture design and object-oriented technology

In order to implement software reusability and extendibility, we used object-oriented technology in RMG development. At the same time, the latest and advanced modeling language, unified modeling language, is also used in system analysis and architecture design of RMG.

1.4.2.1 Unified modeling language

As the author has discussed in section 1.3.2, neglecting or paying inadequate attention to the step of complex system analysis and software architecture design before the software is implemented causes the common problems in much software. To solve the problem, people have been discussing and working on building a standard for system modeling. Unified modeling language, UML, has been successfully developed for this purpose, and it has been more and more used in commercial software design and development. Many application tools implementing UML-style system modeling and implementation have appeared in these days.

In RMG, the author used UML for the whole RMG development process, from system analysis and architecture design at the very beginning until the final implementation and testing. All RMG packages are designed and documented following the rules of UML. In this thesis, the details of RMG design and implementation will also be introduced in UML format in the following chapters.
Since RMG is well designed and documented in UML, it is easy for other people to read the graphic blueprint of RMG, and therefore they could quickly understand the RMG architecture and implementation as well as modify and extend it.

1.4.2.2 Object-oriented technology

Object-oriented programming is the most advanced software development methodology. The object-oriented concepts, such as inheritance, encapsulation, polymorphism, etc., if used in a proper way, make it possible to develop software with good structure, reusability and extendibility. The most popular and widely used object-oriented programming languages are C++, Java, and C#.

In developing RMG, the author applies object-oriented technology through the whole software development process, and Java was chosen as the implementation language for RMG. Developed by object-oriented technology, RMG provides a lot of useful objects, such as Graph, ChemGraph, Species, FunctionalGroups, Reaction, Kinetics, etc, which can be easily reused by some other applications, and they can be also easily inherited and extended to form new objects with improved features. For example, the ChemGraph object is now implemented to model 2-dimensional graphs, but it can be easily extended into a new object modeling 3-dimensional graphs by simply introducing the coordinates of graph nodes.

In summary, object-oriented technology and unified modeling language are two powerful new methodologies used broadly in large-scale software design and development nowadays. In development of RMG, the author made use of both of them to build a good architecture and to implement software reusability and extendibility.

1.5 Chapter overview

This thesis covers a wide range of information on both chemistry and software engineering. Readers interested in different parts should focus on different chapters.

1.5.1 Chapter 2:

Chapter 2 introduces the object-oriented technology, a new programming methodology appearing and becoming dominant in software development area. Object-
oriented technology focuses on development of robust software components before building the whole software system, and to reuse and extend those software components for further development and maintenance is proved to be easy and efficient.

Unified Modeling Language, as a modeling language, provides a powerful tool for system analysis and software architecture design, which will be the foundations of developing good object-oriented software. In chapter 2, the author gives a brief introduction to UML and its important notations that are used everywhere in this thesis to explain the structure of RMG.

Readers not interested in the software design and development details of RMG could skip this chapter. People who are going to work on the RMG as software developer and are not familiar with OOP and UML, are encouraged to read this chapter and some of the reference books mentioned in this chapter carefully.

1.5.2 Chapter 3:

Chapter 3 reviews two important data structures (Graph and Tree), their definition, representation, and standard methods and operations. The author also discusses in detail how Graph and Tree are designed and developed to satisfy the fundamental requirements for chemical structures. Chapter 3 also describes the software design and implementation issues of Graph and Tree objects in the RMG package, “chemUtil”.

If readers are not interested in the detailed review of those basic data structures, they can simply read sections 3.1.4 and 3.2.4 about how Graph and Tree were designed and implemented in RMG, which are the basis for designing and developing RMG fundamental objects for modeling chemical species and functional groups, as well as the objects for building tree-based databases.

1.5.3 Chapter 4:

Chapter 4 discusses two important chemical structures implemented in RMG (ChemGraph and FunctionalGroup) using Graph data structure. ChemGraph represents the chemical species, and FunctionalGroup represents the functional groups used to subcategorize chemical species. To model chemical structure, the author extends the Graph data structure to ChemGraph for accommodating more features and functionalities.
for chemical aspects. FunctionalGroup is a new software concept in modeling chemical systems, and the author also extends the Graph data structure modeling chemical functional groups. The details on how to model and implement those two objects are given in this chapter. Section 4.3 describes the design and implementation details of ChemGraph and FunctionalGroups in the RMG package, “chem.”.

The author strongly recommends readers to read this chapter carefully, especially for the part on building functional group objects, which is crucially important for later understanding of the reaction family definition in chapter 6.

1.5.4 Chapter 5:

Chapter 5 introduces how RMG deals with thermodynamic properties estimation for chemical species. The group additivity algorithm proposed by Benson and widely used for thermodynamic property estimation has been employed in RMG. Furthermore, we proposed a new data-model based method to manage the thermochemical groups in a systematic and extendible way to increase the efficiency of the algorithm and database management. In this chapter, the fundamental ideas on the data-model based group additivity estimation methods and the hierarchy-structured database management will be discussed in detail. This part of work was implemented in RMG package, “chem.”, also. The software design and implementation issues for thermodynamic properties are provided in section 5.5.

Readers that are familiar with the group additivity methods by Benson can skip the introductory sections 5.1 to 5.3 for this algorithm, and go directly to section 5.4 to understand how RMG improves the implementation of this algorithm and how RMG manages a large number of thermodynamic group data into a hierarchy tree structure. The author strongly recommends readers to read section 5.4.2 carefully, where the hierarchy tree idea is first proposed and described in detail.

1.5.5 Chapter 6:

Chapter 6 provides detailed information on how to generate reactions. In RMG, reaction families are defined to describe the reacting patterns that reactant species can undergo, and this is also so-called reacting rule. In this chapter, the author introduces our
new data-model driven methodology to model relation families. In this way, chemists are able to define detailed reaction rules outside of the software, and they can change and enlarge the reaction family definitions at any time without touching the code. Similar to the thermal groups database management, we also used hierarchy tree structure to model the reaction kinetics database, which has been proved to be a fast, easy, and extendible way to handle a large amount of kinetics data. The details on those two aspects will be provided in this chapter. Besides the methodology, in this work, we also built an entire reaction family set for pyrolysis and oxidation reaction system with totally 34 reaction families. We also built the hierarchy-tree structured databases for all the forward reaction families to accommodate a large number of reaction kinetics. Furthermore, about 1000 reaction kinetics rate rules from the literatures are collected to build our reaction kinetics database. All the details mentioned above are covered in chapter 6. This part of the work was implemented in the RMG package, “rxn”, and the software design and development issues for this package are described in section 6.5.

The author strongly recommends all readers to read this chapter to understand how we handle the reaction generation. Our new ideas on data-model driven reaction family modeling and hierarchy tree structured database construction are addressed in good detail.

1.5.6 Chapter 7:

Chapter 7 discusses the iterative rate-based model generation algorithm, originally proposed in Susnow, et al., [1997]. We enrich the original rate-based iterative model generation procedure with an internal iteration for exploring pressure dependent effects on individual reaction kinetics, proposed by Matheu [2003]. In order to correctly generate an oxidation model, a small molecule reaction library based on the Leeds methane oxidation mechanism is also built to account for the irregular small molecule reactions that cannot be generated from regular reaction families. In RMG, rxnSys is the package designed for implementing the model generation algorithm, and the details on rxnSys package are provided at the end of this chapter, section 7.4.

Readers familiar with Susnow’s rate-based iterative reaction model generation algorithm may skip section 7.2. The author recommends readers to read carefully section
of 7.3 if they are using RMG to generate reaction models, since it provides detailed documentation of RMG’s model generation process.

1.5.7 Chapter 8:

Chapter 8 provides model generation examples for the reaction systems, n-butane low temperature oxidation. The reaction system conditions and the model generation results are presented in this chapter, and the analysis for the results is also be provided. The successful model generation practice in this complicated reaction system shows that RMG can be applied to model reaction systems with very complex dynamic properties.

1.5.8 Chapter 9:

Chapter 9 presents a new algorithm for investigating the valid parameter range of the models generated from the rate-based model generation rule. The algorithms for identifying the valid range of parameters for a given model, for generating a model robust over a range of reaction condition, and for estimating the error associated with using a model over a range of conditions are introduced in this chapter. An application using those algorithms in methane pyrolysis case will be discussed. Those examples were generated by our first-generation of reaction modeling software, XMG [Grenda et al., 2000 and 1998], but the methodology can be applied to RMG models.

1.5.9 Chapter 10:

Chapter 10 finally gives the summary and the conclusion of this thesis, and it also discusses some future works on RMG, such as developing a Graphic User’s Interface, GUI, improving the thermodynamics estimation for cyclic structures, and enlarging the set of reaction families and kinetics libraries.
Chapter 2  Object-Oriented Technology

Object-oriented software design and development technology is considered as one of the most important milestones in software engineering. In this chapter, the author will give a review on object-oriented technology concept, its history, and its application.

2.1 Object-oriented software development

2.1.1 Software development methodology

In the history of software development, there have been three major programming methodologies: procedural programming, structured programming, and object-oriented programming.

2.1.1.1 Procedural programming

In procedural programming, people try to decompose a large program into small functional procedures, and each little procedure is considered as an individual module that has special functionality. Any procedure can be seen as a process of taking input data, processing it, and returning the result data. Widely used procedural languages are C and Fortran, and many popular scientific application programs were written in those languages.

Although procedural programming has many advantages, such as it is simple and easy to learn and use, it is not very good for developing large-scale, complicated application programs. Imagine that there are thousands of processes in one program, the problems of how to organize them in a systematic way and how to make them communicate easily and correctly are quite hard using the procedural programming approach. Debugging and testing a large-scale procedural program is also hard because of the unrestricted way that functions communicate to each other. Furthermore, if people want to add any new feature to such software, to find out what procedures will be affected and to efficiently make the changes are not easy, either. Targeting those problems, people developed structured programming methodology.
2.1.1.2 Structured programming

Structured programming is an improved methodology from procedural programming. In structured programming, people are concerned more about the structure of the whole software. The typical methodology used in structured programming is a so-called "top-down" system analysis and abstraction, where people are firstly more interested in the highest-level abstraction, and then break down the largest goal into small ones that can be implemented by smaller modules. In this approach, programmers generally break larger pieces of code into shorter subroutines that are small and can be understood easily. The advantage of this method is that it provides programmers a clear system structure diagram, which later guides them to write the program in a systematic way. The drawback is also very obvious that once the system structure is determined, the desire to change at any level in the "top-down" analysis might cause big troubles.

This idea was popular and widely used for a while in 1970s, (and it is still quite popular in some groups of programmers nowadays), before the object-oriented programming methodology stepped onto the stage.

2.1.1.3 Object-oriented programming

Different from the procedural programming and structured programming that both build everything on the basis of procedures, object-oriented programming methodology takes a total different angle analyzing a complex system based on objects.

In object-oriented programming, people are more interested in building the whole system from small pieces, the so-called objects. A lot of effort is applied in modeling and abstracting an object, its composition, and its behaviors. Meanwhile, people also pay a lot of attention to study the proper relations between objects to answer the questions, such as how they can communicate to each other and how they are related to each other. Once the system objects are understood and developed, people can easily assemble objects into large software to satisfy the requirement.

Important object-oriented concepts include: object, class, inheritance, encapsulation, polymorphism, etc. Object is an entity collecting related data and methods. In software development, objects are often used to model real-world objects in everyday
life, like anything you see, you feel, or even you imagine. Class is a prototype that models an object. Class basically has attributes modeling the data stored in it, and methods corresponding to the operations that can be applied to its attributes. Inheritance is the ability to create a new class by adding to or overriding parts of an existing class. The existing class is called superclass, and the new class inheriting the superclass is called the subclass. Polymorphism is associated with the inheritance concept, and it means the ability to implement different subclasses from the superclass to represent different functionalities of an object. Encapsulation means the ability to provide users with a well-defined interface to a set of methods in a way that hides their internal workings.

The most important advantage of object-oriented programming is reusability. In object-oriented programming, reusability implies that the objects designed and developed by someone at sometime, can be easily reused for other people for other purposes. For example, if someone has already developed a class List, which models the list data structure object, when later you need to use a list, you can simply make use of the List class in your own application. Furthermore, if you need a list with slightly different functionality, you can also inherit the functionality from the old List class and extend it into a new type of subclass of the original List for your specific goal. This is the second advantage of object-oriented programming, extendibility. Another advantage arise from the encapsulation, which means the details implementing any objects will not be exposed to the users of objects, and only an interface will be provided. This offers an easy way for the users to understand the functionality of the objects, and, at the same time, also provides the safety for the original objects by keeping the internal information inside objects.

Object-oriented programming is a concept or a methodology used in software development. To implement the concept, object-oriented programming languages are required to do the concrete work. In the history of object-oriented programming, there have been many different languages. SIMULA I (1962-65) and SIMULA 67 (1967) are the first two object-oriented languages proposing the idea of object, class, inheritance, etc. [Lamprecht, 1983]. SmallTalk is another early well-known object-oriented language, which introduced the graphical user interface concept [Lambert and Osborne, 1997].
C++, developed by Stroustrup [1992], is considered the most famous object-oriented language, and has been widely used in large-scale software development. Java and C#, are two recent and very popular object-oriented languages widely used in many Internet and IT applications nowadays.

In summary, the programming methodologies have been studied for years, and people experienced procedural programming, structured programming, and object-oriented programming stages. Object-oriented technology has been very popular in recent years in developing large-scale software, and it has been proved an efficient methodology to develop a software system in a flexible, reusable, and extendable way.

2.2 Challenges in developing large software

Although the object-oriented programming concept has been introduced many years ago, and mature object-oriented programming languages have been developed and widely used in many different areas, people still face a big challenge when they develop large-scale software.

According to the statistics from Gibbs [1994], on average, many computer programs need more than 50% longer development time than originally planned; 75% of them are operational failures; 25% of them are finally cancelled. Why there are so many troubles in developing large software? The difficulties lie in several aspects. Firstly, human being have limited understanding of a system at one time. Therefore, for a large system, people might have troubles to understand the system, to construct a good structure for the software, to foresee and avoid all the possible problems involved in the system. Secondly, even if people have crystal-clear understanding of the system, without a systematic software development methodology, it is very possible that people will design a program with a bad architecture. Without a good architecture, debugging, maintaining, and modifying a code could be extremely difficult, even for experienced software developers. Thirdly, even if a good architecture of a program has been built, how to program the components and how to assemble the components into large software are not easy. Finally, even if, at every step mentioned above, people pay enough attention to the possible pitfalls, they still might make bugs in their program, and some efficient tools for debugging and testing are necessary for developing any software.
In a word, development of large high-quality software, i.e., programs that are reliable, efficient and easy to understand, modify and control, is challenging, and people have been working very hard on developing systematic methodologies to accomplish this challenging goal. Software engineering, an important field in computer science, aims to develop a systematic approach to the analysis, design, implementation and maintenance of large, complex software.

2.3 Software development life cycle

To understand the difficulties in large software development and maintenance, the author firstly introduces the well-known software life cycle concept. Software life cycle [Liskov and Guttag, 2001] refers to the phases that software goes through from when it is created to when it is discarded and the corresponding software engineers’ activities during different phases in the lifetime of software.

![Software Development Life Cycle Diagram](image)

Figure 2.1 Software Development Life cycle

As shown in Figure 2.1, the software life cycle can be roughly divided into five phases: system analysis, architecture design, implementation, testing, and maintenance and modification.
2.3.1 System analysis

System analysis is to analyze the needs of the customer. In some cases, the customer needs are described in a clear, precise, and complete manner, but in most circumstances, they are defined in some very vague manners, and it is not rare that even the customers don’t have a full understanding of their needs at the very beginning of software development. Therefore, the most important purpose in this stage is to define the software needs explicitly, and to clarify any misunderstanding between the software developers and customers.

Generally, in this stage, documentation specifying the customer’s need in details is produced to describe the functionalities of the software. In the following stages, this documentation serves as a necessary communication tool between the software developers and customers, and of course, it can be revised according to the changes of customers’ requirements and software development progress. However, taking time at the beginning phase of software development to make a detailed and unambiguous system requirements analysis is necessary to make efficient progress at the other stages.

2.3.2 Architecture design

Architecture design is to decide the how a complex system can be decomposed into components, how the components relate to each other, and how they fit together to function as a whole software. In this stage, the documentation specified in system analysis is usually used as a guide, and the system architecture design should be consistent with the requirements of the customers.

Architecture design is the crucial part for developing any large software. As people need a construction blueprint before they build a house, people need to have a clear description of the whole picture of the software before they actually begin implementation. In the early days of software development, people used to just write small programs to help them do small and easy things. In those cases, programs are simple and manageable without complicated architecture design step. With the rapid development of computer system capabilities, computers become more and more powerful in people’s everyday life, and more and more complicated and large-scale software is required to handle complicated jobs. For such cases, architecture design
becomes an indispensable, crucial step before implementation, and people draw this conclusion from many real failures in large software development projects, where design issues were not paid enough attention to. Seeing more and more lessons, people gradually realize the importance of architecture design in the whole software life cycle.

A good architecture design should meet all the functional and performance requirements from the system analysis documents, and should be able to define a modular structure, in which all the components are the proper abstractions of real-world objects, the relations between components are reasonably simple and easy to understand and implement, and the whole structure should be made in a clear way so that it can be easy to extend and modify later. To achieve this goal, people have developed many good methodologies and corresponding tools aiding people do a better job in architecture design. Unified Modeling Language, UML, is the most famous and popular methodology, and is widely used in many commercial software applications development.

2.3.3 Implementation

Implementation is to write all the components and their relations, and to finally compose everything into the whole software, based on the system architecture design.

In this stage, each individual module or component is implemented using concrete programming language. If people put enough efforts into the previous architecture design step, this implementation stage should be straightforward and easy. Actually, software programmers working in just implementation are sometimes considered as “blue collars” in software engineering, while the architecture designers are considered as “white collars”. Although this is more or less a joking way to distinguish design and implementation, it makes an important point that implementation should strictly follow what is decided in the design stage to make the whole program perform correctly.

In implementation, the most important issue it to choose a programming language. In the history, there are three main stages associated with computer programming language evolution. In the first stage, people used machine language composed with zeros and ones, and some simple fundamental language like assembly, to do very simple programming. At the second stage, more so-called high-level languages, such as C and Fortran, were created to make it easier to write powerful programs. Those languages are
also called procedural languages, since the fundamental modules in such languages are procedures and functions. In the third stage, because of the object oriented technology being introduced in recent decades, people now more and more rely on object oriented languages, such as C++, Java, C#, etc, to develop their software. The key improvement of object-oriented technology is to design and implement reusable, extendable objects instead of procedures and functions, and object-oriented languages generally provide convenient tools for such purpose.

2.3.4 Testing

After the implementation, the software should be tested to guarantee its correct performance. To test implemented software, it should be executed on a set of test cases, and the actual running results should be compared with the expected results to make sure that the program performs correctly.

The purpose of testing is to successfully catch the bugs in the programs. To efficiently do so, people should be careful of designing test data to check not only the normal conditions but also many unusual or extreme conditions that might cause problems.

An associated issue with testing is debugging, what should be done once bugs are caught. To make the testing and debugging process easy and efficient, people should be aware of the possibility that fixing bugs in one place might cause a new bug or new bug(s) somewhere else. To avoid that, people should, again, pay more attentions and time in the architecture design stage, so that the components and their dependence relations are clearly understood to make the software debugging and refining process less bumpy.

2.3.5 Maintenance and Modification

After a program has been tested, it can be provided to the customer as the final product. However, the life cycle of software has not ended yet; on the contrary, it just begins its most difficult stage: maintenance and modification. Statistics shows that the maintenance and modification takes on average 80% of the whole software lifetime.

Maintenance on software means correcting errors discovered by users once it is produced. It is not rare some bugs will survive after the final release of a software, no
matter how hard you work in the design, implementation and testing stages. Bugs naturally exist in any software at any time; the only problem is when you catch them. The earlier you catch it, the easier you fix it. If a bug shows up after the program’s release, it could be a huge headache to fix it.

Modification usually happens when the customer desires to change some requirements or to add some new features into the software. Actually, this modification can happen at anytime during the software development. Software designers and developers should be prepared for it all the time.

2.3.6 The whole cycle

As shown in Figure 2.1, the whole software lifetime does not purely go on a sequential order. Instead, at any point in the stages, people might need to step back to make changes. For examples, the customers might change their mind any time, software developers might find some unreasonable designs when implementing the software, and surely the testing stage requires many corrections on the early design and implementation. Although there are many possibilities of turning back to the early stage, the rule of thumb is, the earlier the problem is detected, the less serious its consequences, and the easier it is fixed. That is why people have paid so much time and effort in building good methodologies in system analysis and software architecture design, which are the earliest two stages of software’s lifetime.

2.4 Unified Modeling Language (UML)

The author has concluded in the last section that, without good means for system modeling and architecture design, software could be born with a problematic structure that might bring huge troubles for later-on software development, maintenance, and modification. People have been aware of such problems and trying to build up a standard modeling language for complex system analysis and software structure design. The most fruitful and well-known result in this area is Unified Modeling Language, UML.

2.4.1 A brief history of UML

Object-oriented modeling languages became popular in the mid 1970s and late 1980s, when methodologists used object-oriented programming in increasingly complex
software application development and realized that, to efficiently apply the object-oriented programming concept, a standard modeling language for system analysis and structure design was urgently needed. During that time period, many modeling languages were created, and after some chaotic time of so-called “method wars”, new generation of such methods appeared, with a few clearly prominent ones emerging, most notably Booch, Jacobson’s OOSE (Object-Oriented Software Engineering), and Rumbaugh’s OMT (Object Modeling Technique). Those three modeling languages were all independent complete methods, but each of them had different strength and weakness. The authors of those three modeling methods began to adopt each other’s idea, and finally agreed to create a unified modeling language combining their ideas together, aiming to provide a more standardized, more stable, and more unified methodology to object-oriented marketplace. From 1994 to 1997, many people contributed to the UML project, and, and the first version of UML was born in 1997. The latest version of UML 2.0 has been close to completion.

**2.4.2 An overview of UML**

UML is a standard language for drawing the software blueprint. In a more detailed description, the Unified Modeling Language is a general-purpose visual modeling language that is used to specify, visualize, construct, and document the components of a software system, (Rumbaugh, J., Jacobson, I., and Booch, G., 2001).

![Figure 2.2 Development software procedures](image)

**Figure 2.2 Development software procedures**

In Figure 2.2, the author illustrates the role of UML in a programming procedure. In order to develop software fulfilling any real goal, we need to first simplify the reality into a model, which is actually a blueprint of the real system. A good model reflects a good understanding of the system, specifies the system essentials at a proper abstraction level, constructs a reasonable system architecture, and therefore provides a solid foundation for software implementation and maintenance. UML, the most famous
modeling language, aims to help software developers build such a good model before software implementation.

### 2.4.3 UML notations in a nutshell

In order to master UML, people should first understand UML notations. UML models a system by visualizing system compositions and structures into diagrams. There are many different types of diagrams in UML, including object diagram describing system static compositions, statechart diagram modeling the state changes of an object, use case diagram and activity diagram specifying the dynamic behavior of the system objects, sequential diagram describing the important procedural activities. In RMG, the author used mostly the object diagram for describing the static components and their relationships. The two most important compositions in object diagram are classifier and relationships.

#### 2.4.3.1 Classifiers

A classifier is a discrete concept in the model, having identity, state, behavior, and relationships. There are about ten types of classifiers in UML, and the most important ones among them are class and interface. The author will only introduce those two types of classifiers, and for the details on other classifiers, readers may refer to the UML reference manual. (Booch, et al., 1999)

**Class**

In UML, a class represents a discrete concept within the application being modeled, and it is generally represented by a rectangle with three compartments for the class name, class attributes, and class methods. Figure 2.3 shows the notation of a UML class.
Interface

In UML, an interface is the description of the behavior of objects without giving their implementation, and therefore, the interface only has names and abstract methods declared. In many UML tools, to distinguish an interface from a class, the key word “interface” is displayed at the bottom of the interface block. Figure 2.4 shows the notation of a UML interface.

Figure 2.4 Notation of interface

2.4.3.2 Relationships

Relationships represent the relations between UML classifiers. The important relationships are generalization, association, dependency, and usage. The author will focus on introducing two of them, generalization and association. For others, people may refer to the UML manual book. (Booch, et al., 1999)

Generalization

A generalization describes the relation between a more general object and a more specific object, which has been used for the inheritance relation between two classes, so people usually call it inheritance relation, too. The notation of inheritance is a straight
line with a triangle at the head of the line, pointing to the superclass. In Figure 2.5, the author shows an example of generalization.

![Diagram of Generalization relationship]

**Figure 2.5 Generalization relationship**

**Association**

An association describes that two objects are connected. The notation of an association is a line, and an example is shown in Figure 2.6.

An association might have information on name, multiplicity, and direction. In the example, the name of the association is “contain”, meaning that a room object contains door objects. The arrow of the association line describes the direction of the association, and of course, an undirected association is allowed to represent a symmetric association. People can also provide supplementary information on the multiplicity at each end of the association line. For examples, the numbers, 1 and 1..*, on the room-door association in Figure 2.6, indicates that one room should have at least one door, but multiple doors are allowed.

![Diagram of Association relationship]

**Figure 2.6 Association relationship**

There are two subtypes of associations, aggregation and composition. An aggregation represents a part-whole relation, and its notation is a hollow-diamond
adornment at the “whole” end of the aggregation line. A composition is a stronger
relation than aggregation, and in such a relation, the composite has the full responsibility
to manage its parts, such as their creation and destruction. To distinguish from
aggregation, people use a filled-diamond adornment on the composite end to represent a
composition relation. The aggregation and composition notations are shown in Figure
2.7.

```
Class 1 ──────── Class 2
```

**Figure 2.7 Aggregation and composition notation**

In Figure 2.8, the author shows an example of composition relation. In this
eexample, we need to build a rectangle object with its 4 corner points as the compositions.
When the Rectangle is created, the four corner points should be initialized. When that
rectangle is destroyed, the points are no longer needed, and they should be destroyed as
well.

```
Points 4 ──────── Rectangle
```

**Figure 2.8 Aggregation and composition notation**

So far, the author has briefly introduced the most useful concepts and notations in
UML, and they are just a small part of the whole language. To understand more about
the language, reader should go through the UML users guide or reference manual written
by the creators of UML (Booch, et al., 1999).

### 2.4.4 UML Tool

UML is a modeling language, where the fundamental elements of a language are
defined. However, to use UML in everyday programming, people need some convenient
tool to draw those diagrams, to input the class and interface definition, and to generate
their programs. Just like people usually use MS Visual C++ or other integrated
development environment, IDE, to write C++ programs, and people use JBuilder or
VisualCafe to develop their Java programs, people should have some tool to apply UML
in their system design and development.
The most famous UML tool is Rational Rose from Rational Software Corporation. In this work, the author used a similar tool called Rhapsody, from Ilogix Technology. The main functions of Rhapsody include drawing all types of diagrams, defining components, implementing programs in C++ and Java, and compiling and testing software. In the whole process of developing RMG, the author used Rhapsody as the architecture design and software implementation tool. In this thesis, the author will also use Rhapsody diagrams to introduce the structures and the implementation of RMG.

2.5 Conclusions and Discussions

In this chapter, the author gave a brief introduction to some important topics in software engineering, object-oriented technology and unified modeling language. The author began with review of the programming methodologies used in the history of software development, and focused on clarifying some fundamental concepts, such as class, object, inheritance, etc, in object-oriented programming. After that, the author discussed large software development, difficulties, pitfalls, and the possible solution. Software life cycle was discussed in detail to illustrate the importance of system analysis, modeling, and architecture design. Finally, the author introduced the most important modeling language used nowadays in software system analysis and architecture design, UML.

In conclusion, object-oriented technology is a new programming method that is very useful in developing large-scale, reusable, and extendable software. UML, a unified modeling language, a powerful system modeling tool, plays also an irreplaceably important role in software design and development. Combining those two important methodologies and using their corresponding tools, the author successfully designed and developed RMG software for reaction mechanism generation.
Chapter 3  Fundamental Data Structures

Data structure has long been a key subject in computer science. Studying and understanding data structure and its corresponding algorithms are necessary for any-level programmers in software design, analysis and implementation.

Many people misunderstand a data structure as just a collection of numbers; actually, the study of data structure focus more on the relations between its data elements than on the data themselves. A data structure is a systematic way of organizing and accessing data, that is, each type of data structure reflects one special data management method for some specific application purposes. For example, Queue and Stack are both widely used data structures, and Queue manages data according to first-in-first-out rule that can be used to model a waiting line at a check-out counter of a retail store, while Stack follows first-in-last-out rule that can be used to model the “undo” mechanism in a text editor.

Typical data structures include Linked List, Stack, Queue, Set, Tree, Graph, Hash Table, etc. The representations of those data structure as well as the standard implementation of data objects and their algorithms have been studied carefully. In most object-oriented languages, many of those data structures have been already standardized and implemented in their application libraries. For example, J2SE, LinkedList, Stack, Queue, HashSet, and HashMap are provided as implemented classes, and programmers can directly reuse those objects, or they can extend them to form a new data structure class to fulfill their special programming goal.

Besides those implemented data structures in Java, there are two important data objects, graph and tree, widely used in RMG: Graphs to represent the chemical structures, and Trees to help construct the thermal and kinetics databases. Since Graph and Tree have not been standardized in Java, the author had to design and develop her own Graph and Tree classes.

In this chapter, the author introduces fundamentals of Graph and Tree data structure and discusses the design and implementation details of Graph and Tree in RMG package ChemUtil.
3.1 Graph

Graph is an important data structure widely used in modeling transportation systems, communication systems, project management, chemical species structure, etc. Graph theory has been developed into an individual mathematical topic where the data structure design and the practical algorithms of graphs have been studied by many researchers [Ahuja, 1993; Foulds, 1992; Gondran, 1984; Harary, 1970].

3.1.1 Graph definition

A graph \( G = (V, E) \) consists of a set \( V \) of nodes (vertices) and a set \( E \) of pairs of vertices of \( V \), called arcs (edges). The number of nodes (vertices) and arcs (edges) is denoted by \( |V| \) and \( |E| \), respectively. For a normally connected graph, \( |E| \) could be varied from \( O(|V|) \) to \( O(|V|^2) \). If \( |E| \) is about \( O(|V|) \), the graph is called sparse graph; if \( |E| \) reaches \( O(|V|^2) \), the graph is called dense graph, which is heavily connected.

According to the feature of arc’s direction, graphs can be grouped into directed graphs and undirected graphs. The arcs whose elements are ordered pairs of distinct nodes are called directed arcs, while the arcs whose elements are unordered pairs of distinct nodes are called undirected arcs. If all the arcs in a graph are undirected arcs, we call that graph an undirected graph; otherwise, if there exists at least one directed arc in a graph, that graph is called a directed graph. Figure 1 shows a directed graph and an undirected graph.

![Figure 3.1 Directed graph and undirected graph: (a) directed graph; (b) undirected graph](image)

Directed graphs are mostly used in network flow problems where the directions of flows are important. In this work of modeling chemical structures, since arcs represents chemical bonds that are considered without directions, chemical graphs are all modeled
by undirected graphs. Therefore, from now on, the term of “graph” will refer to undirected graphs.

### 3.1.2 Graph representation

There are a variety of ways to represent a graph. [Drozdek, 2000; Ahuja, Magnanti, and Orlin, 1993]. The three well-known representations are given by adjacency list, adjacency matrix, and incidence matrix.

#### 3.1.2.1 Adjacency list representation

An adjacency list specifies all the nodes adjacent to each node of a graph. For example, the undirected graph shown in Figure 1 (b) can be represented by this list:

<table>
<thead>
<tr>
<th>Node</th>
<th>Adjacent node list</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1, 3, 4</td>
</tr>
<tr>
<td>3</td>
<td>2, 4</td>
</tr>
<tr>
<td>4</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 3.2. Adjacency list representation of undirected graph in figure 1 (b)

The memory cost of storing an adjacency list of a graph is $O(|V|+|E|)$. Considering that the number of arcs ranges between $|V|$ and $|V|^2$, we can omit the $|V|$ term. Therefore, the storing cost for Adjacency list is $O(|E|)$.

#### 3.1.2.2 Adjacency matrix representation

An adjacency matrix, also called node-node adjacency matrix, represents the graph by a $|V| \times |V|$ node matrix, in which the rows and columns both represent the nodes of a graph. The element $a_{ij}$ in this adjacency matrix adopts value of 1, if there is an arc from node $i$ to node $j$, or 0, otherwise. For example, the undirected graph shown in Figure 1 (b) can be represented by adjacency matrix as followed:
Notice that, the adjacency matrix of an undirected graph is symmetric.

The memory cost of storing an adjacency matrix of a graph is obviously $O(|V|^2)$. The extra cost of this representation method, comparing to the adjacency list method, comes from storing the zeros, i.e., unconnected information of a graph.

3.1.2.3 Incidence matrix representation

An incidence matrix, also called node-arc incidence matrix, represent a graph by a $|V| \times |E|$ node-arc matrix, in which the rows represent the nodes and the columns represent the arcs. The element $a_{ij}$ in the incidence matrix adopts value of 1, if arc $j$ is incident with node $i$, or 0, otherwise. For example, the undirected graph shown in Figure 1 (b) can be represented by incidence matrix as followed:

The memory cost of storing an incidence matrix of a graph is $O(|V| \times |E|)$. Obviously, it is the most expensive one among the three representations.

3.1.2.4 Comparison of three representations

After introducing those three representations, can we determine which one is the best? It depends on the problems we need to solve. If the node-node relation is the most
crucial problem in a graph model and such relation has to be reviewed many times, the adjacency list and/or adjacency matrix can be both good choices; if the node-arc relation is quite important, the incidence matrix will be more proper.

Also, another concern is the memory cost for storing graphs if there will be a lot of graph instances in the system. Table 3.1 lists the comparison of storing costs of these three representations. From the table, we can see that, generally speaking, adjacency list is the best for saving quite a lot storing space than the other two representations. Adjacency matrix is the second best, and the incidence matrix is the worst. Notice that for graphs with different complex level of connectivity, those three methods perform differently. For sparse graph, obviously adjacency list is the best one needing only a linear storage space of $O(|V|)$, while the other two representations both requires $O(|V|^2)$ size of space. This is because matrix representations waste too many spaces storing zeros. For dense graphs, there is no big difference between adjacency list and adjacency matrix, either of which is one order of magnitude better than incidence matrix.

<table>
<thead>
<tr>
<th>Method</th>
<th>Graph</th>
<th>Sparse Graph</th>
<th>Dense Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adjacency List</td>
<td>$O(</td>
<td>E</td>
<td>)$</td>
</tr>
<tr>
<td>Adjacency Matrix</td>
<td>$O(</td>
<td>V</td>
<td>^2)$</td>
</tr>
<tr>
<td>Incidence Matrix</td>
<td>$O(</td>
<td>V</td>
<td>\times</td>
</tr>
</tbody>
</table>

Table 3.1. Comparison of the storage cost of three graph representations

People can, therefore, choose different representation method according to their special modeling requirements and the system memory limits.

3.1.3 Graph methods

Beside representations, graph methods, also called graph operations, are very important to abstract the connectivity features of a graph and to make correct graph mutation.

3.1.3.1 General graph methods overview

As any other data structure, graph must firstly own a group of necessary methods revealing fundamental information, like the size of the graph, the elements stored in a node/arc, which two nodes are connected by a given arc, etc. Another type of graph
methods is to implement graph mutation, for example, insertion/deletion of a node, insertion/deletion of an arc between two known nodes,

Table 3.2 shows a list of basic operations and mutation operations. The running time is based on an adjacency list representation.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeNumber()</td>
<td>Return the total number of nodes in a graph</td>
<td>O(1)</td>
</tr>
<tr>
<td>ArcNumber()</td>
<td>Return the total number of arcs in a graph</td>
<td>O(1)</td>
</tr>
<tr>
<td>NodeList()</td>
<td>Return an enumeration of nodes</td>
<td>O(</td>
</tr>
<tr>
<td>ArcList()</td>
<td>Return an enumeration of arcs</td>
<td>O(</td>
</tr>
<tr>
<td>Degree(n)</td>
<td>Return the number of arcs incident to node n</td>
<td>O(1)</td>
</tr>
<tr>
<td>IncidentArcs(n)</td>
<td>Return an enumeration of arcs incident to node n</td>
<td>O(Degree(n))</td>
</tr>
<tr>
<td>AdjacentNode(n)</td>
<td>Return an enumeration of adjacent nodes to node n</td>
<td>O(Degree(n))</td>
</tr>
<tr>
<td>EndNodes(a)</td>
<td>Return the two nodes connected by arc a</td>
<td>O(1)</td>
</tr>
<tr>
<td>OppositeNode(a,n)</td>
<td>Return the end node of arc a distinct from node n</td>
<td>O(1)</td>
</tr>
<tr>
<td>InsertNode(n)</td>
<td>Insert a new node n into graph</td>
<td>O(1)</td>
</tr>
<tr>
<td>InsertArc(a)</td>
<td>Insert a new arc a into graph</td>
<td>O(1)</td>
</tr>
<tr>
<td>InsertArc(a, n1, n2)</td>
<td>Insert a new arc a between existing node n1 and n2</td>
<td>O(1)</td>
</tr>
<tr>
<td>RemoveNode(n)</td>
<td>Remove an existing node n</td>
<td>O(Degree(n))</td>
</tr>
<tr>
<td>RemoveArc(a)</td>
<td>Remove an existing arc a</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

Table 3.2 Some necessary graph methods (based on information in Goodrich and Tamassia, 1998)

Beside the standard graph methods shown in this table, there are three types of species graph operations, traversals, equivalence test, and sub-graph matching, especially important in RMG. In the following text, the author will discuss them in detail.

3.1.3.2 Graph Traversals

A traversal is a systematic procedure for exploring a graph by examining all of its nodes and arcs. (Goodrich and Tamassia, 1998) Two famous algorithms widely used in graph traversal are Depth-First Search, DFS, and Breadth-First Search, BFS.

Just as shown in the name, the fundamental idea behind DFS method is to firstly travel all the way to the deepest unvisited node and return back to the upper nodes level by level, while BFS method is to travel firstly every unvisited node linked at the same level before move onto the next level. Because of this feature, DFS is considered more
“adventurous” than BFS. The pseudo-code for DFS and BFS methods have been provided in many data structure textbooks, readers interested in further details on these two algorithms might check those books. (Goodrich and Tamassia, 1998; Drozdek, 2000; Weiss, 1998; Ahuja, Magnanti, and Orlin, 1993)

Since most complicated graph operation procedures are based on graph traversal, DFS and BFS act more like a fundamental frame for developing advanced graph operations. Based on DFS, people can develop advanced graph operations for finding paths between two nodes, determining if a graph is connected, determining if a graph is cyclic or not, computing a spanning tree, etc. BFS, as well, provides a basis for finding spanning tree, computing the shortest path between any two nodes in a graph, etc.

In RMG, DFS and BFS methods also serve as the basic algorithms for two important advanced graph operations, comparing the equivalence and/or sub-graph relationship between two graphs.

3.2 Tree

Like a graph, a tree is a widely used fundamental data structure in computer science, too. A tree is actually a special type of acyclic graph, which maintains hierarchy relations between its nodes. The most famous example of tree application is the file storage mechanism in all the operating systems. Another important usage of trees is to quickly store and sort data, and generally a special type of binary search tree is used for this purpose.

3.2.1 Tree definition

A tree T is a set of nodes storing elements in a parent-children relationship with the following properties:

1. T has a distinguished node r, called root, that has no parent;
2. Each node other than the root has a parent node;
3. The root is an ancestor of any other node in tree T.

From the definition, we can tell that the difference between tree and other data structures is that a tree maintains a hierarchical relation among all its elements.
Other than parent, children, and root, there are some other useful notations defined for tree elements we are going to use many times. Siblings indicate the nodes that are children from the same parent. Nodes without any children are called “leaves”. The sub-tree rooted at a tree node n in Tree T is the tree consisting of all the descendents of node n in the original tree T.

### 3.2.2 Tree methods

Similar to graph, tree, as common-used data structure, has many methods and operations. Shown in Table 3.3 is a summary of standard tree methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Running Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Root()</td>
<td>Return the root of a tree</td>
<td>O(1)</td>
</tr>
<tr>
<td>Parent(n)</td>
<td>Return the parent of node n</td>
<td>O(1)</td>
</tr>
<tr>
<td>Degree(n)</td>
<td>Return number of children of node n</td>
<td>O(1)</td>
</tr>
<tr>
<td>Children(n)</td>
<td>Return an enumeration of nodes</td>
<td>O(Degree(n))</td>
</tr>
<tr>
<td>Depth(n)</td>
<td>Return the depth from node n to root</td>
<td>O(D[E])</td>
</tr>
<tr>
<td>Height()</td>
<td>Return the largest depth in a tree</td>
<td>O(1)</td>
</tr>
<tr>
<td>InsertChild(n1, n2)</td>
<td>Insert a new child node n2 to its parent node n1</td>
<td>O(1)</td>
</tr>
<tr>
<td>RemoveChild(n1, n2)</td>
<td>Remove a child node n2 from its parent node n1</td>
<td>O(Degree(n1))</td>
</tr>
<tr>
<td>Swap(n1, n2)</td>
<td>Swap the element stored in tree node n1 and n2</td>
<td>O(1)</td>
</tr>
</tbody>
</table>

**Table 3.3 Some necessary tree methods (based on information in Groodrich and Tamassia, 1998)**

There are surely more useful tree operations than the ones shown in Table 3.3, such as tree traversals that are very similar to graph traversal algorithms, sorting algorithm in binary tree, etc., but they are all based on those basic tree methods.

Again, the details of standard implementation of a tree object and its corresponding methods can be found in data structure textbooks. [Goodrich and Tamassia, 1998; Drozdek, 2000; Weiss, 1998; Ahuja, Magnanti, and Orlin, 1993]
3.3 Design chemUtil package in RMG

3.3.1 Graph

Figure 3.5 shows the design UML diagram for Graph in RMG ChemUtil package. It consists of four main components, GraphComponent, Node, Arc, and Graph, as well as their relations.

![UML Diagram for Graph Design]

3.3.1.1 GraphComponent class

Graphs consist of two important components, Node and Arc, which share some similarities. For example, they both are storage containers that hold user-defined node or arc elements, and they both have connecting neighbors. Therefore, not like in other
Graph designs that Node and Arc are designed as totally separated objects, the author introduced a GraphComponent class to account for the similarity of Node and Arc.

GraphComponent is a super class of Node and Arc, and it defines their common attributes and operations. The main GraphComponent attributes include element that represents the object stored in this graph component, visited status that records if this graph component has been visited when doing graph traversal, in-cycle status that indicates if this graph component is a part of a ring, and a collection of neighbors that stores all the graph components directly connecting to this graph component. The main methods in GraphComponent include, access operation for element object, and handling operation for neighbors.

3.3.1.2 Node class and Arc class

Node and Arc classes then inherit existing GraphComponent class and add other special attributes and methods for their specialties. For example, Node class has two new attributes, \( \text{ID} \) representing the unique identification index of each node, and \( \text{centralID} \) accounting for graph centers, which are essential for later defining functional groups centers and chemical reacting sites.

3.3.1.3 Graph class

As discussed earlier, there are three representations for a graph. In order to choose the proper representation, the author considered two facts of RMG: (1) RMG might have a large number of Graph instances to represent distinguished chemical species and functional groups in nature, so that the memory cost of storing a Graph instance could be a crucial problem; and (2) all chemical species and functional group Graphs are sparse graphs with \( O(|E|) - O(|V|) \), since the bond number of such a graph is governed by the atom valence. Table 3.1 has shown that for a sparse graph, the most space-saving representation is the adjacency list.

Two important attributes of Graph class are a node collection and an arc collection, storing all the nodes and arcs included in a graph respectively. Graph class has also the attributes of a cycle list and a centralNode list. The cycle list stores all distinguished cycles in the graph, which is generated from a Graph method
identifyCycle() when this graph instance is created. The centralNode collection stores all
the nodes with a centralID greater than zero, which indicates that node is the center of
this graph. Therefore centralNode collection represents a group of the centers nodes.

As to the important methods in Graph class, the author not only implemented all
the typical Graph methods described in section 3.1.3, but also enhanced Graph
functionality by adding some structure comparison methods for detecting the equivalence
and sub-graph relationship between two graphs.

**Graph Equivalence Test**

In chemical structure modeling, graph equivalence means that two graphs should
be identified equivalent if they represent the same chemical structure, no matter how the
nodes and arcs of the graphs are numbered or ordered, and no matter how graphs are
represented and stored. This is essential test for any chemical application program,
without which people are not able to maintain the uniqueness of a chemical structure.

![Chemical structures](image)

**Figure 3.6 Equivalent graphs for acetic acid**

Figure 3.6 gives an example illustrating the equivalence of two graph
representations of acetic acid. At the first glance, people might be easily fooled to think
those two graphs are different. After careful examination, people might realize those two
graphs are the same. This is just a very simple case. Imagine a complex graph with
hundreds of atoms and bonds; the equivalence test will not be straightforward. For
computers to recognize a chemical species, they store a list or a matrix of all graph nodes
that are numbered for identification. Therefore, if two equivalent graphs are numbered
differently, computers cannot identify the equivalence by simply comparing the
equivalence of the graph representation list or matrix, and an algorithm for the structural
equivalence test is needed.
In order to test the equivalence of two graphs, the author firstly defined the equivalence of two graph components, that is the two graph components are equivalent if and only if such restrictions are satisfied:

(1) The elements that two graph components stored are equivalent. In chemical species case, this restriction requires both graph components store the same type of bonds or atoms;

(2) The neighbor sets that two graph components have are equivalent. Since the neighbor set of any graph component stores also graph components, comparing two neighbor sets then result in comparing, in a combinatorial way, the neighbor graph components’ equivalence, which again requires checking the two restrictions for the two neighbor graph components.

From these two restrictions, we can easily tell this algorithm is a recursive method, and it is similar to DFS algorithm in the way that the algorithm does graph traversal. The pseudo code of this equivalence test algorithm is shown below:

```java
boolean isEquivalent(GraphComponent gc1, GraphComponent gc2)
    if (!gc1’s content isEquivalent gc2’s content) return false;
    if (gc1’s neighbor size != gc2’s neighbor size) return false;
    for each neighbor n2 of gc2
        found = false;
        if (n2 is not visited)
            for each neighbor n1 of gc1
                if (n1 is not visited AND isEquivalent(n2, n1))
                    found = true;
                    break;
            if (!found) return false;
        else
            if (n2’s match is not in the neighbor of gc1) return false;

    set gc1 and gc2 visited matches;
    return true;
```

Based on testing graph component equivalence, the graph equivalence test is to check if there exists a pair of equivalent graph components from two graphs.

This equivalence test algorithm has been applied to many complicated chemical species, and for example the cyclic structures as following:
The testing results for complex chemical structures are all correct, showing that this algorithm works correctly.

**Sub-graph matching**

In many graph-related modeling procedures, the sub-graph relation of two graphs is generally not that crucial, and the discussion of such algorithm is rarely found. On the other hand, since different systems have quite different definitions of the sub-graph relations, this issue can be hardly generalized into a standard graph method, and people always talk about this topic for each special case.

In RMG, however, the sub-graph algorithm, which enables the quick and precise categorization of all chemical species according to the functionality of their local structure, is indispensable to implement the new data model concept for RMG described in the last chapter. Furthermore, since it is the local structure that mainly defines the roles and behaviors when species participate in reactions, this algorithm further enables a better and fast search for thermal properties and reaction kinetics parameters.

RMG sub-graph matching algorithm is quite similar to the graph equivalent test. The only difference is that sub-graph matching algorithm is testing the sub-graph relation instead of equivalence relation between two graphs.

Sub-graph relation has many different meanings in different fields. For chemical structure graphs in RMG, “A is a sub-graph of B” means that the group of chemical structures represented by A is a subset of the group of chemical structures represented by B. Figure 3.7 illustrates such relation between acetic acid and a general acid functional group. Graph (a) represents an acetic acid, and graph (b) represents a group of acid; therefore, graph (a) is a subtype of graph (b), and we can say graph (a) is a sub-graph of graph (b).
Replacing the equivalence relation in graph component equivalence algorithm by a subset relation, we can get a sub-graph matching algorithm for two graph components. The pseudo code of this algorithm is shown as following:

```java
boolean isSub(GraphComponent gc2) {
    if (gc1's content isSub of gc2's content) return false;
    for each neighbor n2 of gc2
        found = false;
        if (n2 is not visited)
            for each neighbor n1 of gc1
                if (n1 is not visited AND isSub(n2, n1))
                    found = true;
                    break;
            if (!found) return false;
        else
            if (n2's match is not in the neighbor of gc1) return false;
    set gc1 and gc2 visited matches;
    return true;
}
```

Similar to the graph equivalent test, the sub-graph matching algorithm is then to check if there exists a pair of sub-graph-relation graph components from two graphs.

### 3.3.2 Tree

Figure 3.8 shows the design UML diagram for Tree in RMG ChemUtil package. This diagram mainly consists of two parts, Tree and HierarchyTree.
3.3.2.1 Tree

Implementing Tree data structure is quite straightforward, and RMG follows a standard way introduced in many data structure textbooks for developing TreeNode and Tree class.

TreeNode class models any node in a Tree. It has an attribute, element, to store the data object. Every tree node also has a reference to its father node and a set of children nodes, so that it can easily access to its upper and lower level tree nodes, except that tree root doesn’t have father.

Tree class represents a general-meaning tree data structure. It has an attribute of root, which is an instance of TreeNode class. It also implements important tree methods described in Table 3.3.

Figure 3.8 UML diagram for Graph Design
### 3.3.2.2 HierarchyTree

HierarchyTree models the hierarchy tree used to store functional group objects, and later we will use it to help construct thermal group library and reaction kinetics rate rule library.

HierarchyTreeNode class represents any node in a HierarchyTree. It is the subclass of TreeNode class, and inherits all the attribute and methods from TreeNode. Different from TreeNode, HierarchyTreeNode only stores FunctionalGroup or a collection of FunctionalGroups as its element object. Besides the methods inheriting from TreeNode, HierarchyTreeNode provides two important searching functions, findMatchedLeaf() and findMatchedPath(). Recall that there is a subtree rooted at any intermediate tree node. Method findMatchedLeaf() is to search through the subtree rooted at current HierarchyTreeNode to find and return the Leaf storing the best-matched FunctionalGroup for the passed-in ChemGraph and FunctionalGroup. Method findMatchedPath() will return the whole searching path instead of only one Leaf.

HierarchyTree class is a subclass of Tree class. In addition, it also provides two more functions, findMatchedLeaf() and findMatchedPath() for searching the entire hierarchy tree.

### 3.4 Conclusions and discussions

In this chapter, the author briefly introduced two fundamental data structures, graph and tree, as well as their definitions, representations, major functionalities, and possible applications. The author also described in detail how Graph and Tree and their related classes were implemented in RMG. Besides implementing the fundamental functionalities for Graph and Tree, the author also developed some supplementary methods for comparing graph equivalence and sub-graph relations and for searching through hierarchy tree, which will be later applied to implement some important functions of species and reaction class, such as identifying uniqueness of chemical structure, searching thermal group and kinetics rate rule library, and so on.
Chapter 4 Fundamental Chemical Objects in RMG

In this chapter, the author introduces the design and development of important fundamental chemical objects, ChemGraph, Species, and FunctionalGroup, and FunctionalGroupCollection. This part of work can be seen as teaching computers Chemistry 101, from which computers learn to know basic chemical elements like atom and bond, to recognize more complex chemical molecules and functional groups, and to compute the fundamental properties for them, so that, more importantly, an unambiguous communication channel between computers and chemists can be built. The RMG package chem covers this part of work.

4.1 ChemGraph and Species

People have long been using Graph to represent chemical structures [Trinajstic et al., 1991; Trinajstic, 1983a and 1983b; Balaban, 1976], and many theories and practical algorithms have been provided from their research. In RMG, we also applied Graph theory in modeling our chemical structures. There are two important classes in RMG to model chemical species, ChemGraph and Species.

4.1.1 ChemGraph

ChemGraph is a class extended from Graph class to uniquely represent any distinct chemical structure existing in nature. The main extensions from Graph to ChemGraph are in two aspects: firstly, the elements stored in a ChemGraph should be limited to chemical units, like atom, radical, and bond; secondly, methods for calculating chemical properties should be added.

4.1.1.1 ChemGraph Elements

RMG defines totally three types of ChemGraph elements: chemical element, free electron, both of which are stored in each ChemGraph node, and chemical bond, which is stored in each ChemGraph arc.

Chemical Element defines the type of atom at each Graph node site. Theoretically, all elements in the periodic table could be employed. However, since the
chemical species in the reaction systems RMG studies mostly consist of carbon, oxygen, and hydrogen, only those chemical elements are currently included in RMG. Users can surely enrich the chemical element set if necessary.

Free electron indicates the number and type of free radical at each ChemGraph node. In RMG, there are five types of free electron: non-radical, mono-radical, bi-radical, singlet, triplet, and tri-radical. Note that singlet and triplet are two kinds of bi-radicals, which act differently in some special type of reactions. Therefore, we distinguish singlet and triplet as needed. Combining the chemical element and free electron information, we can define any type of local atom in a chemical species.

Chemical bond specifies the bond types linking two atoms. RMG bond includes single bond, double bond, double-cis bond, double-trans bond, triple bond, and benzene bond. The author included two sub-types of double bonds: double-cis bond and double-trans bond. The Graph data structure used in RMG is just an undirected connectivity graph without storing relative position of each ChemGraph node; therefore, automatically determining whether a double bond is cis or trans is so far impossible in RMG. If users need to accommodate such information, they have to explicitly specify it.

Table 4.1 summarizes all ChemGraph elements in RMG.
In a nutshell, with the elements in Table 4.1, RMG is able to define any chemical molecule and radical in C/H/O system. If users are interested in more complicated chemical systems, more chemical elements like nitrogen, sulfur, silicon, etc., might be included. Present RMG doesn’t model ions, either. If people are willing to include ion objects into RMG, they can simply add a new type of ChemGraph elements, charge, which indicated the positive or negative electron charge(s) at each atom site.

4.1.1.2 Chemical Adjacency List representation

With the definition of ChemGraph elements, we can apply adjacency list representation for uniquely specifying a chemical structure. The adjacency list representation shown here is slightly different from the one introduced in section 3.1.2.1 for general Graphs, since more information about free electrons and chemical bonds needs to be included in ChemGraph. An example of acetyloxy radical and its adjacency list representations are shown in Figure 4.1.
Figure 4.1 an instance of ChemGraph: (a) acetyloxy radical; (b) Adjacency list representation

Noticing that it wastes users much time to input all the hydrogen atoms, the author provides another simplified adjacency list format, in which users don’t have to specify any hydrogen atom. The example for acetyloxy radical is shown in Figure 4.2. Once RMG reads in such representation, it automatically fills all the unsaturated atoms with hydrogen. Note that such simplification is only valid for ChemGraph; in FunctionalGroup representation, all of the atom definitions must be explicitly specified.

![Diagram of the acetyloxy radical](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>Atom</th>
<th>Radical</th>
<th>Adjacency List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C</td>
<td>0</td>
<td>{2,S} {5,S} {6,S} {7,S}</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>0</td>
<td>{1,S} {3,D} {4,S}</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>0</td>
<td>{2,D}</td>
</tr>
<tr>
<td>4</td>
<td>O</td>
<td>1</td>
<td>{2,S}</td>
</tr>
<tr>
<td>5</td>
<td>H</td>
<td>0</td>
<td>{1,S}</td>
</tr>
<tr>
<td>6</td>
<td>H</td>
<td>0</td>
<td>{1,S}</td>
</tr>
<tr>
<td>7</td>
<td>H</td>
<td>0</td>
<td>{1,S}</td>
</tr>
</tbody>
</table>

Figure 4.2 simplified adjacency list for Acetyloxy radical

4.1.1.3 Important ChemGraph properties

Besides the structure of ChemGraph, there are some other important chemical properties, like chemical formula, molecular weight, symmetry number, and thermal properties, such as enthalpy, entropy, free energy, and heat capacity.

To compute chemical formula, RMG simply counts the number of different type of atoms in a ChemGraph and generates a string with the format of CxHyOz(.). The dots indicate the total radical number of a ChemGraph. It is also straightforward to calculate a ChemGraph's molecular weight from its chemical formula.
In other model generation software, people developed method to compute a unique string name of a chemical structure, and then use it as the unique representation of that chemical structure. It has been shown that this method works well for acyclic structures and simple cyclic structures, but for complex rings, sometimes it fails. In RMG, we didn’t use unique string representations to distinguish chemical structures; instead, every time we compare two ChemGraph objects, we use the equivalence test method introduced in the last chapter.

To estimate thermal properties, such as enthalpy, entropy, free energy, and heat capacity, the Group Additivity method as proposed by Benson is being used in RMG. The details will be covered in the next chapter.

Calculating the symmetry number is essential to estimate the entropy of any chemical species, and this algorithm will also be discussed together with all other thermal properties in the next chapter.

### 4.1.2 Species

Species is a class designed to wrap all the possible resonance isomers for a chemical species.

#### 4.1.2.1 Resonance

Resonance is an important phenomenon in organic chemistry. According to Morrison and Boyd, resonance is described as following: “Whenever a molecule can be represented by two or more structures that differ only in the arrangement of electrons— that is, by structures that have the same arrangement of atomic nuclei—there is resonance. The molecule is a hybrid of all these structures, and cannot be represented satisfactorily by any one of them. Each of these structures is said to contribute to the hybrid.”

An example of resonance is shown in Figure 4.3. In this butenyl radical, there are two possible resonance isomer structures, shown as (a) and (b), contributing to the hybrid (c). This doesn’t imply that (a) and (b) have any existence in nature. The free electron is actually delocalized, i.e., it is not locked at any local carbon site, instead; it is distributed over three carbon sites.
4.1.2.2 Species design

To model this resonance phenomenon in RMG, the author decided to generate and store all the possible resonance isomer structures instead of the hybrid structure for any chemical species. Although the most reasonable way to model a hybrid is to use its uniform structure like shown in Figure 4.2(c), it is not feasible to store it because of the unclear way of modeling the special bond with a delocalized electron.

In RMG, ChemGraph class models any unique chemical structure, so each resonance isomer of a hybrid is modeled as an individual ChemGraph. Species class is then to collect all the resonance isomers’ ChemGraph. Besides the collection, Species class also has another important attribute storing the most thermally-stable resonance isomer, which is the one that has the lowest heat of formation at 298K among all the isomers. In Species class, there is also an important method for generating all of a hybrid’s possible resonance isomers from any one of them, so that all the structures of a hybrid will be recognized and stored at the same time.

4.2 FunctionalGroup

Besides ChemGraph, the author together with Dr. S. Raman introduced another important chemical structure, FunctionalGroup, into RMG to represent a common structural pattern shared by a group of chemical species all possessing.

As discussed in section 2.2, FunctionalGroups in RMG serve as standard patterns for categorizing chemical species into groups with different reacting functionalities so that reaction families can be easily defined. Functional Groups also act as keys when RMG searches the thermal and kinetics databases.

This section discusses the details on defining functional groups in RMG.
4.2.1 Difference between ChemGraph and FunctionalGroup

The differences between ChemGraph and FunctionalGroup lie in three aspects. Figure 4.4 shows a comparison between a ChemGraph and a FunctionalGroup, which illustrates such differences.

![ChemGraph and FunctionalGroup](image)

(a) ChemGraph (b) FunctionalGroup (note that slash acts as separator)

First, while ChemGraph only allows one choice of atom/electron/bond at each ChemGraph node and arc site, FunctionalGroup allows multiple choices of chemical elements at its node and arc site. For example, in the FunctionalGroup shown in Figure 4.4(b), at 3\textsuperscript{rd} node, both carbon and oxygen are allowed; at the arc between 1\textsuperscript{st} node and 2\textsuperscript{nd} node, both double and triple bonds are allowed. In ChemGraph, such flexibility is against the fundamental chemical principle, and is, therefore, not allowed.

Secondly, while ChemGraph specifies all the valences of any atom, FunctionalGroup allows atoms with unspecified valence. For example, at each node in Figure 4.4(b), there might be undefined valence(s), while in the ChemGraph, there is no such flexibility permitted.

Thirdly, FunctionalGroup should have central node(s)/arc(s) defined; otherwise, there is no reasonable explanation for any Functional Group. For ChemGraph, it is not necessary to define centers, although readers will see the usage of ChemGraph centers when reacting ChemGraph, which will be discussed in chapter 7. In the FunctionalGroup shown in Figure 4.4(b), *1 and *2 indicates that the emphasis of this FunctionalGroup is at the part of the multiple (double or triple) bond between the 1\textsuperscript{st} and 2\textsuperscript{nd} carbons. Therefore, we can imagine that such FunctionalGroup might define a group of chemicals participating in some special types of reactions requiring multiple bonds, such as radical addition to multiple bond, cyclo-addition, etc. If the centers are moved to some other sites, for example, *1 and *2 centers are moved to 2\textsuperscript{nd} and 3\textsuperscript{rd} node in the same
FunctionalGroup, the chemical meaning of it will be totally changed. The centers become a single bond between two carbons or between carbon and oxygen. That FunctionalGroup may accordingly summarize the chemicals participating in bond dissociation reactions.

### 4.2.2 FunctionalGroup elements

Similar to extending ChemGraph from Graph, the elements stored in FunctionalGroup should be firstly specified.

Two issues have to be addressed when defining FunctionalGroup elements. First, FunctionalGroups elements should include all the ChemGraph elements, so that people have a consistent way drawing ChemGraph and FunctionalGroup. Accordingly, all the symbols defined in Table 4.1 are also valid elements for defining FunctionalGroup. Secondly, we should add more chemical elements into FunctionalGroup to efficiently and flexibly define more complicated FunctionalGroup. The major improvement is in the part of chemical elements, and the definitions for free electron and chemical bond remain unchanged.

Table 4.2 shows the additional chemical elements for FunctionalGroup. To build this chemical element set, we followed several principles. First of all, the extra FunctionalGroup elements should account for common chemical sense, i.e., they should be consistent with chemistry terminology people use every day. For example, R and R!H are basically used to represent any atom and any non-hydrogen atom, respectively. Secondly, the extra FunctionalGroup element should provide more atom connectivity information. The new FunctionalGroup elements not only define the type of the atom, but also specify an atom’s nearest bonds, sometimes also some of its nearest atoms. Finally, FunctionalGroup elements for the same type of atom should be disjoint. For example, for carbon type, there are seven types of FunctionalGroup elements, Cs, Cd, CO, Cdd, Ct, Cb, and Cbf. Union of those seven elements fully covers all the possible types of carbon in a C/H/O system; furthermore, each of those seven elements will be disjoint to each other. As a result, any carbon atom in nature will fall in only one of those seven categories. The design simplifies and speeds up the definition of FunctionalGroup quite well.
<table>
<thead>
<tr>
<th>Notation</th>
<th>ChemGraph</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cs</td>
<td><img src="image" alt="Carbon bonded to four single bonds" /></td>
<td>Carbon bonded to four single bonds</td>
</tr>
<tr>
<td>Cd</td>
<td><img src="image" alt="Carbon bonded to a double bond and two single bonds" /></td>
<td>Carbon bonded to a double bond and two single bonds (The other end of the double bond should be carbon.)</td>
</tr>
<tr>
<td>Cdd</td>
<td><img src="image" alt="Carbon bonded to two double bonds" /></td>
<td>Carbon bonded to two double bonds</td>
</tr>
<tr>
<td>Ct</td>
<td><img src="image" alt="Carbon bonded to a triple bond and a single bond" /></td>
<td>Carbon bonded to a triple bond and a single bond</td>
</tr>
<tr>
<td>Cb</td>
<td><img src="image" alt="Carbon bonded to two benzene bonds and a single bond (carbon belonging to only one benzene ring)" /></td>
<td>Carbon bonded to two benzene bonds and a single bond (carbon belonging to only one benzene ring)</td>
</tr>
<tr>
<td>Cbf</td>
<td><img src="image" alt="Carbon bonded to three benzene bonds (carbon belonging to two or three benzene rings)" /></td>
<td>Carbon bonded to three benzene bonds (carbon belonging to two or three benzene rings)</td>
</tr>
<tr>
<td>CO</td>
<td><img src="image" alt="Carbon bonded to a double bond and two single bonds (The other end of the double bond should be oxygen.)" /></td>
<td>Carbon bonded to a double bond and two single bonds (The other end of the double bond should be oxygen.)</td>
</tr>
<tr>
<td>Os</td>
<td><img src="image" alt="Oxygen bonded to two single bonds" /></td>
<td>Oxygen bonded to two single bonds</td>
</tr>
<tr>
<td>Od</td>
<td><img src="image" alt="Oxygen bonded to a double bond" /></td>
<td>Oxygen bonded to a double bond</td>
</tr>
<tr>
<td>Oa</td>
<td><img src="image" alt="Oxygen triplet" /></td>
<td>Oxygen triplet</td>
</tr>
<tr>
<td>R</td>
<td><img src="image" alt="Any atom" /></td>
<td>Any atom</td>
</tr>
<tr>
<td>R!H</td>
<td><img src="image" alt="Any non-Hydrogen atom" /></td>
<td>Any non-Hydrogen atom</td>
</tr>
</tbody>
</table>

**Table 4.2 Summarization of additional chemical elements for FunctionalGroup**

With adding more FunctionalGroup elements, we can save much time and storage space in defining FunctionalGroup. For example, Figure 4.5 shows two equivalent FunctionalGroups. Without the extra FunctionalGroups elements, we have to stay with representation in (a), which require defining nine nodes and ten arcs. If the Cs and Cb definitions are available, only two nodes and one arc needed to be created to define the same FunctionalGroup, as shown in (b).
Another benefit of using the additional FunctionalGroup elements is that it makes it possible to define more complicated FunctionalGroup in a compact way. Figure 4.6 shows such an example. In this complicated alkene FunctionalGroup, *1st carbon is required to bond with a hydrogen and a delocalized carbon, and *2nd carbon is required to bond to two nondelocalized carbons/oxygens. If we don’t have the definitions of Cs, Cd, Cb, Ct, CO, etc, to unambiguously represent such FunctionalGroup in a single graph is impossible.

In summary, defining a set of additional elements provide users an efficient and flexible mean to specify complicated FunctionalGroup.

4.2.3 FunctionalGroup adjacency list representation

Similar to ChemGraph, RMG uses a standard format of adjacency list for FunctionalGroup. The adjacency list representation for the FunctionalGroup in Figure 4.6 is given in Figure 4.7.

FunctionalGroup adjacency list is slightly different from the ChemGraph’s. There is one extra column in FunctionalGroup adjacency list, CentralID, recording the
centers of the FunctionalGroup. Also, in the atom and bond sites of FunctionalGroup, multiple choices are permitted, such as the 4\textsuperscript{th}, 5\textsuperscript{th}, and 6\textsuperscript{th} atoms in Figure 4.7.

![Chemical Structure](image)

<table>
<thead>
<tr>
<th>ID</th>
<th>CentralID</th>
<th>Atom</th>
<th>Radical</th>
<th>Adjacency List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*1</td>
<td>Cd</td>
<td>0</td>
<td>{2,S} {5,S} {6,S} {7,S}</td>
</tr>
<tr>
<td>2</td>
<td>*2</td>
<td>Cd</td>
<td>0</td>
<td>{1,S} {3,D} {4,S}</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>H</td>
<td>0</td>
<td>{2,D}</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>{Cd, Ct, Cb, CO}</td>
<td>0</td>
<td>{2,S}</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>{Cs, Os}</td>
<td>0</td>
<td>{1,S}</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>{Cs, Os}</td>
<td>0</td>
<td>{1,S}</td>
</tr>
</tbody>
</table>

Figure 4.7 A complicated alkene FunctionalGroup

With the adjacency list representation, people can also uniquely represent a group of chemicals with the similar structures.

4.3 Design of chem Package

The package of chem is responsible for implementing the fundamental chemical structures, ChemGraph, Species, FunctionalGroup, and FunctionalGroupCollection introduced earlier in this chapter.

This package includes two major object diagrams, ChemicalComponents diagram, which defines the basic components, like atom, bond, free electron, etc., and ChemGraphAndFunctionalGroup diagram which defines the ChemGraph, Species, and FunctionalGroup

4.3.1 ChemicalComponents object diagram

Figure 4.8 shows the ChemComponents object diagram in RMG. It defines the elements that can be stored in a chemical graph’s node and/or arc sites, Atom, FGAtom, and Bond.
4.3.1.1 Atom

Atom class represents a chemical atom. Atom has two attributes, a ChemElement object and a FreeElectron object. ChemElement class defines chemical element, like carbon, hydrogen, oxygen, etc. The collection of total chemical elements included in RMG is represented in ChemElementDictionary, and since RMG currently models pyrolysis and oxidation system, this dictionary only defines carbon, hydrogen, and oxygen, as shown in Table 4.1. If users are interested in adding other chemical elements, they can extend this ChemElementDictionary. FreeElectron class defines unpaired electron(s) sitting on one atom. In RMG, we allowed mono-radical, 1-centered bi-radical, and 1-centered tri-radical, as shown in Table 4.1, and those allowed free electron are declared in FreeElectronDictionary class. Atom class also will provide users some informative methods to reveal the atom’s name, type, weight, valence, etc.

4.3.1.2 FGAtom

FGAtom class models a FunctionalGroup atom. In another word, FGAtom instances are the elements stored in FunctionalGroup graphs nodes. FGAtom also has two attributes, a FGElement object and a FreeElectron object. The FGElement class models 12 types of functional group elements defined in Table 4.2. Similarly, those 12 functional group elements are collection by FGElementDictionary, and users later might extend this dictionary to enlarge the functional group element definition.
Figure 4.8 ChemicalComponents object diagram in RMG
4.3.1.3 ChemNodeElement

ChemNodeElement is an interface that Atom and FGAtom class are required to implement, and it represents a higher-level abstraction on the elements allowed to be stored in node positions in a chemical structures, ChemGraph and FunctionalGroup. ChemNodeElement declares a series of methods that Atom and FGAtom must implement.

4.3.1.4 Bond

Bond class models the chemical bond. Bond class has attributes, name, type, order, etc., and corresponding informative methods revealing those attributes settings. BondDictionary class, again, collects all the bond types allowed in RMG system, as shown in Table 4.1, and can be extended if necessary.

4.3.2 ChemGraphAndFunctionalGroup object diagram

Figure 4.9 shows ChemGraphAndFunctionalGroup object diagram. It defines main chemical structures in RMG, ChemGraph, Species, FunctionalGroup, and FunctionalGroupCollection.

4.3.2.1 ChemGraph and Species

ChemGraph class represents any real chemical structure. The major attribute of ChemGraph class is a graph that stores Atom instances in its nodes and Bond instances in its arcs. Other attributes of ChemGraph include name, chemical formula, thermodynamic properties, symmetry number, etc. There are also many ChemGraph methods to compute the chemical properties, like chemical formula, molecular weight, symmetry number, and thermal properties like enthalpy, entropy, free energy, and heat capacity. In addition, ChemGraph also has methods to examine the chemical restrictions on connectivity and valence. Thermal properties estimation will be provided by group additivity method, which will be introduced in detail in next chapter.
Figure 4.9 ChemGraphAndFunctionalGroup object diagram in RMG
Species class is designed to account for resonance isomers. If a chemical species doesn't have resonance structures, Species class will store only one ChemGraph object, and it behaves the same as a ChemGraph. If a chemical species has resonance isomers, when creating a Species instance, RMG generates and stores all the resonance structures into that Species. When participating in a reaction, any resonance isomer in a Species object is able to react with the other chemical structure.

SpeciesDictionary class is a dictionary collecting all the Species instances appearing in RMG. That is, any time RMG creates a new Species, it will put that new instance into SpeciesDictionary using its chemical structures as keys. Every time a new Species is about to be created, SpeciesDictionary will look through its collections to check if it is already created; if it is, SpeciesDictionary will return the old instance; otherwise, a new Species instance is allowed to be created. This is a typical technique to control the creation of a class's instances.

4.3.2.2 FunctionalGroup and FunctionalGroupCollection

FunctionalGroup class models any functional group structure. It also has a graph, which stores Atom instance, FGAtom instance or a collection of them into its graph’s node sites, and stores Bond instance or a collection of them into its graph’s bond sites. Notice that we allow the collection of atoms or bonds in FunctionalGroup graph to implement the FunctionalGroup flexibility. FunctionalGroup class also has a string name to indicate its identity.

FunctionalGroupCollection class represents a union of FunctionalGroups. It is used in some circumstances wherein a union of several FunctionalGroups represents another meaningful functional group whose graph can’t be easily drawn.

4.3.2.3 Matchable

Matchable interface is a higher-level abstraction interface of ChemGraph, FunctionalGroup, and FunctionalGroup collection. It requires that all those classes implement a sub-graph matching algorithm so that later the sub-graph relationship between instances of any two of them can be tested.
4.4 Conclusions and Discussions

In this chapter, the author discussed the design and implementation issues of main chemical structures in RMG, ChemGraph, Species, and FunctionalGroup. The author firstly introduced the definition, representations, and major functions, of those objects, and then described in detail how they were designed and implemented in RMG.

Modeling chemical species by graphs is well studied by many researchers, but modeling functional groups as graph objects has not been widely used and appreciated. In RMG, we proposed to use the similar graph theory principles in abstracting and modeling functional groups. Readers will realize the benefit of it later when we discuss issues that arise when generating reactions and estimating thermal and kinetics properties.
Chapter 5  Group Additivity  Thermodynamic Properties Estimator

Properly estimating the thermodynamic properties of chemical species is critical for reaction system modeling. The thermodynamic properties discussed here include enthalpy (H), heat capacity (C_p), entropy (S), and Gibbs free energy (G).

5.1 Roles of thermodynamic properties in reaction system

According to second law of thermodynamics, every closed and isolated system approaches an “equilibrium” state, which is governed by the thermodynamic properties of the system. Specific to the chemical reaction system, such equilibrium means that a reacting procedure will end with a steady state where compositions of all chemical species remain unchanged with respect to time. It is the thermodynamic properties of all species involved that determine such steady state of a reacting system.

Besides the steady state, thermodynamic properties also partly affect the reaction kinetics. For any reversible chemical reaction,

\[ A + B \xrightleftharpoons{k_f}{k_b} C + D \]  (5.1)

there holds a relations between forward reaction kinetics and backward reaction kinetics:

\[ K_{eq} = \frac{k_f}{k_b} \]  (5.2)

The \( K_{eq} \) is called the equilibrium constant, and it is related to the free energy change of reaction (5.1) by the equation:

\[ \frac{k_f}{k_b} = K_{eq} = (RT)^{\Delta n} \exp(-\Delta G_{\text{reac}} / RT) \]  (5.3)

Combining equation (5.2) and (5.3), one concludes that, for any reversible chemical reaction, the reaction kinetics of only one direction is independent, and the kinetics of the backward reaction will be determined together by the forward kinetics and the thermodynamic properties of the reaction. This is so-called thermodynamic consistency of a reaction system.
In RMG, in order to maintain the thermal consistency, we define the kinetics library for all the forward reactions independently, and derive the backward reaction kinetics from the reaction free energy change and forward kinetics. In addition, some reaction activation energies are estimated using formulas that depend on $\Delta H_{\text{rxn}}$, the most famous example is the Evans-Polanyi relation: $E = \alpha \cdot \Delta H_{\text{rxn}} + E_0$. Therefore, estimation of the thermodynamic properties is also needed while calculating many kinetic parameters.

5.2 Overview of thermodynamic properties

5.2.1 Species thermodynamic properties

In order to estimate important thermodynamic properties for chemical species, let us first have a brief review on some important thermodynamic definitions.

Heat capacity, $C_p(T)$, is the derivative of heat of formation with respect to temperature at constant pressure:

$$C_p \equiv \left( \frac{\partial H}{\partial T} \right)_p$$  \hspace{1cm} (5.4)

From (5.4), the formula for calculating standard heat of formation, $\Delta H^0(T)$, of a chemical species can be derived as:

$$\Delta H^0(T) = \Delta H^0(T_0) + \int_{T_0}^{T} C_p(T) dT$$  \hspace{1cm} (5.5)

where $\Delta H^0(T_0)$ is the standard heat of formation at temperature $T_0$, and $C_p(T)$ represents the heat capacity curve in the temperature range from $T_0$ to $T$.

Standard entropy change, $\Delta S^0(T)$, of a chemical species at temperature $T$ can be calculated as:

$$\Delta S^0(T) = S^0(T_0) + \int_{T_0}^{T} \frac{C_p(T)}{T} dT$$  \hspace{1cm} (5.6)
where \( S^0(T_0) \) is the standard entropy change at temperature \( T_0 \), and \( C_p(T) \) represents the heat capacity curve in the temperature range from \( T_0 \) to \( T \).

Combining the result from (5.5) and (5.6), we can easily calculate standard Gibbs free energy, \( \Delta G^0(T) \) as:

\[
\Delta G^0(T) = \Delta H^0(T) - T \cdot \Delta S^0(T)
\]

(5.7)

With thermal properties of every species involved in a reaction being calculated, the corresponding reaction thermal properties, such as reaction enthalpy change, reaction entropy change, and reaction free energy change, can be calculated by the stoichiometric summation of the thermal properties of its reactant(s) and product(s).

### 5.2.2 Estimation from current thermodynamic data format

The main goal in thermal property estimation is to calculate species enthalpy, entropy, and free energy at different temperatures. To achieve that, the most straightforward way is to identify a set of temperature-independent thermal parameters for each chemical species and to build the relation between this parameter set with the thermal properties. There are different ways to build such thermal data sets for species; some are more theoretical, and some are more empirical. Here, the author introduces two thermodynamic data formats widely used in many applications.

#### 5.2.2.1 Benson’s Format

From equations (5.5)–(5.7), we can see that it is straightforward to store thermal data by recording \( \Delta H^0(T_0), S^0(T_0) \), and a series of \( C_p(T) \) at different temperatures. Those species parameters can be obtained by fitting experimental or quantum chemical results. Generally, people agree to record standard heat of formation and standard entropy at \( T_0 = 298K \), and to sample the heat capacity curve at discretized temperature points to represent the whole heat capacity profile. For example, a widely used data format consists of \( \Delta H^0(298K), S^0(298K), C_p(300K), C_p(400K), C_p(500K), C_p(600K), C_p(800K), C_p(1000K), C_p(1500K) \) [Benson, 1976].
From this thermal data format, the author derived numerical equations by applying integration rules for calculating heat of formation and entropy from (5.5) and (5.6). The detailed steps of derivation are shown in Appendix 5.1, and the results are as following:

\[
\Delta H^0(T) = \Delta H^0(298) + \sum_{i=0}^{5} \Delta H^0(T_i \sim T_{i+1})
\]

where \[
\Delta H^0(T_i \sim T_{i+1}) = \begin{cases} 
0 & T < T_i \\
\frac{a(T^2 - T_i^2) + b(T - T_i)}{2} + \frac{(C_p(T_i) + C_p(T_{i+1})) \cdot (T_{i+1} - T_i)}{2} & T_i \leq T < T_{i+1} \\
\frac{C_p(T_i) \cdot C_p(T_{i+1}) \cdot (T_{i+1} - T_i)}{2} & T_{i+1} \leq T
\end{cases}
\]

\[
\Delta S^0(T) = S^0(298) + \sum_{i=0}^{5} \Delta S^0(T_i \sim T_{i+1})
\]

where \[
\Delta S^0(T_i \sim T_{i+1}) = \begin{cases} 
0 & T < T_i \\
\frac{a(T - T_i) + b \ln \left( \frac{T}{T_i} \right)}{T_i} & T_i \leq T < T_{i+1} \\
\frac{a(T_{i+1} - T_i) + b \ln \left( \frac{T_{i+1}}{T_i} \right)}{T_i} & T_{i+1} \leq T
\end{cases}
\]

In (5.8) and (5.9), \( T_i \) is the discretized temperature point. In this case, \( T_0 = 300K, T_1 = 400K, T_2 = 500K, T_3 = 600K, T_4 = 800K, T_5 = 1000K, T_6 = 1500K \). Parameter \( a \) and \( b \) are the slope and intercept of linear relations between \( C_p \) and \( T \) in the range of \( T_i \) to \( T_{i+1} \), and they can be calculated as following:

\[
a = \frac{C_p(T_{i+1}) - C_p(T_i)}{T_{i+1} - T_i}, \quad b = \frac{C_p(T_i) \cdot T_{i+1} - C_p(T_{i+1}) \cdot T_i}{T_{i+1} - T_i}
\]

This thermal data format has been used in many thermal estimation program based on Benson’s group additivity method, which will be covered in section 5.3, such as THERM [Ritter and Bozzelli, 1991] and CHETAH [Downey et al., 1994].
5.2.2.2 NASA polynomial format

NASA polynomial thermal data format is another very popular thermal data format used as industrial standard. The key difference between this format and the format in 5.2.2.1 is how the heat capacity curve is recorded. In NASA format, $C_p(T)$ curve is fitted into two fourth-order polynomials with 10 parameters in two different temperature ranges:

\[
C_p(T) = R(a_1 + a_2 T + a_3 T^2 + a_4 T^3 + a_5 T^4) \quad T > T_b
\]
\[
C_p(T) = R(a_8 + a_9 T^2 + a_{10} T^3 + a_{11} T^4) \quad T < T_b
\]  

(5.11)

where, $R$ is the gas constant, $T$ is temperature, and $T_b$ is the break point of two temperature ranges.

Plugging (5.11) into formulas for computing enthalpy (5.5) and entropy (5.6), together with four extra parameters, $a_6, a_7, a_{13}, \text{and} a_{14}$, accounting for the standard heat of formation and entropy at lower temperature boundary result in the equations for computing enthalpy and entropy from NASA polynomials:

\[
H^0(T) = R(a_1 T + a_2 T^2 / 2 + a_3 T^3 / 3 + a_4 T^4 / 4 + a_5 T^5 / 5 + a_6) \quad T > T_b
\]
\[
H^0(T) = R(a_8 T + a_9 T^2 / 2 + a_{10} T^3 / 3 + a_{11} T^4 / 4 + a_{12} T^5 / 5 + a_{13}) \quad T < T_b
\]

(5.12)

\[
S^0(T) = R(a_1 \ln T + a_2 T + a_3 T^2 / 2 + a_4 T^3 / 3 + a_5 T^4 / 4 + a_7) \quad T > T_b
\]
\[
S^0(T) = R(a_8 \ln T + a_9 T + a_{10} T^2 / 2 + a_{11} T^3 / 3 + a_{12} T^4 / 4 + a_{14}) \quad T < T_b
\]

(5.13)

Therefore, in NASA polynomial format, there are totally fourteen polynomial coefficients, $a_1 \sim a_{14}$, stored for each chemical species, with which $C_p(T), H^0(T), \text{and} S^0(T)$ can be quickly computed from (5.11)–(5.13). NASA polynomial format is widely used in industrial applications. For example, thermal property input data file in CHEMKIN uses this format. (Users are cautioned that more than one NASA polynomial format is used in the literature.)

5.3 Group additivity method

There are millions of chemical species existing in nature. Generating and tabulating the thermal data for each of them, no matter what format is used, seems a
hopeless task. Although a large amount of experimental and computational effort has been applied into this area, only a small portion of chemical species' thermal properties has been obtained with a satisfactory precision. People desire a thermodynamic property estimation algorithm that can be applied to all kinds of molecules with an acceptable precision and a good running efficiency, and many people had developed practical estimation method for this purpose [Allen, 1959; Thin et al., 1971; Benson, 1976; Yoneda, 1979; Pedley et al., 1986]. Reid et al. [1987] provided a summary for some of the methods. The “Additivity Rules” proposed by Benson and co-workers [Benson, 1976] play an important role in developing such an algorithm, which, in recent several decades, has been widely accepted as an accurate, efficient method to estimate ideal gas phase heat capacities, heats of formations, and entropies for stable species, especially when no experimental or computational data can be found in the literature or widely used databases [Stein and Fahr, 1985; Bohen and Benson, 1993].

5.3.1 Group additivity hypothesis [Benson, 1976]

The group additivity hypothesis defined in Benson’s text states: “most molecular properties of larger molecules can be considered, roughly, as being made up of additive contribution from the individual atoms or bonds in the molecule.” The fundamental reason supporting this principle is the observation of the “short range” of the forces between atoms in species, from which people assume that every individual atom might have a nearly constant contribution in different molecules. If the individual atom’s contribution could be quantified one by one, thermodynamic properties of any chemical species can be estimated by adding up the contributions from all the atoms.

Not only establishing the group additivity theory, Benson and co-workers also provide a set of group data containing carbon, oxygen, nitrogen, sulfur, halogen, silicon, etc., as well as a set of correction data for extended group interaction, for example ring corrections. Meanwhile, many other groups have also provided more and more thermochemical group contribution data, which gradually improved the precision level of group additivity estimation method. The thermochemical data format Benson and other groups adopted is the one introduced in section 5.2.2.1. Therefore, after the species thermodynamic parameter data set has been estimated from the group additivity
algorithm, equations (5.8)–(5.10) can then be used to compute the species thermochemical properties at any temperature.

In Figure 5.1, the author illustrates the details of group additivity algorithm by an example of acetic acid thermal property estimation. According to group additivity principle, every non-terminal atom will constitute an individual group that has unique contribution to the total species thermodynamic properties. Therefore, the first step is to partition the acetic acid into non-terminal atoms. Secondly, the valences attached to each non-terminal atom are identified to form a group. Thirdly all the groups will be compared to Benson's thermochemical groups to obtain a best-matched group value. In this case, the 1st carbon atom is classified as a C-(H)₃(C) Benson group, which means that the central carbon atom connects to three hydrogen and one carbon by four single bonds. The 2nd carbon atom is classified as a CO-(O)(C) group, which means that a CO center, consisting of 2nd carbon double-bonded to the 3rd atom (oxygen), connects to one carbon and one oxygen by two single bonds. The 4th atom (oxygen) is classified as an O-(CO)(H) group, which means that an oxygen center atom connects to a CO group and a hydrogen by two single bonds. After proper thermal groups are detected, the proper values of \( \Delta H^\circ(298K) \), \( S^\circ(298K) \), \( C_p(300K) \), \( C_p(400K) \), \( C_p(500K) \), \( C_p(600K) \), \( C_p(800K) \), \( C_p(1000K) \), \( C_p(1500K) \) will be obtained from Benson's group values library, as tabulated in Figure 5.1. Finally, adding the thermal data for C-(H)₃(CO), CO-(O)(C), and O-(CO)(H) groups gives the thermal data for acetic acid molecule.
5.3.2 Corrections for “global” interactions

Group additivity theory is valid for a large number of species, except for some special types of species, in which not only “local” interaction between species should be considered, but also “global” interactions. Important examples are ring compounds, where the simple group additivity estimation results in some un-negligible deviations. In order to account for it, Benson suggested applying overall corrections after summing the individual atom-centered groups. Such correction accounts for the extra contribution of “non-local” structure.
Benson described several types of global corrections: entropy correction from molecule symmetry number, gauche correction for steric crowding, group correction for ring structures, and correction for cis conformation in alkenes. The author will cover the details of implementing those corrections in RMG in section 5.4.4.

5.3.3 Radical correction: hydrogen atom bond increments (HBI)

In estimating thermodynamic properties of a free radical, Benson and coworkers used similar group additivity as for stable molecules and provided group contribution values for a set of radical groups.

Lay and co-workers [1995] proposed however a different approach, called hydrogen atom bond increments (HBI) method for estimating free radicals thermochemistry. The key idea in this method is to quantify the differences between a free radical and its stable molecule version, which are illustrated by those equations:

\[
\Delta H^0(298K, R\bullet) = D(R \cdots H) + \Delta H^0(298K, RH) - 52.1 \text{kcal/mol} \quad (5.14)
\]

\[
S^0(298K, R\bullet) = S^0(298K, RH) + \Delta S^0(298K, HBI) - R \ln(\sigma_{R\bullet}/\sigma_{RH}) \quad (5.15)
\]

\[
C_p(T, R\bullet) = C_p(T, RH) + C_p(T, HBI) \quad (5.16)
\]

In those equations, \( R\bullet \) represents a free radical, and \( RH \) represents the corresponding stable molecule. Equation (5.14) gives the radical correction to standard enthalpy at 298K. \( D(R \cdots H) \) is the bond energy for the R-H bond being broken to form a free radical. Equation (5.15) shows the radical correction to the standard entropy at 298K. The term of \( R \ln(\sigma_{R\bullet}/\sigma_{RH}) \) accounts for the entropy change due to the changes in symmetry number from a stable molecule to a free radical. Equation (5.16) shows the radical correction to the heat capacities. For any type of radical, if the HBI values of \( D(R \cdots H), \Delta S^0(298K, HBI) \), and \( C_p(T, HBI) \) are available, we can calculate its thermal properties by adding the radical HBI correction terms to its stable molecule’s thermal data, as shown in equation (5.14)–(5.16).

With efforts of several researchers, particularly Bozzelli, an HBI database is now available covering major radicals in hydrocarbon and oxygenated hydrocarbon systems.
5.3.4 Standard group additivity thermal property estimation procedure

Here the author summarizes the standard steps that a group additivity thermal property estimation algorithm must have:

Step 1: partition species into individual atoms;

Step 2: for each non-terminal atom: determine its group type by its nearest neighbors and find the best matched group value in existing thermal group database;

Step 3: sum up the contribution values of all the atoms;

Step 4: make radical correction if necessary

Step 5: make the proper corrections according to the non-local structure the species has.

Before RMG, there are several programs implementing this group additivity algorithm, THERM [Ritter and Bozzelli, 1991], CHETAH [Downey et al., 1994], GAPP [Grenda et al., 1998] (ExxonMobil’s proprietary, automated extension of THERM), and NIST Structure and Properties [Stein, 1994]. In RMG, the author also implemented this algorithm for thermal properties estimation.

5.4 Improvements of group additivity method in RMG

In RMG, for thermochemical properties estimation, the author followed the standard steps summarized in the last section for the group additivity algorithm; furthermore, the author focused on the implementation issues of ensuring chemical flexibility, reusability and extendibility.

5.4.1 Define and identify thermal group

In the group additivity algorithm, the most essential step is to identify the proper thermal group for each individual atom. Since this step happens to every non-hydrogen atom of a species, the efficiency of this identification step is crucial to the whole algorithm’s efficiency. The most straightforward way to handle this problem is to hardcode all the group structure information into the program. The author shows a part of pseudo code implementing such functionality in Figure 5.2.
Figure 5.2 Pseudo code for identifying proper thermal group for an atom

The advantage of this method is that the running efficiency is very good, since the code containing every chemical classification detail is finally compiled into executable machine language. However, there are some quite significant disadvantages to such a coding exercise. First, the chemical categorization of an atom into proper thermal groups has to be fully translated into programming language, and any misunderstanding during this translation might result in big problems. On the other hand, since such code has a huge chunk of nested if/else statements, programming and debugging it might be quite difficult and time-consuming. Finally, if chemists want, after the program has been completed, to modify the present groups or accommodate more groups corresponding to the new discovery, it could be a big headache, or sometimes even a mission impossible, for a programmer to make such modifications and extensions to the existing code, for a programmer to make such modification and extension to the existing code.
Aiming to implement chemical extensibility and flexibility in RMG, we proposed a different way to handle this problem. The key idea is to separate the thermal group structure specification from the coding. To do so, we defined all the thermal group structures as FunctionalGroup objects in a thermal group dictionary file. For example, the three thermal groups used in Figure 5.1, C-(H)₃(C), CO-(O)(C), and O-(CO)(H), can be represented by RMG FunctionalGroup and stored in adjacency list as following:
The values of thermal groups are stored in another library file by name of the thermal group. When RMG launches, it sets up its one-to-one mapping between the thermal group structure and the corresponding group value by reading in that dictionary file and the library file. In estimating the thermal property for any ChemGraph object, the standard ChemGraph vs. FunctionalGroup sub-graph matching algorithm, discussed in chapter 4, is called to detect the best matching thermal FunctionalGroups for each individual atom in that ChemGraph object, and the corresponding group contribution value will be retrieved. We called such procedure as data-model based design. The pseudo code of such procedure is summarized in Figure 5.4.
For (each atom in a ChemGraph) {
    For (each thermal FunctionalGroup in library) {
        If (atom is sub-graph of this FunctionalGroup) {
            Retrieve its thermal group value and add it to sum;
        }
    }
}

Figure 5.4 Pseudo code for identifying proper thermal group in data-model based design

The first advantage of data-model based design is, of course, reducing the programming burden. The logic of pseudo code looks neat and easy to develop and maintain. The complicated big chunk of the classification code in Figure 5.2 is now divided into two types of work: (1) chemists’ job of correctly defining and checking of the thermal FunctionalGroups structure and value in library, and (2) software developer’s job of providing a correct sub-graph matching algorithm for between any ChemGraph and any FunctionalGroup. Another corresponding advantage of such design is obviously to provide chemists a great flexibility in defining and modifying thermal groups and their values. To add or modify any group and its value, the only changes are to add another FunctionalGroup or to modify the existing FunctionalGroup in the thermal structure file and to add or modify its corresponding thermal value into the thermal group value library file. The thermal property estimation code remains unchanged. By separating chemistry and programming details in such way, both chemists and software developers would find themselves focusing more on the work that takes advantage of their own expertise.

Although this new data-model based design provides flexibility and extensibility to users, some disadvantages of it cannot be ignored. The most important one is the CPU time cost. The worst-case running time of graph matching based thermal group detection is $O(N*M)$, where $N$ is the number of non-hydrogen atoms in the species, and $M$ is the number of total thermal groups in the library, while, for the old method shown in Figure 5.1, the running time is only $O(N)$. Currently, from the work of Benson and other groups, the number of thermal groups have reached 150~200, which is still growing. This might result in two order of magnitude slowing down of the part of thermal properties estimation in RMG. How can we deal with this running efficiency problem? A feasible
solution, organizing all the thermal groups into a hierarchical database, can properly handle the problem, and the details of it will be discussed in the next section.

5.4.2 Organize thermal group data by a tree

5.4.2.1 The hierarchy relations among thermal groups

The main reason causing the extra $O(M)$ times CPU cost for the data-model based design arises from the unordered structure of the thermal group database. Searching through an unordered list requires, in the worst case, checking all the elements stored in that unordered list. Unordered lists are generally used to store entirely unrelated data. If there exist some logic relations between data, people tend to make use of this logic to build a better structure for a database. Is there any “chemical logic” among the thermal group structures that we can use to build a better database? The answer is YES.

In order to describe “the chemical logic”, let us take a close look at an example of the thermal group, C-(H)$_3$(O). To specify it, people define firstly the center carbon atom, secondly the four single bonds connecting to the central carbon, finally the atoms at the other ends of those four single bonds. In general, the Benson groups can be abstracted into three levels:

First level: center atom type
Second level: bonds connecting to the center atom
Third level: atom type at the other end of the bonds defined in second level

This is actually a process defining more and more details propagating from the center atom. We can imagine that fourth, fifth, and deeper levels could be added, if people want more details. The more levels deep the specification goes, the more details that specification gets. Such specification can be seen as a typical hierarchical relation, since, when the specification procedure steps into $n^{th}$ level, all the features defined in previous $n-1$ levels are implicitly inherited. A typical data structure to model a hierarchical structure is hierarchy tree. Therefore, in RMG, we proposed to structure the thermal groups database by a hierarchy tree, where each level of the hierarchy tree corresponds to a level of the specification details.
5.4.2.2 Rules to build the thermal group tree

The goal of building such a hierarchy tree is to arrange all the groups into one thermal hierarchy tree, so that the chemical logic between different thermal groups can be illuminated clearly.

![Parent-child hierarchy tree diagram]

Figure 5.5 the parent-children relation represented by a hierarchy tree

Figure 5.5 shows one-level parent-children relation in a tree. In this relation, there exists a parent-child inheritance between any pair of Parent and Child(i), and all the children have the same importance. In RMG, we strengthen the parent-children relation by adding more chemical restrictions to define a chemical generality-specificity relation among the thermal group tree nodes. The four extra chemical restrictions are described as following:

(Rule 1) Every tree node stores a FunctionalGroup or a FunctionalGroup collection as its element. We named this restriction as “FunctionalGroup element rule”.

(Rule 2) The FunctionalGroup stored in the Parent tree node should have a more chemically general meaning than the one stored in any of its children. Specifically, it implies that the sub-graph matching algorithm should detect that the FunctionalGroup stored in any Child(i) tree node is always a sub-graph of the FunctionalGroup stored in Parent tree node. We named this restriction as “generality-specificity rule”.

(Rule 3) The FunctionalGroups stored in two siblings tree node should have chemically disjoint meanings. In another word, any chemical structure being a sub-graph of child(i) node will not be the sub-graph of any same-level node of Child(i). We called this restriction as “disjoint siblings rule”.

(Rule 4) The FunctionalGroup stored in Parent tree node should be fully partitioned into FunctionalGroups in its children. The expression of “full partition”
means that the chemical meaning of the union of all the children FunctionalGroups should be exactly equal to that of the parent FunctionalGroup. This rule is to guarantee that any of the chemical structures matched with the parent FunctionalGroup will find a properly matched child FunctionalGroup. We called this restriction as “full partition rule”.

Why do we need those extra rules? The “FunctionalGroup element rule” is to enable the comparison between any parent and child by the sub-graph matching algorithm, so that the generality-specificity rule can be established and checked. The “generality-specificity rule” is to test the existence of generality-specificity relation between any parent and child. This rule guarantees that the deeper people step down along the tree, the more detailed matched FunctionalGroup they would find. The “disjoint siblings rule” is to remove the possibility that, at the same tree level, one chemical structure is a sub-graph of two or more sibling FunctionalGroups. This rule is the heart of the tree method to enable a fast retrieval through the database. If this rule is satisfied, once the FunctionalGroup stored in one child is detected as a matched FunctionalGroup, all FunctionalGroups represented by other siblings are obviously not matched. Therefore, this rule allows one to track only the sub-tree rooted at the child node storing the matched FunctionalGroup and ignoring all the sub-trees rooted at its sibling nodes; therefore, a great amount of searching time can be saved. The “full partition rule” is to ensure that for any chemical structure matched to the FunctionalGroup in the parent, there must exist a matched child node. In another word, searching through a tree won’t end at any intermediate tree node until a tree leaf is found to match with the chemical structure. The reason we set up this rule is that we don’t want to miss any possible branch(s) of a Parent node. For any tree node, there are only two choices, either being a tree leaf, or being fully partitioned. Implementing this rule is dependent on the definition of the full chemical system. For example, for a C/H/O chemical system, if any atom R is set in a parent node, it can be branched into three children nodes corresponding to carbon, hydrogen, and oxygen, respectively, which satisfies the “full partition rule”. If the chemical system is extended to include nitrogen, one more child for nitrogen atom should be added.
In summary, the main reason setting those four rules is to construct a database that has “chemical hierarchy logic” so that a fast and correct search throughout the database is possible.

5.4.2.3 Building the thermal group tree

Following the four rules just provided, J. Yu, with the help from Dr. S. Raman, constructed the complete thermal group tree for C/H/O chemical system. Because of the limited space, the full 1132-node tree is not completely displayed. A small part of the complete thermal group tree is shown in Figure 5.6 to clarify its structure.

The first level of the tree, so-called tree root, has the most general thermal group: FunctionalGroup centered at a non-hydrogen atom.

The second level branches the tree root into two children, a thermal group centered at carbon atom and a thermal group centered at an oxygen atom. This is corresponding to atom type specification in the thermal group definition.
Figure 5.6 Part of thermal group tree (constructed by J. Yu)

(1) The symbols used in this figure have similar meanings as the FunctionalGroup element defined in Table 4.2. The only difference is the Cds in this graph is the union of FunctionalGroup element of Cd and CO, i.e., Cds defined here doesn't distinguish the atom its double bond connects to.

(2) The atom in red color represents the central atom of a thermal group.

(3) The dashed lines below many tree nodes indicate the existence of a sub-tree below that node. The author doesn't show the complete tree, where there are totally 1132 nodes.
The third level fixes the bonds connecting to the center atom. For carbon atom, since it has a valence of four, the possible bond arrangement could be, four single bonds, Cs, two single bonds and a double bond, Cds, two double bonds, Cdd, one triple bond and one single bond, Ct, two benzene bond and one single bond, Cb, and three benzene bonds, Cbf. Therefore, below the C atom, there are six children defined. It is noteworthy to emphasize the “disjoint sibling rule” and “full partition rule” here. At this level, people can have many different ways to branch carbon atom satisfying those rules, and we simply enumerated all the bond combinatorial possibilities for carbon valence. For oxygen, since it only has a valence of two, there are two possible bond arrangements, two single bonds, Os, and one double bond, Od.

The fourth level defines the atoms at the other ends of the fixed bonds in the third level. At this level, there are many more possibilities. For example, below the Cs at the 3rd level, there are totally 15 children nodes at this level, and it is impossible to list them all here. Shown in the Figure 5.6 are the children from Ct only. In Ct branch, since Ct has a triple bond and a single bond fixed in the third level, we have to define the two atoms at the other end of those two bonds. In our C/H/O system, only two carbons can be connected by a triple bond, so the atom at the other end of the triple bond must be a carbon. Therefore, there is no partition for different atom type at the other end of the triple bond. It is noteworthy that, if the chemical system is extended to nitrogen, more nodes distinguishing Ct = N and Ct = C should be added. For the single bond, three possibilities, H, O, and C result in three children, Ct-H, Ct-C, and Ct-O.

The fifth level provides more information on the other bonds connecting to the nearest atom specified in last level. For example, for the parent tree node, Ct-C, the nearest carbon atom still has undefined valence of three, so there are four possible bond arrangement for a valence of three: three single bonds, Cs, one single bond and a double bond, Cds, one triple bond, Ct, and one benzene bond, Cb. As a result, Ct-C can be branched into Ct-Cs, Ct-Cds, Ct-Ct, and Ct-Cb at this level.

The normal Benson’s thermal group specification generally ends at this level specifying only the nearest-neighbor bond(s)/atom(s) to the center. If we follow the same
logic, we can rearrange all the Benson's thermal groups into our thermal group tree. However, we are not quite satisfied with stopping here since we believe that, following the same rules, we could build a tree structure holding much more detailed specification than Benson's thermal groups. We therefore added more levels into our thermal group tree to allow us to represent more detailed information that was not available when Benson derived his groups 30 years ago. For example, the Ct-Cds node, which is Benson group already, is branched into two more children, Ct-(Cds=C) and Ct-(Cds=O), and Ct-(Cds=C) are branched into more children and grandchildren.

In summary, in building the thermal group tree, the more levels people add to the thermal group tree, the more detailed thermal group structure they have at the tree leaf, and, if the correspondingly precise thermal group value are developed for those thermal groups, the more precise thermal properties will be obtained from this group additivity method. Of course, there is a trade-off between the specification and the efficiency. We don't want to build a huge tree that is too costly to search, although it gives very detailed description of thermal groups structure. Another associated problem is the availability of the proper thermal group value for the thermal FunctionalGroup structure at each tree leaf. If we couldn't get such data with good precision, the usefulness of this tree method would be greatly reduced.

5.4.2.4 Assigning the proper value to thermal group tree node

After the thermal group tree is constructed, J. Yu also built the thermal group value library by assigning proper contribution values to each of the thermal FunctionalGroups. As we have stated, the current thermal group tree includes totally 1132 tree nodes, which are many more than the thermal group values that are available in the literature. So far, the available data include about 100 thermal group values centered at carbon and oxygen atoms from Benson [1976] and Stein and Fahr [1985], and 18 group values containing ketene group from Sumathi and Green [2002b and 2002c]. In the thermal group library, all those literature thermal values are assigned to the proper thermal FunctionalGroup, and the source of the data is documented in details.

Except those groups whose group values can be found from literature and, therefore, assigned directly, all the other thermal groups have to be assigned an
approximate value. The assignment process used in RMG is based on the similarity of the thermal group structures, proposed by J. Yu. The similarity assumption suggests that two thermal FunctionalGroups with similar structures will have close contribution value to the thermal property, and, if the value of one thermal group is missing, we could use that of the other thermal group to approximate the missing value. Now a new question arises: how to define the structural similarity? We can get a hint from the hierarchy tree structure that properly describes the chemical consanguinity between any two FunctionalGroups. For example, two sibling FunctionalGroups might be considered having “similar” structures. Sometimes, if the value from a sibling is not available, the value from a cousin might be applicable. However, for any thermal FunctionalGroup, there might be more than one sibling and one cousin, and picking the right one with the best similarity needs some input from chemist expert knowledge. Table 5.1 displays eight substitution rules for checking the structural similarity between two thermal FunctionalGroups.

<table>
<thead>
<tr>
<th></th>
<th>Real structure</th>
<th>Substituted structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(Cds=Cdd=Cd)</td>
<td>(Cds=Cds)</td>
</tr>
<tr>
<td>2</td>
<td>(Cdd=Cd)</td>
<td>Cds</td>
</tr>
<tr>
<td>3</td>
<td>Cb</td>
<td>(Cds=Cds)</td>
</tr>
<tr>
<td>4</td>
<td>Ct</td>
<td>(Cds=Cds)</td>
</tr>
<tr>
<td>5</td>
<td>(Cds=Cds)</td>
<td>Cs</td>
</tr>
<tr>
<td>6</td>
<td>(Cds=Od)</td>
<td>Cs</td>
</tr>
<tr>
<td>7</td>
<td>(Cds=Cdd=Od)</td>
<td>(Cds=Od)</td>
</tr>
<tr>
<td>8</td>
<td>Os</td>
<td>Cs</td>
</tr>
</tbody>
</table>

Table 5.1 Substitution rule in the group value assignment (from J. Yu)

In Table 5.1, column one shows the real structure in the thermal FunctionalGroup without a literature group value, and column two shows the substituted structure that might be used to replace the real part in the original FunctionalGroup. It is noteworthy that, in rule six, Cs (and not Cds) is chosen to substitute (Cds=Od) because from an analysis of the available Benson’s group values, groups in which Cs substitutes (Cds=Od) have values closer to the original group than groups in which Cds is the substitute.
The similarity substitution rules are sequentially applied following the order in Table 5.1. As an example, the thermal group Cs-(Cds=Od)(Cds=Cdd=Cd)CbH does not have a literature value. According to 1st rule in Table 5.1, (Cds=Cdd=Cd) is replaced by (Cds=Cds), resulting in another group Cs-(Cds=Od)(Cds=Cds)CbH, which in turn also has no known value. The 3rd rule, substituting (Cb) by (Cds=Cds), is then applied to the new FunctionalGroup Cs-(Cds=Od)(Cds=Cds)CbH, giving another simplified FunctionalGroup Cs-(Cds=Od)(Cds=Cds)(Cds=Cds)H, which, again, does not have a value. Finally the 5th rule, substituting (Cds=Cds) by (Cs), forms a FunctionalGroup Cs-(Cds=Od)(Cs)(Cs)H, which has a known thermal value. In a word, such similar group substitution procedure will repeat recursively until a literature value is found.

It is worthy to emphasize that, in constructing thermal group library, we stored the name of the similar FunctionalGroup detected for a missing-value FunctionalGroup at every substitution step, not the final discovered thermal value. For example, for Cs-(Cds=Od)(Cds=Cdd=Cd)CbH, we store the name of Cs-(Cds=Od)(Cds=Cds)CbH, not the final thermal value of Cs-(Cds=Od)(Cs)(Cs)H. When RMG encounters a thermal group without literature value, the program will be redirect to its similar FunctionalGroup. Such process will be repeated until the real value is found, i.e., the program searches along a tracking path that is exactly the same as that of the substitution assignment process. For example, such tracking path for the thermal group of Cs-(Cds=Od)(Cds=Cdd=Cd)CbH is:

Cs-(Cds=Od)(Cds=Cdd=Cd)CbH
-> Cs-(Cds=Od)(Cds=Cds)CbH
-> Cs-(Cds=Od)(Cds=Cds)(Cds=Cds)H
-> Cs-(Cds=Od)(Cs)(Cs)H
-> real thermal value

The advantage of such handling is that, if any of the real thermal values needs modifying, or if any of the thermal values that used to be missing needs adding into the library, users only need to modify or add the right value at one position in the library, not every position corresponding to all the thermal groups depending on that thermal value.
5.4.2.5 Advantages of using a tree structure

The advantages of using a tree structure for thermal group database can be summarized into three aspects.

Firstly, unlike the original Benson's thermal groups tabulated in an unordered list without clarifying the chemical relations between groups, the hierarchy tree provides a systematic way to model the chemical structural logic and to visualize hierarchical relations between thermal groups. The tree structure describes such relations in a hierarchical way, which localized the specification details step by step. In another word, it only adds a little more details to the children from the parent. People do not have to understand the whole picture at one time, which is what they have to do with the old tabulated list for Benson's groups; instead, for the tree-structured thermal group database, people are able to more focus on and understand the local details described at only one tree level each time, and to build a complete picture gradually and easily.

Secondly, searching through a tree-structured database is much more efficient than searching through a list-structured database. If there are N total thermal groups in the database, the searching time is O(logM) for the tree-structured database and O(M) for the list-structured database. This is a significant increase for the searching efficiency. At the end of section 5.4.1, we claimed that the data-model based group additivity algorithm in RMG added an extra burden of searching through the thermal FunctionalGroup database, resulting in an O(N*M) running time for group additivity estimation instead of the original O(N) running time, where N is the total number of non-hydrogen atoms in chemical species, and M is the total number of thermal FunctionalGroups in the database. The extra O(M) running time is from the O(M) searching time through an unordered list. By constructing a hierarchy-structured thermal FunctionalGroup database, we have successfully reduced database-searching time to O(logM). Therefore, the group additivity algorithm in RMG now has a running time O(NlogM), which is quite close to O(N), unless M is skyrocketing, which is not possibly happening to our thermal group database. In another word, the data-model based group additivity algorithm enables the flexibility and extensibility of the group additivity algorithm, and constructing the tree-structured thermal group database makes that algorithm run efficiently.
Thirdly, using the tree-structured database not only enables the searching efficiency improvement, but also provides a straightforward way to modify and extend the database. In section 5.4.1, we have stated that the data-model based group additivity method provides the extensibility, since adding any new thermal group can be easily implemented by inserting another FunctionalGroup structure and its value into the database. After we have built the thermal database in a hierarchy structure, this extending procedure becomes more systematic and error-free. If the database is in an unordered list structure, when people add a new thermal group, they should carefully investigate if the new thermal group structure has some overlap or conflict with the groups already existing in the database, since the overlapping or conflicting definition of thermal groups will probably cause undetermined, wrong results. However, checking 1000 thermal groups in an unordered list is not that easy. If we have the hierarchy tree structured database, the work of extending the database will be localized to the sub-tree associated with the new FunctionalGroup, and all other groups are absolutely irrelevant because of the “disjoint siblings” rule for building the tree.

### 5.4.3 HBI radical correction

In Section 5.3.3, we have introduced the hydrogen atom bond increments, HBI, method for estimating free radicals. Equation (5.14)–(5.16) provide relations between the thermal properties of a free radical and those of its stable molecule. As discussed in section 5.3.3, to compute the thermal data for a free radical, we need two sets of thermal data: one is the thermal data of the stable molecule of that radical, the other is the HBI thermal data for the radical correction. The stable molecule’s thermal data can be calculated by the group additivity method, and the HBI thermal data can be found in literature \cite{Lay1995, Sumathi2003, Chen1995}.

#### 5.4.3.1 Building HBI radical tree

Similar to the normal thermal group tree, we also constructed HBI radical structure tree to organize the HBI radical correction data. The rules and methodology used in constructing HBI radical correction data are the same as those used in normal thermal groups. The HBI tree is partly shown in Figure 5.7.
Figure 5.7 Part of HBI radical tree (constructed by J. Yu)

(1) Dot represents the free electron. The atom and dot(s) in red color represent the center of a HBI group.

(2) Mono-radical, bi-radical, and tri-radical in C/H/O system are completely covered by this HBI radical tree.

(3) The symbols used in this figure have similar meanings as the FunctionalGroup element defined in Table 4.2. The only difference is the Cds in this graph is the union of FunctionalGroup element of Cd and CO, i.e., Cds defined here doesn't distinguish the atom its double bond connects to.

(4) The dashed lines below many tree nodes indicate the existence of a sub-tree below that node. The author doesn't show the complete tree, where there are totally 76 nodes.
In this HBI radical tree, the first level is a general radical structure. The second level distinguishes the mono-radical, bi-radical, and tri-radical. We include all the possible radical types in our tree to make it fully complete to describe any free radical, although the HBI correction data for some of the bi-radicals and tri-radicals are not available. The third level defines the center atom at the radical center. The fourth level defined the bond arrangement around center atom. Similar steps to construct more levels are applied until a satisfactory specification has been achieved.

The HBI radical structure tree, with totally 76 tree nodes, is much simpler than the thermal group tree, which has 1132 nodes. This is because of two reasons. First, since the radical has possessed one or more of the valences of central carbon or oxygen, there are many fewer bond arrangement possibilities at the central atom. Secondly, since people do not have as much understanding of the radical thermochemistry as for stable molecules, HBI data are available only for quite simple radical groups.

5.4.3.2 Building HBI radical library

After the HBI radical structure tree has been built, we have to assign the proper thermal values gathered from literature for HBI groups. Lay et al [1995] calculated HBI values for the main hydrocarbons groups, while Sumathi and Green [2003] and Chen and Bozzelli [1995] developed HBI values for oxygenated hydrocarbons. Sumathi and Green [2003] provide a more comprehensive set of HBI values for radicals centered at a carbon atom, and thus their values were used, whereas Chen and Bozzelli’s [1995] values for radicals centered at oxygen were used. For bi-radical groups, a differentiation was made between triplet and singlet states. Since in most cases the triplet state is the most stable, when a bi-radical is formed and no criterion is given to determine whether it is a singlet or a triplet, it is assumed to be in the triplet state. For tri-radical group, there is no literature HBI group value for it, J. Yu estimated the value by comparing methylidyene in relation to methane from NIST values.
5.4.4 Corrections

There are three types of thermal corrections in RMG, symmetry number correction to the entropy, ring correction, and cis conformation correction.

5.4.4.1 Symmetry number correction

*Corrections from symmetry number* $\sigma$

Molecule symmetry number, $\sigma$, is defined as the total number of independent permutations of identical atoms (or groups) in a molecule that can be arrived at by simple rotations of the molecule (without breaking any chemical bonds). The total symmetry of a molecule arises from two different contributions: the internal symmetry and the external symmetry. The internal symmetry corresponds to the number of permutations of identical atoms arrived at through internal rotations, whereas the external symmetry takes into account the rotation of the whole molecule. Statistically, entropy is the measurement of the distinguishable configurations that a molecule can have, and the existence of a symmetry number greater than 1 means the decrease of a molecule’s distinguishable configurations, which results in the reduction of the molecule’s calculated entropy.

The group additivity method only takes into account the contribution of every atom with its local nearest neighbors. As a global feature, the overall symmetry of the molecule has to be included as an additional correction as following:

$$ S^0(T) = S^\sigma(T, GA) - R \ln \sigma $$  \hspace{1cm} (5.17)

Where $S^\sigma(T, GA)$ is the standard entropy at 298K calculated from group additivity method, $\sigma$ is the symmetry number, R is the gas constant, and $S^0(T)$ is the corrected standard entropy at 298K.

*Calculation of symmetry number* $\sigma$

To calculate the symmetry number in RMG, J. Yu and author developed an algorithm to determine the symmetry of acyclic molecules, assuming that the total symmetry number have the contributions from the rotations around each atom, each bond and each axis of the molecule, and the mathematics expression is shown in equation (5.18).
\[ \sigma = \prod_i \sigma_{\text{atom},i} \cdot \prod_j \sigma_{\text{bond},j} \cdot \prod_k \sigma_{\text{axis},k} \] (5.18)

Where \( \sigma_{\text{atom},i} \) represents the symmetry number around the \( i \)th atom, \( \sigma_{\text{bond},j} \) represents the symmetry number around the \( j \)th bond, and \( \sigma_{\text{axis},k} \) represents the symmetry number around the \( k \)th molecule axis.

Table 5.2 ~ Table 5.4 shows the rules determine \( \sigma_{\text{atom},i} \), \( \sigma_{\text{bond},j} \), and \( \sigma_{\text{axis},k} \) respectively. Notice that the atom, bond, and axis whose configurations are not shown in Table 5.2 ~ Table 5.4 have a default symmetry number of one.

<table>
<thead>
<tr>
<th>The ( i )th atom</th>
<th>Nearest-neighbor sub-graph configuration</th>
<th>( \sigma_{\text{atom},i} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1 2 3 4</td>
<td>12</td>
</tr>
<tr>
<td>4 ——— Cs ——— 2</td>
<td>A A A A</td>
<td>A</td>
</tr>
<tr>
<td>3</td>
<td>A A A B</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>A A B C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>A B C D</td>
<td>D</td>
</tr>
<tr>
<td>1 ——— Cdd ——— 2</td>
<td>1 2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A A</td>
<td>B</td>
</tr>
<tr>
<td></td>
<td>A B</td>
<td>1</td>
</tr>
<tr>
<td>1 ——— Os ——— 2</td>
<td>1 2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A A</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A B</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1 2 3</td>
<td>6</td>
</tr>
<tr>
<td>Cs ———</td>
<td>A A A</td>
<td>A</td>
</tr>
<tr>
<td>2 ——— 3</td>
<td>A A B</td>
<td>B</td>
</tr>
<tr>
<td>1 ——— Cs ——— 2</td>
<td>1 2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A A</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A B</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 5.2 Calculation of \( \sigma_{\text{atom},i} \) for \( i \)th atom
After the symmetry number for each individual atom, bond, and axis has been determined, the total symmetry number for the whole molecule can be computed from equation (5.18).

This algorithm is different from the traditional algorithm which breaks down symmetry number into external and internal number, but it gives the same results. Table 5.5 shows some complex chemicals' symmetry numbers calculated by this algorithm, and the results are the same as those from the NIST S&P program.

<table>
<thead>
<tr>
<th>The jth bond</th>
<th>Nearest-neighbor sub-graph configuration</th>
<th>( \sigma_{\text{bond},j} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 —— 2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1 —— 2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1 —— 2</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

Table 5.3 Calculation of \( \sigma_{\text{bond},j} \) for jth bond

<table>
<thead>
<tr>
<th>The kth axis</th>
<th>Nearest-neighbor sub-graph configuration</th>
<th>( \sigma_{\text{axis},k} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ( \overset{3}{\underset{4}{\overset{3}{C}}s=Cdd=...=Cdd=Cd} ) 2</td>
<td>1 2 3 4</td>
<td>A A A A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1 ( \overset{3}{\underset{4}{\overset{3}{C}}ds=Cdd=...=Cdd=O} ) 2</td>
<td>1 2</td>
<td>A A</td>
</tr>
<tr>
<td></td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

Table 5.4 Calculation of \( \sigma_{\text{axis},k} \) for kth axis
<table>
<thead>
<tr>
<th>Chemical Formula</th>
<th>Adjacency List Representation</th>
<th>Symmetry Number by RMG</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(\text{CH}_3)_2\text{C}=\text{C}=\text{O}$</td>
<td>1 C 0 {2,D} {5,S} {6,S} 2 C 0 {1,D} {3,D} 3 C 0 {2,D} {4,D} 4 O 0 {3,D} 5 C 0 {1,S} 6 C 0 {1,S}</td>
<td>18</td>
</tr>
<tr>
<td>$(\text{CH}_3)\text{CH}=\text{C}=\text{O}$</td>
<td>1 C 0 {2,D} {5,S} 2 C 0 {1,D} {3,D} 3 C 0 {2,D} {4,D} 4 O 0 {3,D} 5 C 0 {1,S}</td>
<td>3</td>
</tr>
<tr>
<td>$(\text{CH}_3)_2\text{C}=\text{C}=(\text{CH}_3)_2$</td>
<td>1 C 0 {2,D} {4,S} {5,S} {6,S} 2 C 0 {1,D} {3,D} 3 C 0 {2,D} {6,S} {7,S} 4 C 0 {1,S} 5 C 0 {1,S} 6 C 0 {3,S} 7 C 0 {3,S}</td>
<td>324</td>
</tr>
<tr>
<td>$\text{CH}_2=\text{C}=\text{C}(\text{CH}_2)_2$</td>
<td>1 C 0 {2,D} 2 C 0 {1,D} {3,D} 3 C 0 {2,D} {4,S} {5,S} 4 C 0 {1,S} 5 C 1 {3,S} 6 C 0 {3,S} 7 C 0 {3,S}</td>
<td>8</td>
</tr>
<tr>
<td>$(\text{CH}_2)\text{C}=\text{C}(\text{CH}_3)$</td>
<td>1 C 0 {2,T} {3,S} 2 C 0 {1,T} {4,S} 3 C 0 {1,S} 4 C 0 {2,S} 5 C 0 {1,S}</td>
<td>18</td>
</tr>
<tr>
<td>C($\text{CH}_3)_4$</td>
<td>1 C 0 {2,S} {3,S} {4,S} {5,S} 2 C 0 {1,S} 3 C 0 {1,S} 4 C 0 {1,S} 5 C 0 {1,S}</td>
<td>972</td>
</tr>
<tr>
<td>$\text{CH}_4\text{CH}_2\text{CH}_2*$</td>
<td>1 C 1 {2,S} 2 C 0 {1,S} {3,S} 3 C 0 {2,S}</td>
<td>6</td>
</tr>
<tr>
<td>$\text{CH}_3\text{CH}•\text{CH}_3$</td>
<td>1 C 0 {2,S} 2 C 1 {1,S} {3,S} 3 C 0 {2,S}</td>
<td>18</td>
</tr>
<tr>
<td>$\text{CH}_2=\text{CH}_2$</td>
<td>1 C 0 {2,D} 2 C 0 {1,D}</td>
<td>4</td>
</tr>
<tr>
<td>CH(OH)=CH(OH)</td>
<td>1 C 0 {2,D} {3,S} 2 C 0 {1,D} {4,S} 3 O 0 {1,S} 4 O 0 {2,S}</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5.5 Examples of symmetry number calculation by RMG
It is worth mentioning that the calculation of ring structure's symmetry number in RMG has a systematic error, since we don't distinguish an atom in a ring from an atom not in a ring. We can probably set up a set of ring atom symmetry number rules to improve our algorithm in the future.

Although the symmetry number computation is introduced in this section, it is a structure property of a ChemGraph. Therefore, it is stored as an attribute of a ChemGraph, just the same as other ChemGraph properties, like chemical weight, formula, and name, and, when a new ChemGraph is constructed, the function computing the symmetry number is called immediately to determine this property for the ChemGraph. In thermal properties estimation, a ChemGraph’s symmetry number will be accessed to make the corresponding correction to its standard entropy at 298K.

5.4.4.2 Ring correction

Ring structure is typically a non-local feature for species. In Benson’s text, he provides correction values for single ring consisting up to 10 member atoms and for some of the bi-cycles as well. In RMG, we include those ring corrections in a ring correction list, in which every entry includes a FunctionalGroup representation of the ring structure and the ring's correction value. We didn’t use tree to help construct this database, since it is not easy to construct a hierarchy tree for cyclic graphs and since there are small number of ring corrections and searching a small list of ring correction won’t be a big overhead. After all the group contributions of a species have been added up, the whole species will be compared with each of the ring correction structures in this list, and the sum of thermal values of all the distinct corrections will be finally added to the species thermal properties.

5.4.4.3 Cis conformation correction

Benson’s group values were derived for trans conformation in alkenes. When two groups are in the cis position, they repel each other increasing the heat of formation of the alkenes. Benson, therefore, provided corrections for these cases. In RMG, we also built cis correction database in a hierarchy tree structure, and corresponding cis correction values from Benson were arranged into a library.
People should be aware that any chemical species in RMG is represented by a connectivity graph, which can’t provide 3-D orientations for each atom. Therefore, RMG itself cannot figure out any cis or tran conformations of a double bond, unless user specifies it explicitly in the inputted species adjacency list, which is the only circumstance that cis correction will be performed.

5.5 Design highlights for RMG-GATP

In this section, the author discusses the designs on the group additivity thermal properties estimation in RMG, and we called it RMG-GATP.

Figure 5.8 shows the UML design diagram for RMG-GATP. There are three main parts in this diagram. The first part includes two classes, ThermoGAValue and ThermoData to represent the thermal group value and thermal data for species. The second part defines the ThermoGAGroupLibrary, which manages all the thermal group libraries. The third part is our thermo property estimator, GATP, which implements the group additivity algorithm.

In Figure 5.8, the author also shows the relations between ChemGraph class and thermal-related classes. In RMG, besides thermal data attribute, every ChemGraph also has a thermo property estimator as a composition. This gives flexibility to the user to choose different thermal group additivity estimator other than GATP, if later more thermal properties estimators can be developed to implement interface GeneralGAPP.
Figure 5.8 UML design diagram for group additivity thermal property estimator in RMG
5.5.1 Thermal group data representation

There are two types of thermal data classes in RMG-GATP, ThermoGAValue class and ThermoData class.

ThermoGAValue corresponds to the thermal group values stored in group library, HBI radical correction library, and other correction libraries. It has attributes for recording \( \Delta H^0(298K) \), \( S^0(298K) \), \( C_p(300K) \), \( C_p(400K) \), \( C_p(500K) \), \( C_p(600K) \), \( C_p(800K) \), \( C_p(1000K) \), \( C_p(1500K) \), and data source of a thermal group, and corresponding attribute accessing methods. It is noticeable that ThermoGAValue objects are data records, therefore they are immutable objects, i.e., ThermoGAValue objects will remain unchanged.

ThermoData class is to represent the thermal data for a species, not a thermal group. Like ThermoGAValue class, it also has to hold the \( \Delta H^0(298K) \), \( S^0(298K) \), \( C_p(300K) \), \( C_p(400K) \), \( C_p(500K) \), \( C_p(600K) \), \( C_p(800K) \), \( C_p(1000K) \), \( C_p(1500K) \) thermal parameters as its attributes. Therefore, the author made it a subclass of ThermoGAValue, so that it can inherit all the definitions of attributes and data accessing methods from ThermoGAValue. Of course ThermoData class differentiates from its super-class, and it extends ThermoGAValue in two facets. Firstly, ThermoData is a mutable class, whose objects can be changed by some additional methods of arithmetic operations. For example, we can add or minus one ThermoData instances to another ThermoData instance or multiply a scalar to an existing ThermoData instance. Therefore, group additivity algorithm calculation from RMG-GATP can be performed to a ThermoData to estimate the species thermal property. Secondly, as a thermodynamic data for a species, ThermoData is not only able to preserve a set of thermal constants, but also to present thermodynamic properties, such as \( \Delta H^0(T) \), \( S^0(T) \), and \( \Delta G^0(T) \), at different temperatures. To do so, the author appended to ThermoData class a set of thermodynamic properties estimation functions, whose implementations are based on equations (5.7)–(5.10), provided earlier in this chapter.
5.5.2 Thermal group databases

ThermoGAGroupLibrary is a class accommodating all the databases used by
group additivity estimation, including thermal group database, HBI radical correction
database, ring correction database, and other correction database. All of these databases
are in hierarchy tree structures, except that the ring correction database is currently
structured as a list. The documents specifying these four databases are described in
Appendix 5.2.

For each tree-structured database, generally there are three documents needed, the
dictionary, the tree, and library. The dictionary defines the FunctionalGroup graph with a
unique string name for each thermal group in the database. It serves two purposes,
defining the FunctionalGroup graph of a thermal group and providing a shorter, easier
identification, a string name, for each thermal group. The tree document specifies the
tree structure among thermal groups using the string names defined in dictionary. The
library file then provides the thermal group values, also using the string names. For a
list-structure database, there is no tree structure document needed. As our only list-
structured database, the ring correction database only includes a small number of thermal
groups, and its dictionary and library were put together into one document.

Inside ThermoGAGroupLibrary class, it maintains corresponding attributes for
dictionary, tree, and library for every database. Since we only allow one thermal group
library in RMG, ThermoGAGroupLibrary class is designed as a singleton, which means
there is only one instance for that class. (For details on singleton design and
implementation, refer to Gamma, et al., 1995) When the library is initiated, three
documents for each database are read in to set up the dictionary, tree and library attributes
in ThermoGAGroupLibrary. The main functions provided by ThermoGAGroupLibrary
are four searching functions, which implement finding and returning proper thermal
group values, proper HBI radical values, proper ring correction values, and proper other
correction values, respectively. For the tree structured database, searching function
moves from the top to the bottom of the tree until a best matched tree leaf is found; using
the FunctionalGroup stored in that leaf as a key, the searching function is able to grasp
the corresponding thermal value, an instance of ThermoGAValue, from the library and
return it. For the list structured database, searching function checks through the whole list to gather all the distinctive matched thermal group values, and return them as a collection.

5.5.3 Group additivity thermal property estimation

GeneralGAPP is an interface, which requires implementing only one function, generateThermoData(ChemGraph). This interface is designed to regulate all the thermal properties estimators being developed and used, and GATP is the first one.

GATP, as the class actually performing the group additivity estimation for a ChemGraph, is the heart of thermal properties calculation in RMG. GATP, similar to ThermoGAGroupLibrary, was also designed as a singleton. It also includes the ThermoGAGroupLibrary as a composition, so that the libraries for group searching are accessible.

The major duty of this class is to implement generateThermoData(ChemGraph) declared in GeneralGAPP, which estimates the ThermoData for the passed-in ChemGraph object. The main steps in implementing this function in GATP are shown as following:

Step1: if ChemGraph is a free radical, add hydrogen(s) to obtain its stable molecule structure;

Step2: partition the stable molecule into a collection of non-H atoms;

Step3: search through ThermoGAGroupLibrary to gather the group value for each non-H atom;

Step4: sum up the group contributions from step3 into a ThermoData object;

Step5: if ChemGraph is a free radical, find proper HBI radical group from ThermoGAGroupLibrary, and make HBI correction to ThermoData;

Step6: correct standard entropy at 298K by ChemGraph’s symmetry number (that has been already calculated once a ChemGraph is constructed);

Step7: add ring correction to ThermoData, if any;

Step8: add other correction to ThermoData, if any;
5.6 Conclusions and discussions

In this section, author presents the design ideas and implementation details on group additivity thermal property estimation algorithm in RMG. Based on the group additivity method proposed by Benson, we presented a new data-model based implementation for the algorithm, which is more flexible for user to build, modify, and extend. Moreover, we proposed using a hierarchy tree to model the thermal group database, which provides also the flexibility and extensibility for users to maintain and manage large thermal databases. We also collected a large number of thermal group values published in recent years, and selected the most reliable ones to build our thermal group library and correction libraries.

RMG-GATP, the group additivity thermodynamic properties estimator in RMG, was implemented to calculate any molecule’s thermodynamic properties at different temperatures to an acceptable precision level, and, therefore, it provides important, necessary information to predict the thermodynamic property and reaction kinetics for a complex reaction system.

Although the group additivity thermal estimation is a powerful method to calculate thermodynamic properties for all kinds of species, especially for ones where no experimental data is available, this method have some shortcomings. Since this algorithm is based on the assumption that the “local interactions” between components in a molecule plays the key role, the “global interactions” effects are not taken into account as systematically as those from the “local interactions”. For example, the most typical global interaction effect, ring effect, although included as a correction supplementary to group additivity algorithm, is still too far from being thoroughly, systematically considered to provide a good estimation for complex ring structures, such as bipartite graphs. In general, cyclic graph structures bring difficulties to graph-related algorithm development in many graph-based applications, not just molecular thermochemistry.
Appendix 5.1 Derivation of Equation (5.8) and (5.9)

In this derivation, the known thermal parameters are standard enthalpy and standard entropy at 298K, $\Delta H^0(298K)$ and $S^0(298K)$, as well as a series of heat capacity from 300K to 1500K, $C_p(300K)$, $C_p(400K)$, $C_p(500K)$, $C_p(600K)$, $C_p(800K)$, $C_p(1000K)$, and $C_p(1500K)$, and the functional integration expressions for enthalpy and entropy at any temperature are as following:

$$\Delta H^0(T) = \Delta H^0(T_0) + \int_{T_0}^{T} C_p(T')dT$$ (5.5)

$$\Delta S^0(T) = S^0(T_0) + \int_{T_0}^{T} \frac{C_p(T)}{T}dT$$ (5.6)

The goal is to derive a practical expression, i.e. a numerical integration result for $\Delta H^0(298K)$ and $S^0(298K)$ using the known thermal parameters.

We can divide the whole temperature range into six segments, (300K, 400K), (400K, 500K), (500K, 600K), (600K, 800K), (800K, 1000K), and (1000K, 1500K), and the cutting temperatures are corresponding to the sample temperatures of heat capacity. The total entropy and enthalpy integrals are the sums of the contribution from each temperature segment.

Now the problem turns into obtaining the integration expression for each $(T_i, T_{i+1})$ temperature range. In every $(T_i, T_{i+1})$, the $C_p(T)$, where $T \in (T_i, T_{i+1})$, can be approximately expressed by linear interpolation as following:

$$C_p(T) = aT + b$$ (5.19)

where

$$a = \frac{C_p(T_{i+1}) - C_p(T_i)}{T_{i+1} - T_i}$$

and

$$b = \frac{C_p(T_i) \cdot T_{i+1} - C_p(T_{i+1}) \cdot T_i}{T_{i+1} - T_i}.$$
Plugging (5.19) into (5.5) and (5.6) will give the numerical integration expression as following:

\[
\Delta H^0(T_i \sim T_{i+1}) = \int_{T_i}^{T_j} C_p(T)\,dT
\]

\[
= \int_{T_i}^{T_j} (aT + b)\,dT
\]

\[
= \frac{a(T_j^2 - T_i^2)}{2} + b(T - T_i) \quad (5.20)
\]

\[
S^0(T_i \sim T_{i+1}) = \int_{T_i}^{T_j} C_p(T)\,dT
\]

\[
= \int_{T_i}^{T_j} (a + \frac{b}{T})\,dT
\]

\[
= a(T - T_i) + b \ln\left(\frac{T}{T_i}\right)
\]

After considering the boundary condition, we can summarize the results into (5.8) and (5.9) as following:

\[
\Delta H^0(T) = \Delta H^0(298) + \sum_{i=0}^{s} \Delta H^0(T_i \sim T_{i+1})
\]

where \(\Delta H^0(T_i \sim T_{i+1}) = \begin{cases} 
0 & T < T_i \\
\frac{a(T^2 - T_i^2)}{2} + b(T - T_i) & T_i \leq T < T_{i+1} \\
\frac{2}{C_p(T_i) + C_p(T_{i+1})} \cdot (T_{i+1} - T_i) & T_{i+1} \leq T
\end{cases} \quad (5.8)
\]

\[
\Delta S^0(T) = S^0(298) + \sum_{i=0}^{s} \Delta S^0(T_i \sim T_{i+1})
\]

where \(\Delta S^0(T_i \sim T_{i+1}) = \begin{cases} 
0 & T < T_i \\
a(T - T_i) + b \ln\left(\frac{T}{T_i}\right) & T_i \leq T < T_{i+1} \\
a(T_{i+1} - T_i) + b \ln\left(\frac{T_{i+1}}{T_i}\right) & T_{i+1} \leq T
\end{cases} \quad (5.9)
\]

119
Appendix 5.2 Thermal Group Library Documents

Here the author lists all the documents, their file name, their text format, and their descriptions for thermal group additivity estimation method.

Notice that there is a dictionary file for the thermal group and HBI radical correction. The main function of this dictionary file is to build a one-to-one mapping between a String FunctionalGroup name and the internal FunctionalGroup graph representation, so that when the tree and library information is read in, the FunctionalGroup name can be used as a unique index for the thermal group. This is just for I/O convenience. Internally, both the trees and libraries are indexed by FunctionalGroup Graph objects instead of their names. For ring and other correction, since the number of FunctionalGroups involved is not large, the dictionary part is put in together with library.

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Document Description</th>
<th>Document Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group_Dictionary.txt</td>
<td>Definitions of thermal FunctionalGroup structures</td>
<td>FunctionalGroup name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FunctionalGroup adjacency list</td>
</tr>
<tr>
<td>Group_Tree.txt</td>
<td>The thermal group tree</td>
<td>TreeLevel# : FunctionalGroup name</td>
</tr>
<tr>
<td>Group_Library.txt</td>
<td>Thermal group values assigned to thermal FunctionalGroups</td>
<td>FunctionalGroup name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group value in Benson format</td>
</tr>
<tr>
<td>Radical_Dictionary.txt</td>
<td>Definitions of HBI radical FunctionalGroup</td>
<td>FunctionalGroup name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FunctionalGroup adjacency list</td>
</tr>
<tr>
<td>Radical_tree.txt</td>
<td>The HBI group tree</td>
<td>TreeLevel# : FunctionalGroup name</td>
</tr>
<tr>
<td>Radical_Library.txt</td>
<td>HBI group values assigned to HBI radical FunctionalGroups</td>
<td>FunctionalGroup name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group value in Cp format</td>
</tr>
<tr>
<td>Ring_Correction.txt</td>
<td>Definition of ring correction list</td>
<td>FunctionalGroup name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group value in Cp format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FunctionalGroup adjacency list</td>
</tr>
<tr>
<td>Other_Tree.txt</td>
<td>The other correction tree</td>
<td>TreeLevel# : FunctionalGroup name</td>
</tr>
<tr>
<td>Other_Library_Dictionary.txt</td>
<td>Other correction values</td>
<td>FunctionalGroup name</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Group value in Cp format</td>
</tr>
<tr>
<td></td>
<td></td>
<td>FunctionalGroup adjacency list</td>
</tr>
</tbody>
</table>
Chapter 6  Generating Elementary Reactions by Reaction Templates

A reaction mechanism consists of a set of elementary reaction steps. Elementary reactions are the most important components for a reaction mechanism. In this chapter, the author describes how RMG derives elementary reactions from chemical species. The package rxn in RMG takes care of generating elementary reactions from the species. In other words, given a set of species, rxn package is responsible for enumerating all the elementary reactions among the species set. Before discussing reaction generation procedure, the author gives a brief overview of elementary reactions.

6.1 Overview of chemical elementary reaction

According to IUPAC [McNaught and Wilkinson, 1997], an elementary reaction is defined as “a reaction for which no reaction intermediates have been detected or need to be postulated in order to describe the chemical reaction on a molecule scale. An elementary reaction is assumed to occur in a single step and to pass through a single transition state.” We can represent an elementary reaction in such a way:

$$ R_1 + R_2 \xrightarrow{k} R_3 + R_4 $$

(6.1)

where $R_1$ and $R_2$ are reactants, and $R_3$ and $R_4$ are products.

Besides participating species, there are two types of crucial reaction properties dealing with reaction’s equilibrium and dynamics respectively. Thermodynamic properties of an elementary reaction, such as $\Delta H_{\text{rxn}}$, $\Delta G_{\text{rxn}}$, $K_{\text{eq}}$ etc., define the reaction’s equilibrium status, and kinetics constant $k$ defines how the reaction system changes with respect to time. Thermodynamic properties of the reaction can be easily calculated from the reactant/products thermodynamic parameters, which are already given by the thermal properties estimator discussed in the last chapter. Kinetics constant $k$ is a temperature-dependent parameter described by modified Arrhenius expression:

$$ k(T) = A \cdot T^n \cdot \exp\left(-\frac{E}{RT}\right) $$

(6.2)
where \( A \) is the pre-exponential factor, \( n \) is the temperature exponential factor, \( E \) is the activation energy, \( R \) is the gas constant, and \( T \) is the temperature. Kinetics parameters referred commonly by people correspond to a set of temperature-independent constants of \( A, n, \) and \( E \). Sometimes, people relate the activation energy with the \( \Delta H_{\text{rxn}} \):

\[
E = \alpha \cdot \Delta H_{\text{rxn}} + E_0
\]  

(6.3)
called an Evans-Polanyi relationship. In such case, the kinetics parameter set is a set of constants of \( A, n, \alpha, \) and \( E_0 \).

Obtaining kinetics parameters for elementary reactions is not as easy as getting thermodynamic data. People have generally two ways to estimate the kinetics constant, experimental fitting and quantum chemical calculation based on transition state theory [Eyring, 1935; Evans and Polanyi, 1935]. Researchers also have published research results about kinetics for hundreds of elementary reactions, which are reviewed and collected in compendiums, such as, NIST kinetics database. Therefore, to obtain any elementary step reaction kinetics, we can either do the experiments, or try transition state theory calculation, or search through the kinetics databases. With the proper thermodynamic and kinetic data provided, an elementary reaction is considered completely defined.

In summary, generating an elementary reaction is to describe the reactants and products species as well as to provide the proper thermodynamic parameters and kinetic constants.

6.2 Reaction Family

Elementary reactions can be classified into different reaction groups according to their reacting mode. For example, any reaction where a radical abstracts one hydrogen atom from a stable molecule is classified as a hydrogen abstraction reaction. In pyrolysis and oxidation reaction systems, there exist many reaction classes, such as hydrogen abstraction, hydrogen migration, bond dissociation, \( \beta \)-scission, disproportionation, cyclo-addition, cyclic ether formation, etc. In RMG, we call such reaction classes as “reaction families”. The similarity of the reactions in the same reaction family lies in three aspects. First, in the same reaction family, although the reactants and products from different
individual elementary reactions are different chemical species, the reactants participating must have the same reacting center. For example, for all the hydrogen abstraction reactions, the reactant acting as radical abstractor must react at its radical center to abstract a hydrogen atom from the other reactant, while, at the same time, the other reactant must react at its hydrogen atom site to lose this hydrogen forming a new radical center. There is no other type of acting mode in the same reaction family. Secondly, since the acting mode is the same, the transition states of the reactions in the same reaction family should generally look the same at the reacting centers. Therefore, it leads to the third similarity that in the same reaction family, all the reactions have quite similar kinetic constants. Of course, it is not easy to define how close the kinetics of different reactions in the same family could be, which greatly depends on the similarity of their transition states structures. Nevertheless, this similarity still gives a potential understanding and sometimes even a way to estimate the kinetics of some new reaction from its relatives in the same family.

Knowing the reaction families is the first step to understand the types and features of elementary reactions in a chemical system. People have different methods of summarizing and categorizing of reaction families in their reaction mechanism modeling work. For example, in the work modeling n-heptane oxidation by Curran et al. [2002], twenty-five reaction classes were summarized. In RMG, we spent a great amount of time in the design stage studying what reaction classes should be included in our reaction system. The documents of summarizing our reaction families are given in Appendix 6.1. In appendix 6.1, Dr. S. Raman and the author give an overview of thirty-nine the reaction families and rules to make reactions.

In table 6.1, the author summarizes the thirty-three reaction families implemented in RMG. The first column assigns each reaction family an index, which corresponds to the index in the documentation of Appendix 6.1. People interested in definition details for reaction families may refer to Appendix 6.1 by the reaction family index. The reaction family set in RMG is the most complete yet compiled for pyrolysis and oxidation system.
<table>
<thead>
<tr>
<th>Index</th>
<th>Name</th>
<th>Reaction Pattern</th>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>F1</td>
<td>Inter-molecular Hydrogen Abstraction</td>
<td>X-H + Y. -&gt; X. + Y-H</td>
<td></td>
</tr>
<tr>
<td>F2</td>
<td>Radical Addition to Multiple Bond</td>
<td>Y. + X^\circ Z &lt;- Y-X(-1)-Z.</td>
<td>^is = or ≡</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Y. + X^\circ Z -&gt; Y-Z(-1)-X</td>
<td></td>
</tr>
<tr>
<td>F3</td>
<td>Beta Scission</td>
<td>reverse of F2</td>
<td></td>
</tr>
<tr>
<td>F4</td>
<td>Radical Addition to CO</td>
<td>Y. + CO -&gt; Y-C=O</td>
<td></td>
</tr>
<tr>
<td>F5</td>
<td>CO Elimination from Carbonyl</td>
<td>reverse of F4</td>
<td></td>
</tr>
<tr>
<td>F6</td>
<td>Radical Recombination</td>
<td>X. + Y. -&gt; X-Y</td>
<td></td>
</tr>
<tr>
<td>F7</td>
<td>Bi-radical Recombination to Form Cyclic Structure</td>
<td>X.-(^n-1)Y. -&gt; X.-(^n)Y</td>
<td></td>
</tr>
<tr>
<td>F8</td>
<td>Bond Dissociation</td>
<td>reverse of F6</td>
<td></td>
</tr>
<tr>
<td>F7R</td>
<td>Ring open</td>
<td>reverse of F7</td>
<td></td>
</tr>
<tr>
<td>F9</td>
<td>Disproportion</td>
<td>Y. + H-X^\circ Z. -&gt; Y-H + X(^+1)Z</td>
<td>^is – or =</td>
</tr>
<tr>
<td>F10</td>
<td>Molecular Addition</td>
<td>reverse of F9</td>
<td></td>
</tr>
<tr>
<td>F11</td>
<td>1,2 Insertion</td>
<td>:CO_{\text{a}}/CH_{\text{a}}/O_{\text{a}} + X-Y -&gt; X-(CO/CH_{\text{a}}/O)-Y</td>
<td></td>
</tr>
<tr>
<td>F12</td>
<td>1,1 Elimination</td>
<td>Reverse of F11</td>
<td></td>
</tr>
<tr>
<td>F13</td>
<td>1,3 Insertion</td>
<td>W-Z + X^\circ Y. -&gt; W-X(-1)Y-Z</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>W-Z + X^\circ Y. -&gt; W-Y(-1)Y-X</td>
<td></td>
</tr>
<tr>
<td>F14</td>
<td>1,2 Elimination</td>
<td>Reverse of F13</td>
<td></td>
</tr>
<tr>
<td>F15</td>
<td>1+2 Cyclo-addition</td>
<td>X=\text{Y} + :\text{CR}<em>{\text{a}}/\text{O}</em>{\text{a}} -&gt; △</td>
<td></td>
</tr>
<tr>
<td>F16</td>
<td>Three-Ring Cleavage</td>
<td>Reverse of F15</td>
<td></td>
</tr>
<tr>
<td>F17</td>
<td>2+2 Cyclo-addition</td>
<td>= + = -&gt; □</td>
<td>Two way to react</td>
</tr>
<tr>
<td>F18</td>
<td>Four-Ring Cleavage</td>
<td>reverse of F17</td>
<td></td>
</tr>
<tr>
<td>F19</td>
<td>Diels-Alder Addition</td>
<td>Two way to react</td>
<td></td>
</tr>
<tr>
<td>F20</td>
<td>Retro Diels-Alder Addition</td>
<td>reverse of F19</td>
<td></td>
</tr>
<tr>
<td>F23</td>
<td>Keto-Enol Tautomerism</td>
<td>X-C(O)-Y-H -&gt; X-C(OH)=Y</td>
<td></td>
</tr>
<tr>
<td>No.</td>
<td>Reaction</td>
<td>Reaction Template</td>
<td></td>
</tr>
<tr>
<td>-----</td>
<td>----------------------------------------------</td>
<td>-------------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td>F24</td>
<td>Enol-Keto Tautomerism</td>
<td>reverse of F23</td>
<td></td>
</tr>
<tr>
<td>F27</td>
<td>Intra-Molecular Addition across Multiple Bond</td>
<td>X-(______)Y^___Z -&gt; X-(______)Y^____Z.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to Form an Exo-Cyclic Radical</td>
<td>^ is = or ≡</td>
<td></td>
</tr>
<tr>
<td>F28</td>
<td>Ring Open for Exo-Cyclic Radical</td>
<td>reverse of F27</td>
<td></td>
</tr>
<tr>
<td>F29</td>
<td>Intra-Molecular Addition across Multiple Bond</td>
<td>X-(______)Y^___Z -&gt; X-(______)Y^____Z.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to Form an Endo-Cyclic Radical</td>
<td>^ is = or ≡</td>
<td></td>
</tr>
<tr>
<td>F30</td>
<td>Ring Open for Endo-Cyclic Radical</td>
<td>reverse of F29</td>
<td></td>
</tr>
<tr>
<td>F31</td>
<td>Cyclic Ether Formation from Alkyl-Hydroperoxy Radical</td>
<td>X-(______)O-O-H -&gt; X-(______)O + O.-H</td>
<td></td>
</tr>
<tr>
<td>F32</td>
<td>OH Addition to Cyclic Ether</td>
<td>reverse of F31</td>
<td></td>
</tr>
<tr>
<td>F33</td>
<td>Intra-Molecular Hydroxyl Migration</td>
<td>X-(______)O-O-H -&gt; H-O-X-(______)O.</td>
<td></td>
</tr>
<tr>
<td>F33R</td>
<td>Reversed Intra-Molecular Hydroxyl Migration</td>
<td>reverse of F33</td>
<td></td>
</tr>
<tr>
<td>F34</td>
<td>HO_2 Elimination from Peroxy Radical</td>
<td>H-X-Y-O-O-H -&gt; X=Y + H-O-O.</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1 Implemented 34 reaction families in RMG

6.3 Modeling reaction families by reactions templates

After understanding the chemistry concept of a reaction family, we are about to teach computer to understand this concept and to build computer model representations for reaction families. Such computer-modeled reaction family is called a reaction template in RMG, and its main duty is that, given a set of chemical species, it will provide a complete set of the elementary reactions belonging to this reaction family that can possibly occur among the given species. More specifically, a reaction template should be able to generate elementary reactions from species and to provide precise thermodynamic and kinetic parameters for the generated reaction.
6.3.1 Generating elementary reactions from species

In order to generate an elementary reaction from a species, the reaction template is responsible for two jobs: (1) identifying if the species is a reactant candidate for this reaction family and where the reaction will happen, and, (2) if the species can participate in the reaction, making the proper product from the reactants.

To illustrate this process, let us take a look at an example, hydrogen abstraction, abbreviated as “H Abs”, reaction template, which can generally be represented as:

$$X-H + Y\bullet \rightarrow X\bullet + Y-H$$  \hspace{1cm} (6.3)

Where X and Y can be any chemical structure. This representation provides a clear restriction on the chemical structures that can participate in an H Abs reaction: a species with a –H group and a species with a radical center. Therefore, identifying the proper reactant for an H Abs reaction turns into checking if two chemical structures have a –H group and a radical center respectively. Equation (6.3) also describes how the reactant reacts and what are the products formed: the radical group Y• abstracts –H group from the stable molecule X-H to form a new stable molecule Y-H and a new radical X•.

6.3.1.1 Hard-coded reaction generation approach

To implement the procedure for checking reactants and making products is similar to the procedure for identifying the best matched thermal group described in section 5.4.1, where the author has discussed and compared two different implementations: hard-coded approach and data-model based approach, and concluded that the data-model based method is more flexible and extensible. In this work of generating reactions, the same conclusion applies. Before this work, many reaction generation programs implemented reactant identification and reacting action in a hard-coded way, i.e., the specification of the qualified structure testing and the mutation of reactant structures to form products are hard-coded into the reaction generation approach. Typical pseudo codes for finding reactant candidates and making reaction for H Abs reaction family can be summarized as:

```c
FindReactionSiteForHAbstraction(Species1, Species2) {
    if (Species1 has a –H group && Species2 has a radical center) {
        Record and return the reacting centers at each species;
    }
}
```
} else if (Species2 has a \(-H\) group && Species1 has a radical center) {
    Record and return the reacting centers at each species;
}
else {
    return a not-found sign;
}

DoHAbstractionReaction(Species1, Species2) {
    ReactionSites = FindReactionSiteForHAbstraction(Species1, Species2);
    if (ReactionSites is not empty) {
        add \(-H\) onto the original radical forming a new molecule;
        remove \(-H\) from the original molecule forming a new radical;
    }
}

Figure 6.1 Pseudo code for implementing reactant generation in a hard-coded way

Using such approaches, every time people need to generate H Abs reaction between two species, they have to firstly call FindReactionSiteForHAbstraction() to test if and where those species can react as H Abs reaction, and then DoHAbstractionReaction() to perform the reacting action. Similarly, for all the reaction families, there exist a pair of Find...() and Do...() functions to perform such work.

The disadvantage of the hard-coded approach is obvious that a large amount of similar codes have to be written for each reaction template, although the only difference between the Find...() and Do...() functions from different reaction templates is the detailed structure restriction and reacting action. As a result, in the reaction generation package, there exist many blocks of similar codes serving for the same purpose, which unnecessarily wastes much developing time on writing, debugging, and testing on the similar functions. Furthermore, things could become really ugly if people want to make an overall change to all the Find...() or Do...() Functions after they have been completed, which means people should repeat the same change thirty or forty times, and any careless little mistake might cause a fatal malfunction.

6.3.1.2 Data-model based reaction generation approach

Identify the reactants

Having understood the disadvantages of hard-coded reaction generation, we proposed an alternative solution, a data-model based reaction generation approach.
Studying equation (6.3) more carefully shows that the restriction on reactant structure can be easily converted into a FunctionalGroup representation, shown as following:

\[
\begin{align*}
\text{R} & \quad \text{H} \\
\text{R} & \quad \text{H} \\
\text{R} & \quad \text{H}
\end{align*}
\]

<table>
<thead>
<tr>
<th>ID</th>
<th>CentralID</th>
<th>Atom</th>
<th>Radical</th>
<th>Adjacency List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*1</td>
<td>R</td>
<td>0</td>
<td>{2,8}</td>
</tr>
<tr>
<td>2</td>
<td>*2</td>
<td>H</td>
<td>0</td>
<td>{1,5}</td>
</tr>
</tbody>
</table>

Figure 6.2 FunctionalGroup representations of reactants

(Note that here we only show mono-radical as example. In RMG, we also allow biradicals to participate in H abstraction reactions)

With such representation, the structure restriction on reactants for every reaction template can be easily converted into a group of FunctionalGroups, which can be defined by drawing the FunctionalGroup graph and/or writing the FunctionalGroup adjacency list. We named those FunctionalGroup as “reactant pattern” of reaction template. In RMG, every reaction template has its reactant pattern defined in FunctionalGroup format shown in Figure 6.2, which precisely replaces the structure restrictions on reactants programmed in conventional reaction generation programs. Therefore, reactant pattern is actually a proper extraction of chemistry details out of the program. With its definition, checking if a species is a candidate for a reaction template and finding all the possible reactive sites, in turn is simply checking whether a species is a subgraph of a reaction template’s reactant pattern and finding all possible matched sites between a species and a reactant pattern. Therefore, one standard algorithm to identify subgraph relations between a Species and a FunctionalGroup, and one standard algorithm to find all matched sites between a Species and a FunctionalGroup, both discussed in chapter 3, can be employed here to replace all the old Find...() functions performing the reactant identification. As a result, developing, debugging, testing, maintaining, and modifying one single subgraph matching program is much easier, more flexible, more bug-free than working on thirty or forty similar Find...() functions.
Making reactions

After reactant candidates are identified, proper reacting action should be performed to produce the product. Unlike the hard-coded approach, where every single reacting action is written as computer code, the author proposed a data-model based reacting method, called reacting based on a “reaction recipe”.

Reaction recipe is defined as a series reacting actions specific to each reaction template. For example, for H Abs reaction template, the action should be adding –H group to radical center and removing a free electron from it, and abstracting –H group from molecule and add a free radical to it, which can be represented as following:

\[
\begin{array}{ccc}
& *1 & *2 \\
& R & H \\
\end{array} +
\begin{array}{c}
*3 \\
R* \\
\end{array}
\]

<table>
<thead>
<tr>
<th>ID</th>
<th>CentralID</th>
<th>Atom</th>
<th>Radical</th>
<th>Adjacency List</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*1</td>
<td>R</td>
<td>0</td>
<td>{2,S}</td>
</tr>
<tr>
<td>2</td>
<td>*2</td>
<td>H</td>
<td>0</td>
<td>{1,S}</td>
</tr>
</tbody>
</table>

Reaction Recipe:

1. BREAK_BOND \{*1,S,*2\}
2. FORM_BOND \{*2,S,*3\}
3. GAIN_RADICAL \{*1,1\}
4. LOSE_RADICAL \{*3,1\}

Figure 6.3 Reaction recipe for H abstraction reaction template

*1, *2, and *3 indicate the atom sites in the reactant pattern FunctionalGroup:

R means any C/O/H atom, S means single bond

Each line in the reaction recipe represents an individual reacting action, which consists of two main parts, type of action and action details about where and how the action is performed. The first string in each reacting action line after the index, such as “BREAK_BOND”, “FORM_BOND”, “GAIN_RADICAL”, and “LOSE_RADICAL” are the keyword defining action type, and the following String enclosed with a pair of braces
describe where and how the reacting action works. At present five types of reacting actions are implemented in RMG: break-bond, form-bond, change-bond, gain-radical, and lose-radical.

Break-bond action means removing the existing bond between two atoms. The action details given in the braces, tells where the action happens, and what bond should be broken, for example, in line (1) in Figure 6.3, {*1, S, *2} indicate that a single bond between *1 and *2 atom sites in the reactant pattern should be removed. It is noteworthy that the reaction recipe definition should be consistent with FunctionalGroup definition in the reactant pattern. In this example, it means that there should exist a single bond between *1 and *2 atom in R-H FunctionalGroup.

Form-bond action is adding a new bond between two atoms, the opposite action of break-bond. Similar to Break-bond, the action details in the brace indicate where the new bonds are formed and the type of new bond. For example, in the example in Figure 6.3, the form-bond action will be performed between *2 and *3 atom sites to form a new single bond.

Change-bond action is not shown in the example, but it is a quite necessary one happening in some important reaction templates, like radical addition to a multiple bond, cyclo-addition, etc., where the original bond is not broken but reduced or increased one bond order. For example, if a radical is added to a triple bond, the reacting action will change the bond order from 3 to 2, meaning a triple bond reduced to a double bond. The keyword of this action is “CHANGE_BOND”, and the action details looks like {*1, -1, *2}. *1 and *2 indicate the two atom sites connected by the original multiple bond, and -1 means reduce bond order by one.

Gain-radical action means adding free electron(s) at an atom site. The action details defined in the brace tell what atom gains electron(s) and how many electron(s) will be added to that atom.

Lose-radical action is the opposite action of Gain-radical, and it indicates how many free electron(s) will be removed from the specified atom site. It is also noteworthy that this action should be consistent with the reactant pattern FunctionalGroup definition,
i.e., if this action says removing $n$ electron(s) from an atom, the corresponding atom site
defined in reactant pattern FunctionalGroup should have more than $n$ electron(s).

With the five reacting actions defined, all the reaction patterns can be described
precisely, and, in this way, people can also make their own reaction recipe quickly.

6.3.1.3 Definition of all the reaction families

Using the new data-model approaches on identifying reactants and making
reaction, we defined totally 34 reaction families shown in Table 6.1. Appendix 6.1
provides the detailed descriptions of the reaction families, which were created and
documented by S. Raman and the author before RMG was written. During RMG
development, we made some changes to the original reaction families’ documentation,
and people should refer to the files described in Appendix 6.2 for the latest updates.

6.4 Obtaining elementary reaction parameters

After the reaction has been generated from a reaction template, all the related
parameters, like thermodynamic properties and kinetic properties, should be provided as
well.

6.4.1 Thermodynamic parameter

Thermodynamic properties of an elementary reaction mainly include $\Delta H_{\text{rxn}}$, $\Delta G_{\text{rxn}}$,
$K_{\text{eq}}$ that can be straightforwardly calculated from the stoichiometric summation of the
same thermodynamic properties of the reactants and products of the elementary reaction.

6.4.2 Thermodynamic consistency

In section 5.1, we have introduced the concept of thermodynamic consistency to
illustrate the relation between thermodynamic properties and kinetics properties.
Thermodynamic consistency states that for any reversible reaction:

$$\text{Reactant(s)} \rightleftharpoons \text{Product(s)}$$

there exists a relation between forward and backward reaction kinetics as following:
\[
\frac{k_f}{k_b} = K_{eq} = (RT)^{-\Delta n} \exp(-\Delta G_{rea} / RT)
\]

where \(K_{eq}\) is equilibrium constant of the forward/backward reaction, \(T\) is the reaction temperature, \(R\) is the gas constant, \(\Delta G_{rea}\) is the reaction free energy, and \(\Delta n\) is the mole change in the reaction.

Equation (6.4) reflects a natural relation between forward kinetics, backward kinetics, and thermodynamic equilibrium constant of a reaction, and if people are able to calculate or measure them all in good precision level, such relations should be satisfied automatically. However, current calculation and/or experimental methods for kinetics and thermodynamic properties estimation all have quite large error bars, so that equation (6.4) could be frequently broken, if we estimate all three parameters independently. People notice that, generally, comparing to kinetic rate estimations, the methods for thermodynamic properties estimation have better precision and basically give more reliable results for \(K_{eq}\). Consequently, people decide to use the forward kinetics and \(K_{eq}\) as independent parameters, and the backward kinetics will be calculated by equation (6.4).

In RMG, as shown in Table 6.2, we have chosen the directions for each reaction family. For a forward reaction family, it has its own kinetics rate library, where reactions in such family will get the kinetics directly; for a backward reaction family, it will refer to its corresponding forward reaction family’s kinetics rate library, which will be used, combined with reaction thermodynamics, to calculate kinetics of reactions in backward reaction family.

People might have noticed that there exist some special reaction families, whose forward and backward reactions are the same type, like H abstraction. For such reaction family, we still maintain its kinetics rate library, but the directions of each individual reaction will be determined by the reaction thermodynamics; that is exothermic reaction is considered as forward, and endothermic direction is backward.

6.4.3 Kinetics parameters

Handling the kinetics parameters for the complex reaction mechanism is a quite hard issue that has been bothering people for years. The major difficulty is to smartly
manage a complex kinetics database consisting of a large number of data from different sources, maybe in different formats, with different levels of uncertainties and different application restrictions. For example, some of them are from fitting the experimental data, and others are from the quantum calculations; some of them are for one single elementary step, but others are raw estimations for a group of similar elementary reactions. Can we organize so many different types of reaction kinetics systematically into one kinetics library? How to evaluate their qualities and how to select the proper ones for different reaction situations? How to find a substituting kinetics for a reaction if the exact kinetics is not found in existing database? Those are hard questions to answer and always confuse people building and using the kinetics database.

Many present kinetics databases just simply put everything together, and provide different ways to search for all the possibly matched reactions' kinetics, instead of organizing the database into a better structure at the first place. In RMG, we presented a novel idea to systematically manage a large number of kinetics rate rules into hierarchy tree structures, which is proposed by Dr. S. Raman.

The idea of using hierarchy tree structure originates from the fact that for a group of similar elementary reactions in one family, very similar kinetics parameters apply. For example, elementary reactions from the same reaction family generally tend to have closer kinetics than the ones from different reaction families, and kinetics experts are able to provide a raw estimation of the kinetics for a family of reactions. However, people are not satisfied with just one set of kinetics parameters for a large family of reactions, some of which could be quite different, like these two H abstraction reactions: \( H + C_2H_4 \rightarrow H_2 + C_2H_3 \) and \( C_6H_5 + Cl_4 \rightarrow C_6H_6 + CH_3 \). The solution to provide more precise kinetics is to subcategorize each reaction families into subfamilies, and assign different kinetics for the subfamilies. For example, Curran et al. [2002], subcategorizes H abstraction reaction families into 44 smaller subgroups according to the stable molecule's structure and radical's property, and provided the rate rule for each H abstraction subfamily. The categorization method they use is to distinguish H abstraction sites into 4 types, and to divide radicals into 11 types; the combination of any types of H abstraction site and radical will be a valid H abstraction subfamily, and have a specific rate rule.
The approach provided in RMG to handle kinetics data is originated from this subcategorization idea, but it has a better, improved, systematic way to subgroup a big reaction family: hierarchy-tree subcategorization. Instead of just one layer of subcategorization used in other literatures, hierarchy tree structure offers multiple levels of systematic subgroups, and provides users a clear picture on how to categorize the subfamilies. After the subcategorization is completed, corresponding kinetics rate rules collected from literature will be assigned to each subfamily to finally form our kinetics rate rules database. Once a new reaction is generated from a reaction family, RMG will automatically search through this tree-structured database to find the best-matched rate rules for this reaction.

6.4.3.1 Building hierarchy tree for kinetics database

Similar to the hierarchy tree used in organizing thermal group data, the hierarchy tree built for the kinetics database are built following the four rules mentioned in section 5.4.2.2, i.e., FunctionalGroup element rule, generality-specificity rule, disjoint siblings rule, and full partition rule.

Shown in Figure 6.4 and Figure 6.5 are two hierarchy trees, R-H tree and R\• tree, corresponding two reactants, for the H abstraction reaction family.

Let us take a close look at R-H tree to understand how this method works. In the first root level of R-H tree, only a single bond connected to one hydrogen atom is required, and the atom at the other end of the single bond does not matter at this level. The activation barrier for H abstraction is known to depend upon the bond strength of both the forming and breaking bonds. So, it is essential to classify R-H and Radical into different subclasses and form concise families with nearly comparable bond strength. On the second level, we specify the type of the nearest neighbor atom, which can be H, Cs, Cd, CO, Ct, Cb, and Os. All the possible types of C/H/O molecules are covered here, and no chemical structure overlaps exist between siblings. If users are satisfied with such specification, we can stop branching tree here; but if people feel the tree needs more level specifying more details other than the nearest atom type, we could keep moving down to the next level. Theoretically, all the second-level nodes except H-H could have more sublevels, since they all have undetermined neighbor atoms or bonds. Below Cs-H node,
since there are three other atoms bonded to Cs by three single bonds, and they all could be H, Cs, Cd, CO, Ct, Cb and Os; therefore, there would be too many combinatorial possibilities. We realized that it made more chemical sense to merge some similar nodes into one, and used only the number of hydrogen to branch Cs-H node into four children, Cs with four H, with three H, with two H, and with only one H. For the other second-level nodes, Cd-H, CO-H, and Os-H, using the similar hydrogen number rule, we divided them into 2 children respectively, to count for two cases in which 0 or 1 hydrogen atom attaches to the center Cd, CO, and Os atom. For the left second-level nodes, Ct-H, and Cb-H, we believe that those specifications are detailed enough for kinetics searching, and stopped at this level; therefore, they exist as leaves of this tree. From this point, the author will only focus on the sub-tree rooted at Cs-H to illustrate the hierarchy concept, and all other branches are constructed following the similar idea. After Cs-H is partitioned into four children based on the number of hydrogen atoms connected to Cs, we can further subcategorize those four third-level children using the nature of the non-hydrogen atoms. If the non-hydrogen atom is associated with a multiple bond for e.g., Cd, Ct, CO, Cb, then the resulting radical after hydrogen abstraction will be stabilized via delocalization and that will introduce variation in the barrier height. Consequently, the third-level nodes can be subcategorized as primary carbon with non-delocalized substituent and primary carbon with delocalized substituent. For example, the third-level node as shown in Figure 6.4, Cs with two hydrogen and two non-hydrogen, can be divided into four children according to the properties of the two non-hydrogen substituents: the first child for two non-delocalized carbon substituents, the second child for two non-delocalized substituents with at least one oxygen, the third child for one non-delocalized and one delocalized substituent, and the last child for two delocalized substituents. For the fourth level, we didn't subcategorize the first and the last children, and added one level for the second and third ones to distinguish Cs and Os for the non-delocalized substituent. Following similar rule, the R-H reactant is finally categorized into a 5-level-depth tree with totally 50 nodes, where there are 33 leaves. FunctionalGroup defined in each leaf represents a final subgroup of R-H reactants; therefore, R-H is finally subcategorized into 34 subgroups.
For the radical tree, we decided to allow biradicals to participate in H abstraction as well, and therefore, the root of radical tree is a union of mono-radical $R\cdot$, 1-centered biradical $R\cdot\cdot$, 2-adjacent-centered biradical $\cdot R_1-R_2\cdot$, as shown in Figure 6.5. The subtree rooted at $R\cdot$ in radical tree is identical to that of X-H tree, except that the position of H is now being occupied by a radical center. The $R\cdot\cdot$ node only has two children accounting for $\cdot O\cdot$ atom, and $\cdot CH_2$ biradical. The $\cdot R_1-R_2\cdot$ node also has two children for $\cdot O-O\cdot$ biradical and $\cdot C=\cdot C$. Finally, Radical reactant is categorized into a 6-level tree with totally 57 nodes, where there are 38 leaves. Similarly, these 38 leaves mean that the whole radical reactants are divided into 38 subgroups finally.

Comparing to Curran’s 1-step subcategorization for the H abstraction reactions, which divided R-H and Radical into 4 and 11 groups, respectively, our hierarchy tree method easily and quickly subdivide H abstraction reactants into 34 R-H subgroups and 38 Radical subgroups. This is because hierarchy tree method localizes the subcategorizing action at a small part of tree, and therefore it is a better, more systematic subcategorization method. Another advantage of the tree method is that if later we need to add more partitions, we can simply subcategorize the tree leaf that needs more dividing, and other parts of the tree will remain intact.

Similar to the H abstraction reaction family, we have successfully built the hierarchy trees for all the reaction families for gas-phase pyrolysis and oxidation reaction systems, shown in Table 6.1. In this part, Dr. S. Raman has contributed a lot of efforts on building all the original hierarchy trees, C. D. Wijaya helped build trees for some families, and the author also helped test the correctness of the hierarchy relations for all the trees.
Figure 6.4 Part of R-H tree (constructed by S. Raman and C. D. Wijaya)

(1) The symbols used in this figure have similar meanings as the FunctionalGroup element defined in Table 4.2.

(2) The atom(s) and bond(s) in red color represent the center(s) of a FunctionalGroup.

(3) The dash lines below many tree nodes indicate the existence of a sub-tree below that node. The author doesn’t show the complete tree, where there are totally 50 nodes.
(1) The symbols used in this figure have similar meanings as the FunctionalGroup element defined in Table 4.2.

(2) The atom(s) and bond(s) in red color represent the center(s) of a FunctionalGroup in kinetic tree.

(3) The dash lines below many tree nodes indicate the existence of a sub-tree below that node. The author doesn't show the complete tree, where there are totally 57 nodes.

(4) The radical tree is similar to R-H tree. The subtree rooted at $R^\bullet$ node has exactly the same structure as the R-H tree with one radical center substituting -H group.

(5) We added two more radical nodes at the second levels, $R^\bullet\bullet$ and $R_1^\bullet-R_2^\bullet$ to allow biradical H abstraction reaction. $R^\bullet\bullet$ represents 1-centered biradical, like $\cdot O\cdot$; $R_1^\bullet-R_2^\bullet$ represents the 2-centered biradical, whose radical centered at adjacent position, like $\cdot O-O\cdot$, when we treat Oxygen as radical.
6.4.3.2 Building hierarchy tree for kinetics database

After the hierarchy tree(s) for a reaction family have been built, the next step is to build our kinetics library from the literature kinetics data.

Collecting and format kinetics rate rule from literature

To manage kinetics rate rules, we firstly prepared all the possibly available kinetics rate rules by collecting and reviewing the kinetic data from literature [Atkinson et al., 1997a and 1997b; Baulch et al., 1994 and 1992; Tsang, 1991, 1990, 1988, and 1987; Tsang and Hampson, 1986], and also by doing quantum calculations for some of the reaction families [Sumathi et al., 2001a, 2001b, and 2002; Sumathi and Green, 2002a; Wijaya et al., 2003; Green et al., 2004]. As reviewed in section 5.1, kinetics parameter is a set of temperature-independent constants. Here we used two popular kinetics formats, the Modified Arrhenius kinetics format with kinetics parameter set of $A$, $n$, and $E$, and Modified Arrhenius combined with Evans-Polanyi relation with kinetics parameter set of $A$, $n$, $\alpha$, and $E_0$.

Beside the kinetics itself, we also include uncertainties $\Delta A$, $\Delta n$, $\Delta \alpha$, and $\Delta E_0$ for $A$, $n$, $\alpha$, and $E_0$, respectively. There are two types of uncertainties, adder uncertainty, which means that the uncertainty value should be added or subtracted from the nominal value to get the valid range, and multiplier uncertainty, which means the uncertainty value should be multiplied or divided to the nominal value. To distinguish them, we add a prefix, $\ast$, to the multiplier uncertainty value.

To store the source and quality of kinetics parameters, we assigned a rank number to each kinetics rate rule. C. D. Wijaya made the rules for ranking the kinetics. Here, the author quotes the description of the ranking rules in her thesis to illustrate the meaning of kinetics rank: "The quality rank number ranges from 1 to 5, with the 1 being most reliable. Rank 1 is assigned to rate rules parameters that are in good agreement with several literature data. Parameters obtained from direct measurements in experiments, or from high-level quantum chemistry calculations that are in good agreement with experimental results are ranked 2. Results from direct measurements, but at very limited temperature
and pressure range are ranked 3. When using results from experiments, efforts were made to take only the rates at or near the high-pressure limits. Rate rules from high level quantum chemistry calculations, but without comparison with experimental results are also ranked 3. Rank 4 is for data from indirect measurements, or extensive literature review. Finally, rank 5 is for rate parameters estimated using a big mechanism to fit experimental data, usually in a limited temperature and pressure range. A default value is assigned to the top level of the rate rules tree of each reaction family. This is a very rough estimation of what the order of magnitude of the reaction should be, and is ranked 10.

The total number of the rate rules we collected is totally about ~1000. Sometimes multiple rate constants are available; for such case, we used the data that came from the most recent work, employed the most reliable method in deriving the rate parameters, and covered the widest range of temperature, and stored them in the order of priority in the kinetics library, but only the first data of the multiple values will be chosen to generate reaction mechanism in RMG. Users should pay attention to this rule when they add new rate rules.

Assign best-matched kinetics FunctionalGroup keys to kinetics rate rule

In the second step, we assign the proper subcategorization FunctionalGroup keys for each rate rule. For example, in the H abstraction reaction family, Curran’s rate rules provide that for subfamily of Secondary H + •OH, the modified Arrhenius parameters are $A = 9.00 \times 10^5$ (cm$^3$/mol/sec), $n = 2.0$, and $E = -1.133$ (kcal/mole). We then draw the FunctionalGroup graphs for those two reactants, which indicate a subgroup of H abstraction family that above kinetics rate rule covers. However, such representation may not already exist in our hierarchy tree structure, and we have to find the best-matched FunctionalGroup representation in our R-H tree and radical tree for this subfamily. Now we do this match by hand, i.e., we investigate all the literature kinetics and assign a proper FunctionalGroup in our hierarchy trees to each reactant. In this case, we assigned (C-H)/H/Cs/Cs to the secondary H and •O-H to the •OH radical, respectively, as shown in Figure 6.6, which finally become the index keys assigned to
this literature kinetics rate rule. We call those index keys for rate rule kinetics FunctionalGroup keys.

![FunctionalGroup keys of one subfamily of H abstraction: secondary H + •OH](image)

**Figure 6.6 FunctionalGroup keys of one subfamily of H abstraction: secondary H + •OH**

The author here proposes an automatic way to find out the best-matched kinetics FunctionalGroup keys to replace the manual key assignment procedure. In the new approach, once a new kinetics rate rule is published, people only need to draw the FunctionalGroup graphs representing the group of reactions that the new kinetics rate rule refers to, and the computer can automatically match those FunctionalGroups for the new rate rules with the FunctionalGroups defined in kinetics trees of the proper reaction family, and find the best-matched keys automatically. In this part, the matching algorithm introduced in chapter 3 will be used again to automatically identify the subgraph relationships between ChemGraph and FunctionalGroups or two FunctionalGroups. This feature has not been implemented in RMG, but it is very straightforward to add a new method to do so.

**Build kinetics rate rule library**

After the kinetics FunctionalGroup keys are found for the kinetics, we can then put in the kinetics rate rule into our kinetics library indexed by the kinetics FunctionalGroup keys. Consequently, our kinetics rate rule library for each reaction family is a one-to-one mapping of FunctionalGroup keys from kinetics trees to the rate rule. It is noteworthy that we maintain an individual kinetics library for each reaction family to avoid accidental messing-up of the similar kinetics FunctionalGroup keys from different reaction families.
In total, we collected ~1000 kinetics rate rules from literature, calculations, and experiments, and filled them into their corresponding family’s kinetics rate rule library. Part of the H abstraction rate library is shown in Table 6.2 to illustrate the standard format of rate rule library. The first column is just a counting number, the second and third columns store the FunctionalGroup keys for this kinetics rate rules. Fourth column is the temperature range that the kinetics can be applied. From fifth column to twelfth column, we record the kinetic parameters and their uncertainties. For the ones with unknown uncertainties, we simply put zeros. The last column is the rank of that kinetics rate rules. Besides the kinetics rate rules, we also recorded all the source of the rate rules at the end of the rate rule table. Because of the limitation of space, author doesn’t include the full documentation of our kinetics library constructed by C.D.Wijaya, and users interested in reviewing all the rate rules may refer to her thesis.

<table>
<thead>
<tr>
<th>No.</th>
<th>X_H</th>
<th>Y_rad_birad</th>
<th>Temp</th>
<th>A</th>
<th>N</th>
<th>α</th>
<th>E₀</th>
<th>ΔA</th>
<th>Δn</th>
<th>Δα</th>
<th>ΔE₀</th>
<th>rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X_H</td>
<td>Y_rad_birad</td>
<td>300-1500</td>
<td>1.0 E+05</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>X_H</td>
<td>H_rad</td>
<td>300-1500</td>
<td>2.4 E+08</td>
<td>1.5</td>
<td>0.65</td>
<td>9.4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>X_H</td>
<td>O_atom_trip</td>
<td>300-1500</td>
<td>1.7 E+08</td>
<td>1.5</td>
<td>0.75</td>
<td>6.6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>X_H</td>
<td>O_pri_rad</td>
<td>300-1500</td>
<td>1.2 E+06</td>
<td>2.0</td>
<td>0.50</td>
<td>10.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>X_H</td>
<td>O_sec_rad</td>
<td>300-1500</td>
<td>1.4 E+04</td>
<td>2.69</td>
<td>0.60</td>
<td>11.3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

**Table 6.2 Part of the rate rule library documentation**

6.4.3.3 Searching proper kinetics for given elementary reaction

The numbers of available rate rules in different families vary much, from ~200 for the well-studied reaction families like H abstraction and radical addition to multiple bond, to ~10 for the not well-known reaction families like intra-addition to form an exo(endo)-cyclic radical.

After the kinetics library for reaction families have been built, we need to search through the library to find the best kinetics for a given elementary reaction. Typically, this approach has two steps: firstly find out the best-matched kinetics FunctionalGroup keys, secondly we need to retrieve the kinetics rate using FunctionalGroup keys. This
procedure is very similar to filling in kinetics rate rules in library discussed in last section, except that in kinetics searching, we are obtaining kinetics from the library instead of assigning the proper rate rules.

The author uses an example from, again, H abstraction family, to illustrate this process. For the H abstraction reaction:

![H abstraction reaction diagram]

We first compare the reactant C(H)$_2$(CH$_3$)$_2$ to the H abstraction R-H tree. The searching procedure has been described in Chapter 3, and the matched path is: (R-H) -> (Cs-H) -> (Cs-H)(C/O)$_2$(H) -> (Cs-H)(Cs)$_2$(H). For the radical reactant, the matched path is: (Radical) -> (R•) -> (O•) -> (O•)-H. Finally, two best matched FunctionalGroup keys: (Cs-H)(Cs)$_2$(H) and (O•)-H, whose graph representations are shown in Figure 6.6, are found for this reaction. Using these keys to access the H abstraction library gives the user the proper kinetics rate for this reaction: $A = 9.00 \times 10^5$ (mole/cm$^3$/sec), $n = 2.0$, and $E = -1.133$ (kcal/mole).

Finally, the author would like to point out the advantage of using the hierarchy tree structure for kinetics searching, which is another reason that we construct kinetics hierarchy tree for reaction family. Imagine that we store all H abstraction kinetics rate rules with their kinetics FunctionalGroup keys into an unordered List; when we search kinetics for above reaction, we need to compare the reactants with the kinetics FunctionalGroup keys of all the rate rules until we find out the best-matched rate rule, which the worst-case running time is $O(N)$, where $N$ is the totally number of rate rules in H abstraction, in H abstraction, $N \approx 200$. For our procedure, we search through the R-H and radical trees, which will take $O(D)$, where $D$ is the largest depth of tree; in H abstraction case, $D = 6$. It is obvious that hierarchy tree method provides a faster
kinetics search method. This is quite similar to the thermal group tree search introduced earlier in last Chapter.

6.4.3.4 Obtaining approximate kinetics

This section tries to solve a problem possibly happening in RMG: what if there is no rate rule found for the best-matched FunctionalGroup keys in our library? This problem is quite common since we only have totally about 1000 rate rules, which obviously can’t cover all possible subfamilies of each reaction family. For example, in last section, we subcategorize R-H and Radical reactants of H abstraction into 34 and 38 subgroups, respectively; therefore, we need 34×38=1292 rate rules to completely cover all the H abstraction subfamilies. In fact, although H abstraction is a well-studied reaction family, there are only ~200 rate rules available, which means ~80% of the H abstraction subfamilies have unknown rate rules. Meanwhile, we allow reaction families generate all possible reactions from given species, so it is very possible that some reactions falling in the subfamily with unknown kinetics rate rule. This problem partly results from the fact that there are not enough kinetics rate rules in literature, and people surely will see the rate rule library become more and more complete with new rate rules published in the future. However, we, right now, need a feasible approximate rule dealing with missing kinetics rate rules in RMG.

The most straightforward way dealing this problem is to assign approximate kinetics rate rules for all the subgroups of a reaction family, for example, for H abstraction, we need to handle ~1000 subgroups missing rate rules. It seems impossible to do it by hand, and we need a good algorithm for this job. Such algorithm needs to be able to identify the similarity of two subgroups, so that if one misses a rate rule, we can use the other’s to approximate it. However, to identify the similarity of two subgroups is not easy, especially for the bi-molecular reaction families. Therefore, this method was not used in RMG, but the author still suggests and encourages people work on this way to solve this problem.

The author developed another approach to provide an algorithm to look for approximate substitute kinetics from a larger subgroup. For example, for H abstraction reaction family, assume that one reaction’s best-matched FunctionalGroup keys are (Cs-
H)(Cs)\textsubscript{2}(H) and (O\bullet)-H, and there is no exact rate rule for this subgroup in the library, we then can move one level up to their parents, (Cs-H)(C/O)\textsubscript{2}(H) and (O\bullet). It is possible that there is a kinetics rate rule for the combination of one leaf and other leaf’s parent, for example, (Cs-H)(C/O)\textsubscript{2}(H) + (O\bullet)-H subgroup might have a rate rule in library. If this is true, we will use this rate rule approximate the original subgroup of (Cs-H)(Cs)\textsubscript{2}(H) + (O\bullet)-H. If we don’t find rate rules at the parent level, we will keep moving up until we reach the root of kinetics tree. A default root kinetics rate rule has been assigned to every reaction family in our library to guarantee at least a raw rate rule for every reaction. In order to measure how good an approximation is, we assign a parameter, distance, which is calculated by: distance = \sum d_i, where d_i indicates how many levels between the exact kinetics FunctionalGroup key and the kinetics FunctionalGroup key for the rate rule selected in the i\textsuperscript{th} kinetics tree. Therefore, the larger the distance is, the worst the approximation is. We generally pick the approximate rate rules with the smallest distance; sometimes multiple approximate rate rules with the same distance could be found, and in such situation, we use the average kinetics of those multiple rate rules.

During applying this algorithm in RMG, the author found a fact that a large number of reactions take the raw root values. That is because that many kinetics libraries haven too many blanks for the intermediate subgroups, so that it is very easy for a reaction move all the way above to the top to find the final approximation. For the intermediate subgroup, the author means the subgroup represented by at least one non-leaf FunctionalGroup key. Can we do something better here? The author developed another algorithm to fill in the intermediate subgroups trying to stop searching approximate kinetics as early as possible. Shown below are two schematic parent-children levels from two kinetics trees.
FunctionalGroup keys from P₁ and P₂ will form an intermediate subgroup. Assume that we don’t have the kinetics rate rule for this subgroup. But we have some rate rules from the subgroups defined by P₁ and P₂’s children. Assume that we find the rate rules of (C₁₁ + C₂₁), (C₁₁ + C₂₁), (C₁₁ + C₂₁), and (C₁₁ + C₂₁) subgroups, we can then average those rate rules and assign the result to the intermediate subgroup (P₁ + P₂). This algorithm comes from the fact that (P₁ + P₂) intermediate subgroup is actually the union of all their children subgroup; if some of the children subgroups rate rules are available, we use those rates estimate the whole (P₁ + P₂) subgroup. This surely is a raw approximation with errors, but it provides a better approximation than the root default rate rule. As a result, with this algorithm added, the kinetics searching procedure in RMG now has improved its kinetics approximation greatly.

6.5 Design of rxn package in RMG

Rxn package in RMG models the process of generating completely all possible reactions from a given set of species, following the reacting rules defined in a reaction family. Therefore, the major duty of rxn package is to implement Reaction, ReactionTemplate, and ReactionGenerator. Reaction describes the fundamental features of chemical reaction; ReactionTemplate models reaction families; ReactionGenerator is to generate any elementary reactions given a set of species. The UML object diagrams describing Reaction, ReactionTemplate, and ReactionGenerator, are shown in Figure 6.7, 6.8, and 6.9, respectively.

6.5.1 Reaction object diagram

In Reaction object diagram, Figure 6.7, we defined the basic objects for modeling general chemical reactions. Reaction class has two major compositions, a Structure object and a RateConstant object. In RMG, Structure class manages direction, reactants, and products of a reaction. RateConstant specifies the kinetics rate information of a reaction.
6.5.1.1 Structure

Structure class defines the reactants and products of a reaction, and since non-elementary reaction is allowed, the size of the reactants and products list are not fixed; instead, the LinkedList, able to store a length-varied list of objects, is used to store any number of reactants and products. The chemical objects stored in a structure’s reactant and product list are ChemGraph objects, not Species objects, since we need to know the detailed structure of the resonance isomer participating in the reaction.

Structure has an important feature, direction, which indicates if the reaction is a forward reaction or a backward one. This feature is essentially important later when we deal with the thermodynamic consistency issue in kinetics estimation.

Another important feature of Structure is redundancy. Redundancy represents the number of equivalent reacting sites for the same reaction. For example, H abstraction...
reaction, \( \text{CH}_4 + \text{H}\cdot \rightarrow \cdot\text{CH}_3 + \text{H}_2 \) has a redundancy of 4, since there are four equivalent Hydrogen atoms in methane to be abstracted by a radical. In our rate rules library, H abstraction rate rules are all regulated to the abstraction rate per hydrogen atom, so the real kinetics for an H abstraction will be its rate rule from the library times its redundancy. To calculate reaction structure redundancy is a tricky problem. In RMG, we simply counted the number of equivalent reaction structures from the same reactants, and test showed this method gives right redundancy number.

Structure class provides useful methods for managing reactants and products information and calculating reaction thermodynamics from reactants and products. It is also noteworthy how the equals() method determines if two structure instances are equivalent. In RMG, two reaction structures are considered the same if they are the same direction, and they have equivalent reactants list and products list, no matter how the reactants(products) are ordered in the list. For example, \( \text{A} + \text{B} \rightarrow \text{C} + \text{D} \), is equal to the reaction of \( \text{B} + \text{A} \rightarrow \text{D} + \text{C} \).

6.5.1.2 RateConstant

Before we discuss RateConstant class, let us first take a look at several important kinetics-related classes, Kinetics, ArrheniusKinetics, ArrheniusEPKinetics, and KineticsTemplate.

Kinetics in RMG is designed as an interface, which requires its subclasses implement a set of methods, for example, calculateRate(). Implementing Kinetics interface, ArrheniusKinetics class manages modified Arrhenius parameter set of \( A \), \( n \), and \( E \). In order to also include the Evans-Polanyi relation, the author added a subclass of ArrheniusKinetics, ArrheniusEPKinetics, where only one more parameter \( \alpha \) is added. Both classes are able to return the kinetics parameters and the rate given the proper temperature (and also \( \Delta H_{\text{rxn}} \) for Evans-Polanyi expression). It is noteworthy that in handling kinetics parameters, RMG includes uncertainty information for them. In RMG, kinetics parameters are stored as UncertainDouble, a class for storing uncertain double value, which has a nominal value, an uncertainty value, and an uncertainty type that is either an adder uncertainty or a multiplier uncertainty. For, prefactor \( A \), we always use the uncertainty type of multiplier so that \( \log A \) will have an even uncertainty range around
nominal point. For other kinetics parameters, we use adder uncertainty. With this method, we can read in and store the uncertainty information together with kinetics from our rate rule library, partly shown in Table 6.2. Later if people are willing to add uncertainty analysis on kinetics into RMG, they can make use of those UncertainDouble instances of kinetics parameters for further calculations.

KineticsTemplate uniquely represents a kinetics rate rule that a group of reactions share, i.e., it reflects the one-to-one mapping relation between a kinetics rate rule and its applicable reactions represented by FunctionalGroup keys. Therefore, KineticsTemplate class includes two compositions, one Kinetics rate rule and the set of kinetics FunctionalGroup keys that describe the reaction subfamily that Kinetics rate rule accounts for. This class is very important to form the kinetics library for reaction families we have discussed in section 6.4.3.2.

RateConstant class holds the proper rate rule for a reaction. It stores one KineticsTemplate instance or a set of KineticsTemplate instances. As we have discussed in section 6.4.3, it is possible that, for an individual reaction, we might either find out a best-matched kinetics rate rule or a set of possibly second-best-matched kinetics rate rules from the library. RateConstant class provides storage room for both cases. If the best-matched kinetics is found, that will be always stored and returned as the kinetics for this reaction; if only a set of approximate kinetics are found, RateConstant will store them all, and calculate and return the average of those approximate rate rules as this reaction’s kinetics parameters.

6.5.1.3 Reaction class

Reaction class in RMG represents chemical reaction in a general meaning, and an instance of a Reaction can be elementary reaction or non-elementary reaction. Reaction class has two compositions, a Structure and a RateConstant. The Structure class and RateConstant have been introduced above, and Reaction class simply wraps them together and is responsible for properly handling and passing the corresponding information from Structure and RateConstant to users.

A very important function of Reaction class is to deal with the thermodynamic consistency, discussed in section 6.4.2. To handle the kinetics for the backward reaction,
the author added another attribute of Reaction, reverseReaction. For a backward reaction, its reverseReaction points to its corresponding forward reaction, and when calculating the kinetics of this backward reaction, RMG will access to its reverseReaction’s rate constant to calculate the forward reaction’s kinetics, and then combine it with the Keq parameter to provide the thermodynamic-consistent kinetics for the backward reaction. For a forward reaction, it uses the kinetics calculated from its own RateConstant attribute.

6.5.2 **ReactionTemplate object diagram**

ReactionTemplate object diagram, shown in Figure 6.8, describes the design of reaction family in RMG. This diagram mainly includes: StructureTemplate class defining reactants structure and kinetics FunctionalGroup tree; ReactionAdjList class specifying the reacting actions; KineticsTemplateLibrary class managing the kinetics rate rule library; combining those above three objects forms ReactionTemplate class. In this diagram, we also define TemplateReaction class, a subclass of Reaction, describing the elementary reactions generated from a reaction family.
6.5.2.1 StructureTemplate class

StructureTemplate class is the core to identify the possible reactants for a reaction family and to search for the best-matched kinetics FunctionalGroup keys for a reaction; in another word, it governs all the structure-related test of a reaction family. In detail, it stores the FunctionalGroups describing valid reactants and kinetics hierarchy trees subcategorizing a whole reaction family into subfamilies.

To record reactant candidate pattern, it has a LinkedList attribute, reactants, storing the FunctionalGroup description of the valid candidate of a reaction family. For example, in H abstraction reaction family, its StructureTemplate stores two FunctionalGroup representations for R-H and Radical as its reactants. Moreover, this class also provides a method of identifyReactedSites(ChemGraph) to identify all the valid reacted sites of a passed-in ChemGraph.
StructureTemplate also maintains hierarchy trees subcategorizing the reactants. For example, for the H abstraction, two hierarchy trees shown in Figure 6.4 and 6.5 will be read in and stored in this class. It also provides a searching method `getMatchedFunctionalGroup(reactant list)` to find out the best-matched kinetics FunctionalGroup keys for a passed-in reactant list.

Another important functionality of StructureTemplate is to generate its reverse StructureTemplate. This will be used later in generating reverse reaction template to satisfy thermodynamic consistency, and the details will be given in the ReactionTemplate class section.

6.5.2.2 ReactionAdjList class

ReactionAdjList class takes care of making reactions. In detail, it stores a list of reacting actions for a reaction template, so-called “reaction recipe” in RMG. As we have introduced earlier, there are five types of reacting actions implemented in RMG, `break_bond`, `form_bond`, `change_bond`, `gain_radical`, and `lose_radical`. Those actions are modeled by `Action` class, which simply stores information about type, reactive sites, and reacting parameters of an action.

Besides storing a list of actions, ReactionAdjList can apply those actions to the valid reactant candidates to produce products. The method `reactChemGraph()` takes in the list of reactants, mutates the reactants’ graphs according to its action list, and finally makes and returns a list of new products.

ReactionAdjList class is also able to generate its reverse ReactionAdjList, which is again used for generating reverse reaction template.

6.5.2.3 KineticsTemplateLibrary class

KineticsTemplateLibrary class stores all the kinetic rate rules of a reaction family. The main attribute of this class is a `HashMap` of instances of `KineticsTemplate`. `HashMap` is a type of collection implementing key-to-object mapping functionality in Java. As introduced earlier, `KineticsTemplate` class stores a kinetics rate rule and a set of kinetics FunctionalGroup keys. KineticsTemplateLibrary then uses the set of kinetics
FunctionalGroups keys of a KineticsTemplate instance as the key to store this KineticsTemplate instance, and also to search for a proper kinetics rate rule.

6.5.2.4 ReactionTemplate class

ReactionTemplate class models the chemical reaction family. The instances of ReactionTemplate could be H abstraction reaction template, beta scission reaction template, etc.

ReactionTemplate includes three important compositions, a StructureTemplate, a ReactionAdjList, and a KineticsTemplateLibrary. ReactionTemplate assembles those three classes together to implement reaction generation, i.e., making a set of possible reactions from given reactants. The process of making reactions involves three steps: (1) passing the species to StructureTemplate to identify all the reacted sites; (2) passing the valid reacted sites to ReactionAdjList to make products; (3) obtaining the rate rule from KineticsTemplateLibrary for the generated reaction.

ReactionTemplate also has two informative reaction dictionaries, one indexed by the reactants, the other indexed by reaction structure. The reaction dictionaries record all the reactions generated from this ReactionTemplate for two purposes: firstly it provides a quick search of the same family of reactions from its reactants and structure, and secondly it avoids generating multiple instances for the same reaction.

Next important issue in ReactionTemplate class is to deal with the relation between forward and backward reaction templates. In RMG, we define independently only the forward reaction template. For the backward reaction template, RMG derives the StructureTemplate from that of the forward reaction template to guarantee the symmetry between the structure of forward and backward reaction template. If we allow both the forward and the backward reaction template to each have its own independent StructureTemplate, it is possible that a reaction generated by backward reaction template will not have its symmetric forward reaction to provide proper kinetics. RMG also derives the ReactionAdjList of backward reaction template from that of the forward reaction templates, since they are naturally opposite to each other. For the KineticsTemplateLibrary, the backward reaction template will simply share the one for
its forward reaction template. To implement such dependence, in initializing a forward reaction template, RMG adds an extra step to create its backward reaction template.

6.5.2.5 TemplateReaction class

TemplateReaction, a subclass of Reaction, represents the elementary reactions generated by a reaction template. TemplateReaction class inherits the common features and functions defined in Reaction, and besides those, it has extra information about its reaction template. For example, for a template reaction \( \text{H} + \text{CH}_4 \rightarrow \text{H}_2 + \text{CH}_3 \), this reaction object also has a pointer linking to the H abstraction reaction template. The reaction template information will tell users where the reaction and its kinetics are coming from, and in addition, the creation of TemplateReaction, controlled by its factory method, makeTemplateReaction() function, is also greatly depending on the reaction template behavior. (For details on factory method, refer to Gamma, et al., 1995)

6.5.3 ReactionGenerator

The ReactionGenerator object diagram, shown in Figure 6.9, describes the procedure by which RMG generates all possible reactions from a given set of species. There are two types of ReactionGenerator in RMG, TemplateReactionGenerator that generates all the template reactions by existing reaction templates in RMG, and LibraryReactionGenerator that searches for reactions in a library. Both of them implement an interface called ReactionGenerator, which indicates that all types of reaction generators are required to implement a method called react(), which takes in a set of species and returns a set of reactions. People can later add their own type of ReactionGenerator if they have other methods for generating reactions from species.
Figure 6.9 UML diagram for ReactionGenerator
6.5.3.1 TemplateReactionGenerator

TemplateReactionGenerator class generates reactions from reaction templates. It has an attribute of the TemplateReactionLibrary, which collects and manages all the reaction templates in RMG.

In the key method of react(given species set) of TemplateReactionGenerator, every species in the given species set will be passed to all unimolecular reaction templates, and every combinations of two species from the given species set will be passed to all bi-molecular reaction templates. The returned template reactions from all the reaction templates will be united and returned as one reaction set.

6.5.3.2 LibraryReactionGenerator

LibraryReactionGenerator class searches qualified reactions from a reaction library. A reaction from library is modeled by LibraryReaction class, another subclass of Reaction, representing special reactions that cannot be generated by reaction templates, for examples, some smaller molecule oxidation reaction, like $O_2 + CO = CO_2 + O$. ReactionLibrary class is a singleton, collecting all the important library reactions of a reaction system.

In the key method of react(given species set) of LibraryReactionGenerator, every library reaction stored in ReactionLibrary will be compared with the species in the given species set checking if a given species participates in that library reaction, and if it is true, that reaction will be added to the returned reaction set.

6.6 Conclusions and discussions

In this section, the design ideas and implementation details on generating reactions from reaction families are presented. With an in-depth understanding of the reaction family, we proposed a new data-model based method for identifying reactant candidates and for making reactions, which provide users a more flexible, extendable way to define and modify reaction families. We also proposed a new hierarchy tree method to manage kinetics databases, which can systematically accommodate a large number of the kinetics data, can be quickly searched, and can be easily extended with the new rate rules.
We also put a lot of efforts into building the full documents, including FunctionalGroup definitions, reacting actions, kinetics hierarchy trees, and rate rule libraries, for all 18 forward reaction families. This is the richest reaction family set among all the automatic model generation software. Moreover, since we paid much attention to flexibility and extensibility issues, it will be easy to modify and extend the current reaction templates and to add any new reaction family and/or kinetics rate rules.

The author also successfully implemented RMG-rxn package to model important reaction-related objects, like Reaction, ReactionTemplate, ReactionGenerator, etc. The most important functionality that this package provided is to generate a complete set of reactions from a given set of species by system reaction templates, which provides a fundamental and very flexible reaction generator tool for building detailed reaction mechanisms.
Appendix 6.1: Documentation of Reaction families

By Sumathy Raman and Jing Song

0. **Notations used in this documentation**

**Bond Notation:**

1. Single bond: (S) -
2. Double bond: (D) =
3. Triple bond: (T) ≡
4. Benzene bond: Θ
5. Bond (need specification for different case): ↔
6. Any bond (all possible bond including single, double, triple, benzene, CO, COm, O₂): ~
7. Multiple bond (double, triple, benzene, COm, CO, O₂): ^
8. One more bond formed: bond(+1)
9. One less bond formed: bond(-1)

**Atom Notation:**

1. Cs
   \[ \text{C} - \text{C} \]

2. Cd
   \[ \text{C} \equiv \text{C} \]

3. Ct
   \[ \text{C} \equiv \text{C} \]

4. Ca
   \[ \text{C} \equiv \text{C} \equiv \text{C} \]
5. Cb

6. Cbf

7. CR₂₅

8. CR₂₇

9. Oₐ

10. COₘ

11. Ck

Special molecule Library: 1~4 are biradicals, 5 is a triradical

1. O₂: Oxygen molecule
   (1) O₂ᵇ: treated like 2-adjacent-centered biradical, •O•O•, which is more stable.
   (2) O₂ᵈ: treated like a normal double bond, O=O.

2. Oₐ: Oxygen atom, treated like 1-centered biradical, and it can be involved in abstraction and addition rxn.
   (1) Oₐᵗ: treated like 1-centered biradical with high spin, called triplet. (two orbitals, the same spin direction, more stable)
   (2) Oₐˢ: treated like 1-centered biradical with low spin, called singlet. (two orbitals, the opposite spin direction)
3. **CO**: Carbon monoxide molecule, treated like a 1-centered biradical, and it can be involved in addition and insertion rxn. (It is a singlet)

   ![C=O](image)

4. **CCH**: Vinylidene molecule, treated like a 1-centered biradical

   (1) **CCH**: treated like 1-centered biradical with high spin, called triplet. (two orbitals, the same spin direction, more stable)

   ![C=CH₂](image)

   (2) **CCH**: treated like 1-centered biradical with low spin, called singlet. (two orbitals, the opposite spin direction)

   ![C=CH₂](image)

   (Note: we didn’t include CCR₂C in our system.)

5. **CH₄**: Carbenic radical, treated like a triradical

   ![C-H](image)

6. **C₂**: treated as 2-centered biradical, •C≡C•

7. **CO₂**: O=C=O, stable molecule with 2 double bonds

**Radical categorization**

1. Mono-radical: (# of radical =1)
2. Bi-radical: (# of radical =2)
   a. 1-centered: CH₂C, CO₉, O₉
   b. 2-centered:
      i. adjacent center: O₂, C₂
      ii. far-away center: normal biradical
3. Tri-radical: (# of radical =3) CH₄ (this is the only triradical considered)
1. **Intermolecular hydrogen abstraction reaction**

\[ X-H + Y\bullet -> X\bullet + Y-H \]

Restriction:

1. X-H and Y\bullet don’t belong to the same molecule.
2. X-H can be any one of the H-H, C-H, and O-H type. Example: carbon can be Cs, Cd, Ct, Cb, and CO; oxygen can be O.
3. H can be any hydrogen from H-containing groups.
4. Y\bullet can be any mono-radical and 3 special biradical: O\textsubscript{ats}, CO\textsubscript{m}, and O\textsubscript{2b}. With O\textsubscript{2b}, the resulting radical should be centered at the adjacent oxygen.
5. X-H should be molecule, or 3 mono radical: •O-H, H•C=O, •O-O-H, corresponding to the reverse reaction of the 3 special cases for Y\bullet in forward direction.

2. **Radical addition to multiple bonds**

\[ Y\bullet + X^\wedge Z -> Y-X^\wedge(-1) Z\bullet \]

\[ Y\bullet + X^\wedge Z -> Y-Z^\wedge(-1) X\bullet \]

Note: we should make this addition at both sites of the multiple bond.

Restriction:

1. X^\wedge Z can be Cd=Cd, Ct=Ct, Ck=Cd, Ca=Cd, Ca=Ca, Ck=Ck, Ck=Ck, Cb Cb Cbf Cbf, C=O, CO\textsubscript{2}, and O\textsubscript{2d}; it also means ^, the bond between X and Z, could be double, triple, benzene, carbonyl, O\textsubscript{2d}, CO\textsubscript{2}. X^\wedge Z cannot be CO\textsubscript{m}, if it is refer to radical addition to CO\textsubscript{m}.
2. There must be two reactants.
3. Y\bullet can be any radical, except special radicals like CH\textsubscript{2s}, O\textsubscript{ats}, O\textsubscript{2b}, and CO\textsubscript{m}.

Prohibition: if the multiple bonds is O\textsubscript{2d}, then O\textsubscript{2b}, ROO•, and HOO• are forbidden.
3. **β-scission**

\[ Z\leftrightarrow X-Y \rightarrow Z\leftrightarrow (+1)X \rightarrow Y\cdot \]

Restriction:

(1) There is one reactant.

(2) \( Z \) and \( X \) can be Cs, Cd, Cb, Cbf, O, and CO; \( Y \) should be H, Cs, Cd, Ct, Cb, O, and CO.

(3) \( Z\cdot \) can be any radical center except biradical.

(4) \( Z\leftrightarrow X \) can be single/double/benzene bonds, (Note, O•-O-Y and O•-CO-Y are allowed), and \( \leftrightarrow \) cannot be triple bond.

(5) X-Y cannot be in a part of a cyclic ring; if it is, refer to the ring opening reaction.

Note: multiple β positions of the radical should be taken into consideration, multiple reactions are possible.

4. **Radical addition to CO\(_m\)**

\[ Y\cdot + CO_m \rightarrow Y\cdot - CO \]

Restriction:

\( Y \) can be any radical including cyclic radical, but not biradical.

5. **CO\(_m\) elimination from carbonyl radical**

\[ Y\cdot - CO \rightarrow Y\cdot + CO_m \]

Restriction:

\( Y \) can be anything including cyclic structure.

6. **Radical recombination**

\[ R\cdot + R'\cdot \rightarrow R-R' \]

Restriction:
(1) $\text{R}\cdot$ and $\text{R}'\cdot$ must be two molecules, and they can be any radical and biradical.

(2) $\text{R}\cdot$ can be $\text{O}_2$, $\text{O}_3$, $\text{CH}_2$, $\text{CO}_m$.

(3) If $\text{R}\cdot$ and $\text{R}'\cdot$ are simple radicals, then the product is a stable molecule; if $\text{R}\cdot$ and $\text{R}'\cdot$ are a simple radical and a biradical, then the product is a radical; if $\text{R}\cdot$ and $\text{R}'\cdot$ are one-centered biradicals, then the product is a stable molecule with a double bond; if $\text{R}\cdot$ and $\text{R}'\cdot$ are a one-centered biradical and a two-centered biradical, then the product is a two-centered biradical.

(4) Prohibition list:
   a. Both $\text{R}\cdot$ and $\text{R}'\cdot$ cannot be $\text{O}_2$ simultaneously.
   b. The two radicals cannot be $\text{R}''\text{OO}$ and/or $\text{HOO}$ simultaneously.

7. **Biradical recombination to form cyclic structure**

   \[
   \text{R}\cdot \sim (\_)_n \sim \text{R}'\cdot \rightarrow \text{R} \sim (\_)_n \sim \text{R}'
   \]

   Restriction:

   (1) There is one biradical reactant.

   (2) Biradical centers are separated from each other by 1, 2, 3, and 4 intervening atoms; i.e., we form up to 6-member ring.

   (3) Two cases are not allowed:
   a. $\text{R}$ and $\text{R}'$ are $\text{OO}$ and $\text{OO}$
   b. $\text{R}$ and $\text{R}'$ are $\text{O}$ and $\text{OO}$

8. **Bond dissociation**

   \[
   \text{R-R}' \rightarrow \text{R} + \text{R}'
   \]

   Restriction:

   (1) $\text{R}$ and $\text{R}'$ must be connected by single bond.
(2) R-R’ can be in cyclic structure, but it cannot be any radical except that the reaction forming O_{2b}, O_{at}, and CH_{2s}.

(3) No biradical is generated from a radical.

9. **Disproportionation**

\[ Y\bullet + R'\sim(XH) \leftrightarrow R''\bullet \rightarrow YH + R'\sim X\leftrightarrow (+1)R'' \]

Restriction:

(1) Y\bullet can be any radical and biradical, including O_{2b}, O_{at}, CH_{2s}, and CO_{m}. When it is a biradical, the resulting YH should be a radical. If Y\bullet is a one-centered biradical, the resulting radical (YH) should be on the same center; if Y\bullet is a two-centered biradical, the resulting radical (YH) should be on the adjacent atom.

(2) X can be Cs, Cd, O, and CO.

(3) R’ and R”\bullet can be anything satisfying the valence of X.

(4) \leftrightarrow can be only single and double. When it is double, XH can have all 7 possibilities, including Cd_sec, shown in tree structure; when it is single, the node Cd_sec is gone.

(5) R’\sim(XH) \leftrightarrow R”\bullet can be a cyclic radical containing a \beta H to the radical center, and R’ can be a radical center.

10. **Molecule Addition**

\[ Y-H + X\leftrightarrow R'' \rightarrow Y\bullet + H-X\leftrightarrow (-1)R'' \]

\[ Y-H + X\leftrightarrow R'' \rightarrow Y\bullet + H-R'' \leftrightarrow (-1)X\bullet \]

Restriction:

(1) \leftrightarrow can’t be single bond, benzyne bond.

(2) Y can be a radical or a biradical, including O_{2b}, O_{at}, CH_{2s}, and CO_{m}.

(3) X and R” can be C_d, C_t, C_b, C_O, CO_{m}, O_{2d}.
11. *Insertion reaction (1,2-insertion)*

\[
\text{CO}_m/\text{CH}_2e/\text{O}_a + \text{R}-\text{R}' \rightarrow \text{R-} /\text{CO}_m/\text{CH}_2e/\text{O}_a/-\text{R}'
\]

Restriction:

(1) R can be Cs, H, Cd, and Cb.

(2) R’ can be Cs, H, OH, OR.

(3) This should be moved to the special reaction family library for CO\textsubscript{m}/CH\textsubscript{2}e/O\textsubscript{a}, if later those reaction families are built.

12. *Small molecular elimination reaction (1,1-elimination)*

\[
\text{R-} /\text{CO}_m/\text{CH}_2e/\text{O}_a/-\text{R}' \rightarrow \text{CO}_m/\text{CH}_2e/\text{O}_a + \text{R}-\text{R}'
\]

Restriction:

(1) R can be H, Cs, Cd, and Cb.

(2) R’ can be Cs, H, OH, OR.

(3) This should be moved to the special reaction family library for CO\textsubscript{m}/CH\textsubscript{2}e/O\textsubscript{a}, if later those reaction families are built.

13. *Insertion reaction (1,3-insertion)*

\[
W-Z + X\leftrightarrow Y \rightarrow W-X\leftrightarrow(-1)Y-Z
\]

\[
(W-Z + X\leftrightarrow Y \rightarrow W-Y\leftrightarrow(-1)X-Z)
\]

Restriction:

(1) Insertion happens in two ways.

(2) \(\leftrightarrow\) can be double, triple, benzene, CO, and CO\textsubscript{2}.

(3) WZ can be H-OH, H-OR, R-OH, and R-H.

(4) If X\leftrightarrow Y is not a C=O or O=C=O, then all the 4 nodes for WZ are possible; if X\leftrightarrow Y is a C=O or O=C=O, only H-R is possible.

(5) All the Rs can be at most 2-Carbon long. (No long molecule addition.)
14. **Small molecular elimination reaction (1,2-elimination)**

\[ W-X \leftrightarrow Y-Z \rightarrow W-Z + X \leftrightarrow (+1)Y \]

Restriction:

1. $\leftrightarrow$ can be single, double bond, and benzene bond
2. X and Y can be Cs, Cd, O, and CO, but X and Y cannot be O simultaneously.
   a. When X is Cs, CO, O, then X-Y is single bond, and Y can be anyone of the four-element set
   b. When X and Y are simultaneously Cd or Cb, then X=Y or $X=\emptyset Y$
3. W and Z can be Cs, O, H, but W and Z cannot be O simultaneously

15. **1+2-cycloaddition**

\[ X=\emptyset + Z:(CR''/Oa) \rightarrow \triangle \]

Restriction:

1. Z: can be carbene or Oas. CR'' is any carbene, which means R” can be H, Cs, or Cd(including vinylidene). In CR”2S, R” can be only methyl, ethyl, iso-propyl, phenyl.
2. The double bond can be Cd=Cd, Cd=Ck, Cd=Ca, Cb=Cb, C=O, and O=O.
3. All reactants are not radicals.

16. **Cleavage of cyclic 3-member ring**

\[ \triangle \rightarrow X=\emptyset + Z:(CR''/Oa) \]
\[ \triangle \rightarrow X=\emptyset + Z:(CR''/Oa) \]
\[ \triangle \rightarrow Y=\emptyset + X:(CR''/Oa) \]

Restriction:

1. All bonds in the 3-member ring should be single bonds.
2. 3-member ring can contains 1 or 2 oxygen atoms.
(3) Reactant is not a radical.

**17. 2+2-cycloaddition**

\[
\begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2}
\end{array}
\end{array} + \begin{array}{c}
\begin{array}{c}
\text{3} \\
\text{4}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2}
\end{array} \\
\begin{array}{c}
\text{3} \\
\text{4}
\end{array}
\end{array}
\]

Restriction:

(1) No reactant is radical.

(2) The double bond can be C\text{d}=\text{C}d, C=O, C\text{d}=\text{C}k, C\text{d}=\text{C}a, and O=O, but the two double bonds cannot be O=O simultaneously, or O=O and C=O simultaneously.

(3) Addition can’t happen at the C\text{k} site. No addition to C=O in C=C=O, just like no addition to O=C=O.

**18. Cleavage of cyclic 4-member ring**

\[
\begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2}
\end{array} \\
\begin{array}{c}
\text{3} \\
\text{4}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2}
\end{array} \\
\begin{array}{c}
\text{3} \\
\text{4}
\end{array}
\end{array}
\]

Restriction: all bonds in the 4-member ring should be single bonds.

**19. Diels-Alder Addition (2+4-cycloaddition)**

\[
\begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2}
\end{array} + \begin{array}{c}
\begin{array}{c}
\text{5} \\
\text{6}
\end{array}
\end{array} \rightarrow \begin{array}{c}
\begin{array}{c}
\text{1} \\
\text{2}
\end{array} \\
\begin{array}{c}
\text{3} \\
\text{4}
\end{array} \\
\begin{array}{c}
\text{5} \\
\text{6}
\end{array}
\end{array}
\]

167
A + B -> C

Restriction:

(1) A contains a double/triple bond, and it can be also molecular oxygen, B contains a delocalized cis-oriented two double bonds. Note: all the multiple bonds should be between Carbons, double/triple/benzene bonds.

(2) A and B can also be part of a ring.

(3) A and B shouldn’t be radical/biradical.

20. Retro-Diels-Alder Addition (Cleavage of cyclic 6-member ene ring)

Restriction:

(1) 6-member carbon ring containing one double bond or two non-conjugated double bond.

(2) 6-member ring containing two adjacent O at position 1 and 2 when double bond is at position 4 and 5. The resulting O₂ will be O₂d.

21. Direct HO₂ elimination (duplicated family 34)

H-R-R'-O-O• -> R=R' +HO₂

Note: this is a special type of disproportionation reaction 6

Restriction:

R and R' must be carbons
22. **Concerted HO₂ addition (duplicated by the reverse of family 34)**

R=R' + HO₂ -> H-R-R'-O-O

R=R' + HO₂ -> H-R'-R-O-O

Note: this is a special type of molecular addition reaction

Restriction:
R and R' must be carbons

23. **Keto-Enol tautomerism**

R-CO-R'-H -> R-C-(OH)=R'

Restriction:
(1) Only 1,2-H migration happens here.
(2) R' should have a H for migration
(3) R can be anything, but R should be connected to CO by a single bond

24. **Enol-Keto tautomerism**

R-C-(OH)=R' -> R-CO-R'-H

Restriction:
(1) Only 1,2-H migration happens here.
(2) R can be anything, but R should be connected to CO by a single bond

25. **Intra-molecular hydrogen migration**

R• ~ ( )₀ ~ R'-H -> H-R ~ ( )₀ ~ R'•

Restriction:
(1) R can be Cs, Cd, Cb, Cbf, O, O-O, CO; if R• is centered at Cb or Cbf, the migrating H should come from the substitution on the adjacent carbon.
(2) R' can be any groups containing H; if R’ is a Cb or Cbf, then H should migrate to the adjacent position on the aromatic ring.
(3) \( n \) can be any number from 0 to 3.

(4) \((\_\_\_\_\_\_\_\_)\) can be anything. When it is an aromatic ring, only adjacent migration is allowed; when it is a cyclic ring, count for the shortest path through the ring for calculating \( n \).

26. **Intra-molecular methyl migration (not considered in the rate rule tree structure now)**

\[
\begin{align*}
R\bullet \sim (\_\_\_\_\_\_\_\_) & \sim R'-CH_3 \rightarrow CH_3-R \sim (\_\_\_\_\_\_\_\_) \sim R'\bullet
\end{align*}
\]

Restriction:

(1) \( R \) can be Cs, Cd, Cb, Cbf, O, CO; if \( R\bullet \) is centered at Cb or Cbf, the migrating \( CH_3 \) should come from the substitution on the adjacent carbon.

(2) \( R' \) can be any groups containing \( CH_3 \); if \( R' \) is a Cb or Cbf, then \( CH_3 \) should migrate to the adjacent position on the aromatic ring.

(3) \( n \) can be any number from 0 to 3.

(4) \((\_\_\_\_\_\_\_\_)\) can be anything. When it is an aromatic ring, only adjacent migration is allowed; when it is a cyclic ring, count for the shortest path through the ring for calculating \( n \).

27. **Intra-molecule addition across double bond to form an exo-cyclic radical**

\[
\begin{align*}
R\bullet \sim (\_\_\_\_\_\_\_\_) & \sim R' \leftrightarrow R'' \rightarrow R\sim(\_\_\_\_\_\_\_\_) \sim R' \leftrightarrow (\_\_\_\_\_\_\_\_)\bullet
\end{align*}
\]

Restriction:

(1) \( 1 \leqslant n \leqslant 4 \)

(2) \( \leftrightarrow \) can be double, triple, benzene(can only happen in the aromatic ring at the carbon site the same as the carbon containing the radical substitution), CO

(3) \( R \) and \( R' \) should not be simultaneously O and CO, forbid the 3-member ring with 2 Os or 2 COs.
(4) \( R \) can be any radical (except biradical)

### 28. Ring opening for an exo-cyclic radical

\[
R \leftrightarrow R' - R'' \rightarrow R \leftrightarrow (+1)R' - R'' \rightarrow R''
\]

Restriction:

1. One cyclic reactant.
2. \( R \) can be Cs, Cd, O, CO.
3. \( R' \) can be Cs, Cd, but \( R' \) cannot be O and CO.
4. \( R'' \) and \( R''' \) can be Cs, Cd, Cb, O, and CO, not H.
5. Prohibition: no 3-member ring with 2 Os or 2 COs is allowed.
6. \( R \) can be any radical center except biradical.

(6) \( R \leftrightarrow R' \) can be single/double bonds, and \( \leftrightarrow \) cannot be triple bond

Note: two cyclic \( \beta \) positions to the radical should be taken into consideration.

### 29. Intra-molecule addition across double bond to form endo-cyclic radical

\[
R \sim ( )_n \sim R' \leftrightarrow R'' \rightarrow R \sim ( )_n \sim R' \leftrightarrow (+1)R''
\]

Restriction:

1. \( 0 \leq n \leq 3 \)
2. \( \leftrightarrow \) can be double, triple, benzene(can only happen in the aromatic ring at the carbon site the adjacent to the carbon containing the radical substitution), CO
3. \( R \) can be any radical (except biradical)
30. **Ring opening for an endo-cyclic radical**

\[
\begin{align*}
R' & \leftrightarrow R'' \rightarrow \bullet \leftrightarrow R'' \rightarrow \rightarrow R' \leftrightarrow (+1)R'' \\
R'' & \leftarrow R'' \leftrightarrow R
\end{align*}
\]

\[
\begin{align*}
R' & \leftrightarrow R'' \rightarrow \bullet \leftrightarrow R'' \rightarrow \rightarrow \rightarrow R' \leftrightarrow (+1)R'' \\
R'' & \leftarrow R'' \leftrightarrow R
\end{align*}
\]

Restriction:

1. One cyclic reactant.
2. R' can be Cs, Cd.
3. R and R'' can be Cs, Cd, O, CO, but cannot be aromatic.
4. R''' and R''''' can be Cs, Cd, O, CO, but cannot be aromatic.
5. R-R'' is a single bond.
6. R'''-R''''' is a single bond.
7. \( \leftrightarrow \) can be anything satisfying valence.

Note: two cyclic \( \beta \) positions to the radical should be taken into consideration.

31. **Cyclic ether formation from alkyl-hydroperoxy radical**

\[
\begin{align*}
R & \leftrightarrow (\cdot)_nO' \cdot OH \rightarrow R \leftrightarrow (\cdot)_nO' + \bullet OH
\end{align*}
\]

Restriction:

1. Reactant should be a single alkyl-hydroperoxy radical.
2. \( 1 \leq n \leq 4 \), so that 3, 4, 5, 6-member cyclic ether formed.
3. R can be Cs, Cd, CO.
4. \( \leftrightarrow \) can be single/double.
5. (\( \cdot \))\( _n \) can include single/double and CO, but not include triple bond. If it includes benzene bond, only 5-member ring can be allowed to be form.
32. **OH radical with cyclic ether forming alkyl-hydroperoxy radical**

\[
\begin{align*}
R & \leftrightarrow ( )_n \leftrightarrow R' \rightarrow \cdot OH \rightarrow R \leftrightarrow ( )_n \leftrightarrow R' - O - OH \\
R & \leftrightarrow ( )_n \leftrightarrow R' \rightarrow \cdot OH \rightarrow R' \leftrightarrow ( )_n \leftrightarrow R - O - OH
\end{align*}
\]

Restriction:

Reactants must be OH radical and cyclic ether.

33. **Intra-molecular hydroxyl migration**

\[
R \cdot \sim ( )_n - O' - OH \rightarrow HO - R \sim ( )_n - O'\cdot
\]

Restriction:

1. R can be Cs, Cd, CO, O.
2. \(0 \leq n \leq 3\).
3. \((\ )_n\) can be anything, but if it includes benzene, then the -OOH and \(R\cdot\) should only at the adjacent positions.

34. **HO2 elimination from peroxy radical**

\[
RH - R' - O - O\cdot \rightarrow R = R' + H - O - O'\cdot
\]

35. **Electrocyclic rearrangement (from cyclic ene to acyclic diene)**

- cyclic ene \(\rightarrow\) acyclic diene

- cyclic diene \(\rightarrow\) acyclic triene (similar to the first one)
Restriction: R has to attach to Cs

**36. Bergman cyclization**

\[
\text{HC}=\text{C}=\text{C}=\text{C}=\text{C} \rightarrow 2 \bullet \text{C=C-C}<4 \bullet 5
\]

Restriction:

1. This is a unique reaction to form benzene from recombination of propargyl radical.
2. Some substitution can be at the ene position.

**37. Conversion of vinylcyclopropane to cyclopentene**

Restriction:

1. This is a unique reaction.
2. All positions can have alkyl substitution.

**38. Cope rearrangement**

Restriction:

1. Reactant must be a 1,5-diene or a 1,5-triene.
2. Reactant must have substitutions.
Appendix 6.2 Reaction Template Documents

Here the author lists the documents for defining every forward reaction template in RMG, including their file name, their text format, and their descriptions.

Notice that there are totally four documents defining each reaction family. The dictionary file is to build a one-to-one mapping between a String FunctionalGroup name and the internal FunctionalGroup graph representation, so that when the tree and library information is read in, the FunctionalGroup names can be used as unique indexes. Internally, both the trees and libraries are stored and searched using FunctionalGroups instead of their names as keys. The tree file defines the kinetics tree structure. The ReactionAdjList file defines the reaction template’s names, valid reactants, and reaction recipe. The rate library files gives kinetics rate rules collected from literature.

<table>
<thead>
<tr>
<th>Document Name</th>
<th>Document Description</th>
<th>Document Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>dictionary.txt</td>
<td>Definitions of kinetics FunctionalGroup structures</td>
<td>FunctionalGroup name FunctionalGroup adjacency list</td>
</tr>
<tr>
<td>tree.txt</td>
<td>The kinetics tree</td>
<td>TreeLevel# : FunctionalGroup name</td>
</tr>
<tr>
<td>reactionAdjList.txt</td>
<td>Definition of valid reactants and its reaction recipe</td>
<td>Reaction Template’s name Its reverse reaction template’s name</td>
</tr>
<tr>
<td>rateLibrary.txt</td>
<td>Kinetics rate rule library</td>
<td>FunctionalGroup name Reaction action list</td>
</tr>
</tbody>
</table>
Chapter 7  Automatic Reaction Mechanism Generation

7.1 Introduction of a reaction mechanism

According to IUPAC definition [McNaught and Wilkinson, 1997], a reaction mechanism is a collection of elementary reactions describing the dynamic process leading from the reactants to the products of a reaction system, with a characterization as complete as possible of the composition, structure, energy and other properties of reactants, reaction intermediates, and products. People sometimes use reaction mechanism concept loosely, referring it to a set of reactions that are not elementary. In this work, we strictly define reaction mechanism as a set of elementary reactions to clear the confusions.

Generally, for a homogeneous chemical reaction system, if a proper reaction mechanism can be obtained, we can mathematically express the system into a set of ordinary differential equations (ODEs) [Steinfeld et al., 1999]:

$$\frac{dC}{dt} = f(C, t, k) \quad C(t_0) = C_0$$

(7.1)

where \(C\) is a n-dimensional vector representing the mass fraction of species and any other state variables such as \(T\), \(t\) is time, and \(C_0\) is the vector of initial condition. \(k\) is the kinetics parameters for all the elementary reactions. \(f(C, t, k)\) represents the species rate change functions, which are determined by all the elementary reactions rates as following:

$$f_i(C, k) = \sum_j v_{ij} R_j$$

(7.2)

where \(i\) and \(j\) represent \(i^{th}\) species and \(j^{th}\) reaction in reaction system, respectively, \(v_{ij}\) is the stoichiometric coefficient of the \(i^{th}\) species in \(j^{th}\) reaction, and \(R_j\) represents the \(j^{th}\) reaction's kinetics rate, which, for elementary reaction, can be calculated by multiplying the \(j^{th}\) reaction's kinetics constant and the reactants' concentrations. If the initial
condition $C_0$ and kinetics rate parameters for all the reactions are available, people can directly solve the ODEs to get the dynamic properties of the reaction system.

Therefore, we can see that generating a proper reaction mechanism is crucial to model a reaction system and to estimate its dynamic properties.

7.2 Algorithm for reaction mechanism generation

So far, RMG can enumerate all the possible reaction pathways, given a set of chemical species. The question remains unanswered is how to choose the chemical species set that be involved in a reaction system, and if people are able to easily determine all the important species involved in a reaction system, then the mechanism generation turns into only one-step call of our model generator discussed in last chapter to produce the corresponding reaction paths. Unfortunately, to identify a proper species set for a reaction system is not that straightforward.

People have different ways for selecting species. First, an experienced chemist might be able to announce all the important species from his or her in-depth understanding of the reaction system. However, to be or to find such an expert is not that easy; furthermore, even for such person, it is easy to make mistakes like missing important species, such as some intermediate radicals.

The second way to fulfill this goal is to teach the computer some rules to pick up just the right species. The advantage of this approach is that computer never makes random mistakes, and it will loyally follow and obey the rules people teach them. The disadvantage is obviously that the computer doesn’t adjust their selections, as cleverly as people do, and if the rule sometimes doesn’t make sense, the computer will do quite stupid things. As a result, the best way here then is to make the species-selecting rule clear and smart. People have been studying this problem for a while. Originally, people used the species/radical size as the selection rule. For example, some of model generation software ignores all the C4 and heavier radicals in pyrolysis. This doesn’t make that much sense since the complexity of the reaction system doesn’t purely depend on the size of the species, and sometimes, heavy hydrocarbon radicals are quite important beginning points of many necessary reaction pathways.
Another widely used rule is to employ the dynamic properties like the species concentration and flux as the standard to choose the next species candidate. This rule depended on the flux, so-called rate-based termination rule proposed by Susnow and coworker [1997], is an acceptable rule. Its key idea is this assumption: the reason people need to enlarge the reaction model is because that there exists some species whose formation flux is so significantly large that we have to include it into reaction system and explore and add all the reaction pathways from it. In another word, the formation flux of the unknown species to the system is the measurement of the potentials or the ability that species jump into the reaction system becoming a reactant. Note that such approach is an iterative approach, where, at each iteration step, the reaction mechanism is enlarged for one more species and its reaction pathways, and it could grow into infinite if no termination rule is set. To ensure a reaction mechanisms properly include important species and reactions while to avoid generating a model with unnecessarily huge size, a termination criterion is used to judge the completeness of a reaction mechanism. The termination rule proposed by Susnow et al. [1997] is, again, a rate-based termination rule that states the reaction mechanism generation will be terminated when all the formation fluxes of the non-included products are less than the system minimum rate.

### 7.2.1 Rate-based iterative model generation

This rate-based iterative model generation algorithm includes two important parts: the iterative reaction scheme generator that generates all the possible reactions from given reactants and a rate-based rule that selects the most important species and reactions for inclusion in the kinetic model.

An illustration on this algorithm by a schematic example is shown in Figure 7.1. The squares in Figure 7.1 are called reaction pool, which indicates the reaction mechanism’s boundary. Species and reactions inside the reaction pool are called “reacted species” or “core species” and “reacted reactions” or “core reactions”, while the species and reactions outside and cross the reaction pool are called “unreacted species” or “edge species” and “unreacted reactions” or “edge reactions". 

178
We can see that during the model generation process, the termination criterion plays an important role in defining the completeness of a reaction mechanism. This termination criterion, so-called the rate-based termination rule, can be simply expressed as:

\[ r_j(t) < R_{\text{min}}(t) = \varepsilon \cdot R_{\text{char}}(t) \quad \text{for all } j = 1, \ldots, m \]

\[ t_0 \leq t \leq \tau \]  

\[ R_j \leq R_{\text{min}} \]

Termination

Figure 7.1 rate-based reaction mechanism generation

Without losing generality, in the example in Figure 7.1, we assume that there are three initial reactants, shown as A, B, and C, in reaction system. At the first iteration of reaction mechanism generation, all the reactants are reacted to each other to create all the possibly involved elementary reaction steps, and as a result, produce new products, shown as D, E, F, and G. The system dynamic simulator then solves the present reaction mechanism inside the reaction pool to calculate the formation rates of all the new products D, E, F, and G. Subsequently, the unreacted species with the largest formation rate, assumed to be D, is then added into the reaction pool and becomes a new reacted species, and the reaction pathways between D and all other reacted species are included into the reaction mechanism as well. As a result, the mechanism grows by one more species. Similar steps then can be repeated to the reaction pool until the rate-based termination rule is satisfied. At the termination, all the reacted species and reactions are output as the final reaction mechanism.
where $r_j(t) = \frac{dC_i}{dt}(t)$ represents the formation rate of the $j^{th}$ unreacted species; $m$ is the total number of unreacted species. $R_{\text{min}}(t)$ is defined by the product of the characteristic reaction rate of the system $R_{\text{char}}(t)$ and the user-specified tolerance $\varepsilon$. To estimate reaction system $R_{\text{char}}(t)$, in original rate-based termination rule, Susnow et al used $L_1$-norm of all formation rate of the reacted species flux, i.e., $R_{\text{char}}(t) = \sum |R_i(t)|$, where $R_i(t) = \frac{dC_i}{dt}(t)$ denotes the rate of change in the concentration the $i^{th}$ reacted species. In Song et al. [2002], a slightly different $R_{\text{char}}(t)$ was proposed, which is calculated as the $L_2$-norm of the reacted species flux vector, $R_{\text{char}}(t) = \sqrt{\sum R_i^2(t)}$. The tolerance, $\varepsilon$, is user-defined, and it represents the desired precision level. The user may obtain a better model with more species and reactions by simply shrinking down the tolerance $\varepsilon$. The meaning of $\varepsilon$ and the relation between the tolerance and the model structure has been discussed in Susnow et al. [1997]. Matheu et al. [2001] have presented explicit bounds on the model truncation errors as a function of $\varepsilon$ for linear systems. Bhattacharjee et al. [2002] have shown how to choose $\varepsilon$ to ensure the model truncation error is less than the numerical errors associated with conventional ODE solver tolerances. The time scale, $\tau$, is also a user-specified parameter to indicate the ending time of system simulation. Alternatively, the user can instead specify the goal conversion of the main reactant to indirectly define the time scale. The physical meaning of this rate-based termination criteria (7.3) can be illustrated in Figure 7.2.
The advantages of this rate-based iterative model generation algorithm are three-fold. First, it comprehensively identifies every possible reaction of the species in the model and every species they can form in a single step. Second, it provides the user not only a kinetic model, but also the “edge” species and reactions surrounding the model, so that people have a understanding of what species and reactions could be potentially added into present mechanism. Finally, this algorithm quantifies the relations between a reaction mechanism and its edge, and tests that the “edge” species and reactions are negligible compared to the core model. In summary, this algorithm properly determines the completeness of a kinetic model by checking the negligibility of the model’s edge.

7.2.2 Pressure dependent effects

Pressure’s effects on reaction kinetics have been studied by different groups, and algorithms and corresponding software for predicting pressure-dependent kinetics rate for individual reaction have been developed successfully, for example, CHEMDIS [Chang et al., 2000], a public-available Quantum-Rice-Ramsperger-Kassel/Modified Strong Collision (QRRK/MSC) program for calculating $k(T,P)$.

However, most model generation software hasn’t taken pressure dependent effects into account, and only the high-pressure limit rate rules are used for reaction mechanism generation. This could be a serious problem for the rate-based reaction generation
algorithm, since its rule to determine the important species and to terminate model
generation process is based on the rates of all the species, which greatly depends on the
quality of reactions’ kinetics. Using high-pressure limit rate kinetics for a pressure-
dependent network overestimates the reaction rate significantly; sometimes the
differences are more than 5 or 6 orders of magnitude. With such a large error in reaction
kinetics, it is very possible that rate-based termination algorithm generates a reaction
mechanism with an unsatisfying prediction.

Targeting this problem, D. Matheu incorporated, for the first time, the pressure-
dependent reactions into rate-based reaction model generation algorithm in a systematic
way, and improved XMG, the first-version rate-based reaction generation software, to
XMG-PDEP [Matheu, 2003], a rate-based pressure-dependent reaction mechanism
generation software. The application of XMG-PDEP to methane pyrolysis revealed a
new, unexpected mechanism, which successfully explained the low-temperature
autocatalytic behavior [Matheu et al., 2003].

Matheu’s idea for adding pressure dependent reactions is similar to the original
rate-based reaction model generation rule, where the unreacted species rate is considered
as a quantified measurement of the error introduced into a reaction system by not
including that species, and if the error is over a certain level, \( R_{\text{min}} \), the unreacted species
will be added. In Matheu’s algorithm, he defined a new variable, \( R_{\text{leak}} \), to quantify the
error accounting for the non-exploring part of pressure dependent network, and if the
error is greater than \( R_{\text{min}} \), that pressure-dependent network will be explored one level
deeper. Therefore, the algorithm of XMG-PDEP only needs to insert an additional test in
every iteration step to compare not only the formation rate of every unreacted species, but
also the \( R_{\text{leak}} \) of each pressure-dependent network with \( R_{\text{min}} \), and if any of them is
greater than \( R_{\text{min}} \), the reaction mechanism will be enlarged by either adding a new
unreacted species or exploring a pressure-dependent network. Therefore, Matheu’s
algorithm provided not only an algorithm to estimate pressure dependent effects, but also
a systematic and feasible way to embed the pressure-dependent part into the original rate-
In this work, the author, with the help from D. Matheu, implemented this pressure dependent reactions mechanism generation algorithm in RMG.

7.2.3 Primary reaction library for oxidation

So far, the author has been discussing algorithms and rules to generate elementary reaction from reaction families. Although we have studied and constructed a quite complete set of reaction families, there are still some irregular elementary reactions not following the reaction family’s generation rule, i.e., RMG cannot automatically generate them from our present reaction template. For example, in oxidation reactions, there are some molecules, such as O, CH, CH₂, CO, and CO₂, whose reacting behaviors are quite different from normal species, and as a result, there exist a set of strange reactions from them, and using reaction family to describe such reactions seems difficult. The best way to dealing with this problem is to build a reaction library collecting those small molecule reactions and their kinetics. RMG will then use this library as a core model when it generates hydrocarbon oxidation mechanisms.

To building this small molecule oxidation library, the author firstly studied and compared current hydrocarbon oxidation mechanisms in literature, and chose the Leeds methane oxidation mechanism [Hughes et al., 2001] as the basis building this library, since most of their kinetics rates are from literature values that have been agreed by many kinetists. Secondly, the author picked the Leeds reactions that can’t be generated from our reaction families to form our small molecule oxidation library with 45 species and 121 reactions. The detailed species and reactions definitions are shown in Appendix 7.1.

To use this small molecule oxidation library, RMG will read in this small molecule oxidation library at the beginning of an oxidation mechanism generation, and build the initial core reaction mechanism on it. After the initialization, RMG will follow the same rate-based iterative model generation algorithm to finally construct an oxidation mechanism.

7.3 RMG approach for reaction mechanism generation

Combining rate-based reaction generation algorithm and pressure dependent network explore algorithm, and the small molecule reactions library, RMG is able to
generate reaction mechanisms for pyrolysis and oxidation reaction systems. Shown in Figure 7.3 is a description of RMG model generation algorithm.
Figure 7.3 RMG approach for reaction mechanism generation
Next, the author will give a step-by-step description of this RMG model generation algorithm.

### 7.3.1 Initialize reaction system

To initialize a reaction mechanism generation, we ask users provide necessary information about reaction system initial conditions and users' choices on the model generation parameters.

Reaction system initial conditions include temperature, pressure, reactants structure, initial concentrations or mole fractions of reactant(s) and inert gas, if there is inert gas. Users' choices of model generation parameters include the reaction families selected for modeling a reaction system, the model enlarger defining the rules to grow a reaction mechanism, the finish controller describing the termination rule of model generation as well as its precision parameters, the dynamic simulator selected for simulating reaction system dynamics and its precision requirements, including sensitivity analysis or not, and, finally, using an small molecule oxidation reaction library or not. In figure 7.4, the author shows the input file format of the initialization of model generation.
TemperatureModel: Constant 715 (K)
PressureModel: Constant 0.73 (atm)

Initial Status:
(1) C4H10 5.3E-7 (mol/cm3)
  1 C 0 {2,S}
  2 C 0 {1,S} {3,S}
  3 C 0 {2,S} {4,S}
  4 C 0 {3,S}

(2) O2 1.1E-6 (mol/cm3)
  1 O 0 {2,D}
  2 O 0 {1,D}
END

InertGas:
N2 1.08E-5 (mol/cm3)
END

ReactionModelEnlarger: RateBasedPDepModelEnlarger

FinishController: RateBasedPDepFinishController
(1) Goal Conversion: C4H10 0.8
(2) Error Tolerance: 0.1

DynamicSimulator: DASPK
TimeStep: 100 (sec)
Atol: 1e-25
Rtol: 1e-4

SA: Off

PrimaryReactionLibrary: on
Name: Leeds
Location: database\primaryReactionLibrary\combustion_core\version4

Figure 7.4 format of input document for RMG model generation

For the temperature and pressure model, people could choose using a constant temperature/pressure or a varied temperature/pressure, and then give the initial value for temperature and pressure. Temperature units, Kelvin, degree F, and degree C, represented by K, F, and C, respectively, are all allowed. Pressure units, Atm, Torr, Pa, bar, are all allowed.

For reactants, users have to specify the name, the concentration/mole fraction, and the structure, of a reactant. In specifying reactant structure, user can choose the simplified adjacency list representation omitting the hydrogen atoms, as shown in this
example. If a resonance isomer is a reactant, user only needs to define one of its resonance structures, and RMG will automatic generate the others. Note that a key word “END” should be put at the end of the reactants initial condition specification section as a section indicator. Next section is to define the concentration(s) or mole fraction(s) of inert gas(s). Again, there is an “END” indicator for this section, too.

Next part is to indicate the type of the reaction mechanism enlarger. There are two types of them in RMG, RateBasedModelEnlarger that implements rate-based species-selecting rule, and RateBasedPDepModelEnlarger that combines the rate-based species-selecting rule with pressure-dependent network enlargement rule, which will be introduced soon in details in section 7.3.5.

In next part, user has to specify the type of finish controller determining if a reaction mechanism generation process is completed. There are also two types of finish controller available in RMG, RateBasedFinishController that use purely rate-based termination rule, and RateBasedPDepFinishController that combines the rate-based termination rule with the pressure-dependent network requirements. The details will be discussed in section 7.3.4.

The coming part specifies the dynamic simulator used in RMG to solve a reaction mechanism. The available dynamic simulators in RMG are DASSL and DASPK. The corresponding parameters specifying the initial time step, relative tolerance, and absolute tolerance should also be defined at this part.

The sensitivity analysis part indicates if user wants to turn on the sensitivity analysis during the model generation. In this example, the author turned it off. Users may want to turn it on to check the sensitivity data.

The final section specifies if the user is willing to include a primary reaction library for oxidation. Currently, there is only one available primary oxidation library, developed based on Leeds methane oxidation mechanism. People can later add their own primary reaction library if necessary.
7.3.2 Build initial reaction mechanism

There are two types of initial reaction mechanism in RMG, an initial reaction mechanism purely built from initial reactants, and an initial reaction mechanism built from both initial reactants and the primary reaction library.

For the first type, RMG will pass a set of initial reactants to its reaction families to generate all the elementary reactions from those species to form an initial reaction mechanism.

For the second type, RMG will first read in the species and elementary reactions defined in the primary reaction library as the initial part of the reaction mechanism, and then pass the union of initial reactants and the species read from primary reaction library into its reaction families to generate all possible elementary reactions, which will be combined with inputted primary reactions to form the initial reaction mechanism.

7.3.3 Solve reaction mechanism dynamics

There are different ways to solve reaction mechanism dynamics for different types of reaction systems. For example, when a reaction system is a gas phase homogenous system, we can just use a set of ordinary differential equations (ODEs) to describe it, and an ODE solver will be proper to solve its dynamic properties. However, reaction mechanisms are also widely used in computational fluid dynamics (CFD) simulation, where the transport phenomena should be taken into account, and the reaction system is then represented by a set of partial differential equations (PDEs). In such case a PDEs solver is required and generally the solving procedure for PDEs is much more complex and expensive than that for ODEs.

In RMG, since the main focus is on model generation, the author assumed that reaction systems are homogeneous so that we can use simple ODEs solvers to simulate the reaction system dynamic. People interested in improving dynamic simulators connected with RMG could later add new and more powerful PDE solver to deal with complex CFD problems. Not only interested in solving ODEs problems, the author also desires to study the system sensitivity and uncertainty problems. The algorithms for solving ODE and sensitivity analysis problems have been thoroughly studied [Lutz, et al.,
1988; Maly, T., and Petzold, L. R., 1995; Feehery, W. F., et al., 1997], and corresponding software has been developed and published. In RMG, DASPK 3.0 [Li and Petzold, 1999] is used to solve the ODE/SA problem.

As introduced earlier, user could set up the time step and relative tolerance and absolution tolerance for the ODE solver.

### 7.3.4 Test termination rule

After system dynamic simulator solves the ODEs, users will gain a list of concentrations and fluxes profiles of the reacted species in reaction system. That information, so-called system snapshots, will be provided into our reaction system finish controller to test the completeness at two aspects: (1) if the reaction has ended, and (2) if the reaction mechanism is valid.

To do the first test, finish controller of RMG simply calculates the major reactant’s conversion, and compare it with the goal conversion user specified in the initialization documents. If the goal conversion is achieved, this test is passed.

To do the second test, finish controller of RMG will calculate the reaction system $R_{\text{min}}(t)$ profile according to equation (7.3), and unreacted species flux profiles $R_j(t)$ from unreacted reactions kinetics, and then compare the $R_{\text{min}}(t)$ and $R_j(t)$ to check if, for every unreacted species, the rate-based rules termination rule has been satisfied.

If a reaction mechanism passes both two tests simultaneously, we will conclude that it is a complete reaction mechanism for this model generation process, and output the reacted species and reactions as the final reaction model; otherwise, RMG will continue on generating a bigger mechanism.

### 7.3.5 Enlarge reaction mechanism

There are two ways to enlarge a reaction mechanism, enlarging it by adding one unreacted species and enlarging it by exploring one pressure-dependent network. In this step, RMG firstly identify what is the right way to enlarge a reaction model. Again, the $R_{\text{leak}}$ from every pressure dependent network and $R_j$ of every unreacted species calculated from the present system snapshot are compared to each other, and the object
with maximum value will be chosen as the next candidate to update the reaction mechanism.

If the next updated candidate is an unreacted species, RMG will firstly add it into its core species set, secondly call its reaction generator to generate all possible reactions from this species and between this species and any other reacted species, and finally add more new-generated reacted reactions into the core reaction set. By this way, the reaction mechanism is enlarged by one more species, and the dynamic simulator should be then called again to calculate the dynamics for this new mechanism.

If the next updated candidate is a pressure dependent network, RMG will enlarge that network by exploring one more isomer of it. The details on building and enlarging a pressure dependent network has been thoroughly discussed in D. Matheu’s PhD thesis [2003], and people might refer to it for more information.

7.3.6 Output reaction mechanism

Finally, once the mechanism generation process is finished, the complete reaction mechanism will be output into a text format, shown in Figure 7.5. Besides this format, RMG also offers the option of outputting CHEMKIN format of reactions.
Model Core:
This model includes totally 66 Species and 1092 Reactions.

Species Set:
Totally 66 Species:
Species 1  Name: C4H10
ChemFormula: C4H10
1  C 0 {2,S}
2  C 0 {3,S} {1,S}
3  C 0 {4,S} {2,S}
4  C 0 {3,S}

Species 2  Name: O2
ChemFormula: O2
1  O 0 {2,D}
2  O 0 {1,D}

Reaction Set:
Totally 1092 Reactions:

<table>
<thead>
<tr>
<th>Reaction</th>
<th>Forward/Back ward</th>
<th>Type</th>
<th>A</th>
<th>n</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2H(22) + C2H(22) -&gt; C4H2(10)</td>
<td>Direction = 1</td>
<td>R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0.291667</td>
</tr>
<tr>
<td>C2H(22) + C2H5(24) -&gt; C2H4(6) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>1.08E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2H(22) + C3H5(37) -&gt; C3H4(8) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>5.41E+12</td>
<td>-0.08875</td>
<td>-0.03656</td>
</tr>
<tr>
<td>C2H(22) + C3H7(40) -&gt; C3H6(9) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>2.17E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2H(22) + C3H7.(48) -&gt; C3H6(9) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>1.21E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2H(22) + C4H7.(826) -&gt; C4H6(267) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>1.14E+13</td>
<td>-0.071</td>
<td>-0.00217</td>
</tr>
<tr>
<td>C2H(22) + C4H7.(912) -&gt; C4H6(267) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>1.08E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2H(22) + C4H9(46) -&gt; C4H8(49) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>1.21E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C2H(22) + C4H9.(47) -&gt; C4H8(49) + C2H2(5)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>1.08E+13</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
7.4  Design of rxnSys package in RMG

RxnSys package of RMG is designed to generate a complete reaction mechanism for a reaction system from initial reaction conditions and users requirements. Therefore, the main goal of rxnSys package is to implement ReactionSystem that represents a reaction system.

7.4.1  ReactionSystem object diagram

Figure 7.6 shows the UML object diagram for reaction system design. The center of this diagram is ReactionSystem class. Surrounding the ReactionSystem class, there are classes defining important reaction system composition. TemperatureModel and PressureModel represent the reaction conditions temperature and pressure. SystemSnapshot class defines system dynamic state at any time. InitialStatus is a special type of SystemSnapshot that describe the reaction system initial status. DynamicSimulator class is responsible to solve reaction system ODEs to provide system dynamic profiles. ReactionModel is the central class in this part describing the reaction mechanism RMG builds to model this reaction system. In order to generate a reaction model, we have to define ReactionGenerator that generates all possible TemplateReactions from reaction system’s reacted species, PrimaryReactionLibrary that defines irregular elementary reactions not able to be generated from reaction family, ReactionModelEnlarger that systematically decide how to enlarge a model as needed, and finally, FinishController that judges if a reaction model generation is completed.

After those composition classes are defined, ReactionSystem class assembles and organizes them together to model a real reaction system. The main methods in ReactionSystem are to handle the communications among its components, for example, when we need to solve the system ODEs, solveReactionSystem() method will be called to organize the information on the reaction mechanism, temperature, pressure, and initial and present system status and transfer them to the dynamic simulator; similarly, after dynamic simulator finish solving ODEs, this method is also response for parsing the result to remember create a new system snapshot. For the details on methods functionality and implementation, people should refer to the UML documentation for RMG.
Figure 7.6 Reaction system object diagram
In next several sections, the author will discuss the design and implementation of important composition classes of ReactionSystem.

### 7.4.2 ReactionModel object diagram

ReactionModel object diagram, shown in Figure 7.7, is the key part in RMG since it models a reaction mechanism.

#### 7.4.2.1 ReactionModel

A general-meaning reaction mechanism is just a collection of reactions, which could be elementary or non-elementary ones, and the associated species included in reactions. Therefore, the necessary parts of a reaction model are a reaction set and a species set.

Aiming to model different types of the reaction mechanisms, the author designed the ReactionModel as an interface, where main methods like getSpecieSet() and getReactionSet() that reveal the species and reactions of a reaction mechanism are required. With such definition, users can easily define their own reaction mechanism classes implementing ReactionModel interface.

#### 7.4.2.2 AbstractReactionModel

AbstractReactionModel class is then a prototype of implementation class of ReactionModel interface.

The main compositions of AbstractReactionModel include a set of species and a set of reactions, defined by Species and Reaction classes in chem package. Note that it is important to maintain a logic relation between the species set and the reaction set in an AbstractReactionModel, i.e., when a reaction is added into the reaction model, we need to check if that reaction includes some species new to this model and make sure to add corresponding new species set into the model.

Main methods of AbstractReactionModel include adding/removing species and reactions into model, testing the logic relation between species and reaction sets, checking the logic relations between two AbstractReactionModel instances, etc.
Figure 7.7 Reaction model object diagram
7.4.2.3 CoreEdgeReactionModel

CoreEdgeReactionModel class is designed to model the reaction models used in rate-base model generation process, where a reaction mechanism has a core consisting of reacted species and reactions and an edge consisting of unreacted species and reaction. CoreEdgeReactionModel model has two important compositions, a core and an edge, which are modeled by Core class and Edge class.

Core class represents the collection of reacted species and reactions, and Edge class represents the collection of the unreacted species and reactions. They are both subclasses of AbstractReactionModel class, and their behaviors are almost the same as their superclass.

In CoreEdgeReactionModel’s implementing ReactionModel’s methods of getSpeciesSet() and getReactionSet(), the core model’s species and reactions are returned, since the core of a CoreEdgeReactionModel is the actual model, and the edge provides some supplementary information. There are some other important CoreEdgeReactionModel methods for checking and maintaining the right relationship between the core and the edge of a CoreEdgeReactionModel. There are many important rules for core-edge relations, for example, the two species sets from core and edge should be disjoint, and the two reaction sets from core and edge should also be disjoint; furthermore, the reactants species in every edge reaction should be already included in core’s species set, and edge reaction should have at least one product defined as edge species. Those rules are naturally from the definition of core-edge reaction model described in details in section 7.2.

In RMG, so far, the rate-based model generation algorithm is the key to generate a reaction model; therefore, all the reaction models built in RMG are instances of CoreEdgeReactionModel. People later can implement their own type of ReactionModel if some other kind of model generation algorithm is introduced.
7.4.3 DynamicSimulation object diagram

DynamicSimulation object diagram defines how RMG computes and manages the reaction system dynamics. There are two main parts in this diagram, the dynamics simulator definition and the system snapshots definition.

7.4.3.1 DynamicSimulator

DynamicSimulator is an interface defining the dynamics simulator used in RMG. The method required in this interface is solve(), which can successfully solve a reaction system’s dynamic profiles. There are three types of dynamic simulators declared in RMG, ODE solver, SA solver, and DAE solver.

ODESolver is for solving an ordinary differential equations’ initial value problem. There are many algorithms and software developed for this purpose. The one implemented in RMG is JDASSL, a class building the communication between RMG and DASSL, a FORTRAN program for solving ODEs. SASolver is for solving a set of ordinary differential equations’ initial value problem with sensitivity analysis. There are many algorithms and software developed for this purpose. The one implemented in RMG is JDASPK, a class building the communication between RMG and DASPK, a FORTRAN program for solving ODE/SA problem. DAESolver is for solving a system of differential algebraic equations. The author didn’t implement this section, and it was designed to having a place for people willing to include a DAE solver into RMG.

Since DASPK is able to solve both ODE and ODE with SA problem, the author will focus on describing how to implement JDASPK class only. DASPK is a Fortran program, and RMG is developed in Java; if we want to employ DASPK as RMG’s dynamic simulator, we have to deal with the issue of calling a FORTRAN program from a Java application. This issue is not an easy case, since the Java program running on Java Virtual Machine, JVM, a layer between applications and the fundamental operating system, and any communication between a Java application with any native program running on the fundamental operating system has to go through the JVM [Anderson, 1997]. In order to integrate programs written in other languages, Java program provide users a library, Java Native Interface, JNI [Liang, 1999], allowing Java code operate with applications and libraries written in other languages, such as C, C++, and Assembly.
Note that there is no direct communication library for linking a Java program and a FORTRAN program. To do so, users have to develop their own C or C++ interfaces connecting Java and FORTRAN, as shown in Figure 7.8.

![Figure 7.8 Connecting Java program with Fortran program](image)

In RMG, the author built a C++ interface, daspk, successfully linking RMG and DASPK.

7.4.3.2 SystemSnapshot

SystemSnapshot class models the reaction system status with respect to reaction time. Therefore, a reaction system will include a sequence list of SystemSnapshot instances recording the system status changes. The main attributes of SystemSnapshot are reaction time and a set of species status. ReactionTime class models the reaction time, and SpeciesStatus class stores the concentration and flux information of a species at a special time. In detail, a SystemSnapshot instance describes all the species' concentrations and fluxes at one specific reaction time. For any reaction system, after the dynamic simulator is called and returns the dynamic solution at one reaction time, RMG will create a new SystemSnapshot instance and added it into the system snapshot list of the reaction system. Finally, when the model generation process is completed, the system snapshot list also records all the dynamic information of any species at any reaction time. On the other hand, system snapshot list will also be used in judging the validity of the model generation based on species rate, which will be discussed in detail in next section.

There are three subclasses of SystemSnapshot to represent three types of special reaction system status, Initial Status class describing the beginning state, Milestone class recording the important system states, PresentStatus recording the reaction system states at current reaction time. Every reaction system will begin with an initial state, so ReactionSystem class always include an instance of InitialStatus. PresentStatus always
representing the current system state, and it is used to judge if the goal conversion of major reactants has been achieved, i.e., if the reaction has completed. Milestone should be used when people are really interested in some special species behavior. For example, in combustion, the system is considered to be ignited when some important intermediate radical reaches a certain level. At such circumstance, the system snapshot at the ignition time can be recorded as a Milestone instance.
Figure 7.8 Dynamics simulation object diagram
7.4.4 ReactionModelEnlarger object diagram

ReactionModelEnlarger object diagram, shown in Figure 7.9, defines how RMG enlarges its reaction mechanism.

![ReactionModelEnlarger object diagram]

Figure 7.9 ReactionModelEnlarger object diagram

ReactionModelEnlarger is an interface requiring its children classes implement a method, \texttt{enlargeReactionModel()}, which defines a way to enlarge the present reaction mechanism.

There are two model enlargers implemented in RMG, RateBasedRME and RateBasedPDepRME. RateBasedRME implements the rate-based model generation rule, i.e., it compares all the unreacted species’ formation rates, and add the one with the highest formation rate into reaction mechanism. RateBasedPDepRME class combines the rate-based termination rule with the pressure-dependent network exploring. Not only comparing the unreacted formation rates, RateBasedPDepRME also checks the R\text{leak} from each pressure-dependent network, and the pressure dependent network with the max R\text{leak} will be explored one level further if its R\text{leak} is greater than the maximum unreacted species formation rate.

7.4.5 FinishController object diagram

FinishController object diagram, shown in Figure 7.10, defines how RMG determines a reaction mechanism generation process is terminated.
FinishController class is in charge of determining if the model generation for a reaction system is complete. It includes two main parts: termination tester and validity tester. If and only if the tester for termination and validity are both satisfied, the FinishController concludes the model generation is successfully completed.

TerminationTester interface is designed to impose the test on the termination of a reaction system. In RMG, there are currently two types of termination testers, ConversionTT and ReactionTimeTT. ReactionTimeTT checks the termination by comparing the present reaction time to the final reaction time that user defined in the initialization document. ConversionTT decides if the system terminated by calculating
the current reactant conversion from present system snapshot and comparing it to the goal conversion described in the initialization document.

ValidityTester interface is to define the validity of the reaction mechanism modeling a reaction system. In this work, since we use rate-based rule to decide if a reaction model is valid, we designed a RateBasedVT to implement this validity test. When checking the validity, RateBasedVT will calculate the system $R_{\text{min}}$ and unreacted species formation rates from system snapshot, and use rate-based termination rule (7.2) to check if the model is valid. To incorporate the pressure-dependent network exploring into RMG, the author also added a RateBasedPDepVT class for the pressure-dependent network validity checking. RateBasedPDepVT class inherits all the features from RateBasedVT, and extends itself by one more test on comparing active pressure-dependent network’s $R_{\text{leak}}$ rate with system $R_{\text{min}}$.

7.4.6 ReactionCondition object diagram

ReactionCondition object diagram describes main reaction system conditions, such as temperature and pressure. The diagram is shown in Figure 7.11.

TemperatureModel is an interface imposing its children class implement a method, getTemperature(time). In RMG, the author declared two types of temperature models, ConstantTM, and CurvedTM. ConstantTM class models an isothermal reaction system, and it will always provide a constant temperature as user specifies in the initialization documents. CurvedTM class models a non-isothermal reaction system, where temperature is varied with respect to time. There are also two types of CurvedTM declared in RMG, AdiabaticTM modeling an adiabatic reaction system and GenericTM whose temperature profile with respect to reaction time has been specified. The author didn’t implement those two curved temperature models, and RMG is now only able to model isothermal reaction system. People could later implement those two temperature models to enhance the RMG ability to model more types of reaction systems.

PressureModel’s design is very similar to TemperatureModel. The author also only implemented the constant pressure model, but left space for people to improve RMG’s pressure model.
7.5 Conclusions and discussions

In this chapter, the author briefly introduced the implementation details of reaction mechanism generation process in RMG. The rate-based model generation method proposed by Susnow and coworkers was used as the fundamental algorithm for this purpose. The algorithm for pressure-dependent effects developed by D. Matheu has also been employed in RMG for pressure dependent rate rules estimation.
In implementing the model generation algorithm, the author paid much attention to software reusability and extendibility. Also the currently implemented RMG can only handle isothermal homogeneous reaction system modeling, the author has left proper design spaces for people to add more features into RMG later.
Appendix 7.1 Small molecule oxidation library

By Jing Song

In this appendix, the author summarizes the small molecule oxidation library developed based on Leeds methane oxidation mechanism v1.5:
http://www.chem.leeds.ac.uk/Combustion/Combustion.html

This small molecule oxidation library includes two parts, species definition and reaction definition. Species definition is shown in following table

<table>
<thead>
<tr>
<th>Molecules</th>
<th>Radicals</th>
</tr>
</thead>
<tbody>
<tr>
<td>H2</td>
<td>H</td>
</tr>
<tr>
<td>1 H 0 {2,S}</td>
<td>1 H 1</td>
</tr>
<tr>
<td>2 H 0 {1,S}</td>
<td>CH</td>
</tr>
<tr>
<td>CH4</td>
<td>1 C 3</td>
</tr>
<tr>
<td>1 C 0</td>
<td>CH2</td>
</tr>
<tr>
<td>C2H2</td>
<td>1 C 2T</td>
</tr>
<tr>
<td>1 C 0 {2,T}</td>
<td>CH2(S)</td>
</tr>
<tr>
<td>2 C 0 {1,T}</td>
<td>1 C 2S</td>
</tr>
<tr>
<td>C2H4</td>
<td>CH3</td>
</tr>
<tr>
<td>1 C 0 {2,D}</td>
<td>1 C 1</td>
</tr>
<tr>
<td>2 C 0 {1,D}</td>
<td>C2H</td>
</tr>
<tr>
<td>C2H6</td>
<td>1 C 1 {2,T}</td>
</tr>
<tr>
<td>1 C 0 {2,S}</td>
<td>1 C 1 {2,D}</td>
</tr>
<tr>
<td>2 C 0 {1,S}</td>
<td>2 C 0 {1,T}</td>
</tr>
<tr>
<td>C3H4</td>
<td>C2H3</td>
</tr>
<tr>
<td>1 C 0 {2,D}</td>
<td>1 C 1 {2,D}</td>
</tr>
<tr>
<td>2 C 0 {1,D}</td>
<td>2 C 0 {1,D}</td>
</tr>
<tr>
<td>3 C 0 {2,D}</td>
<td>C2H5</td>
</tr>
<tr>
<td>C3H6</td>
<td>1 C 1 {2,S}</td>
</tr>
<tr>
<td>1 C 0 {2,D}</td>
<td>2 C 0 {1,S}</td>
</tr>
<tr>
<td>2 C 0 {1,D}</td>
<td>3 C 0 {2,S}</td>
</tr>
<tr>
<td>3 C 0 {2,S}</td>
<td>C3H2</td>
</tr>
<tr>
<td>CH2HCO</td>
<td>CH3O2</td>
</tr>
<tr>
<td>1 C 1 {2,S}</td>
<td>2 C 0 {1,S} {3,D}</td>
</tr>
<tr>
<td>3 O 0 {2,D}</td>
<td>3 O 0 {2,S}</td>
</tr>
<tr>
<td>CH3O2</td>
<td>C3H5</td>
</tr>
<tr>
<td>1 C 0 {2,D} {3,S}</td>
<td></td>
</tr>
<tr>
<td>3 O 1 {2,S}</td>
<td>C3H502</td>
</tr>
<tr>
<td>1 C 0 {2,D} {3,S}</td>
<td></td>
</tr>
<tr>
<td>3 C 1 {1,S}</td>
<td>4 O 0 {3,S} {5,S}</td>
</tr>
<tr>
<td>5 O 1 {4,S}</td>
<td>C2H502</td>
</tr>
<tr>
<td>1 C 0 {2,S}</td>
<td>2 C 0 {1,S} {3,S}</td>
</tr>
<tr>
<td>3 O 0 {2,S} {4,S}</td>
<td></td>
</tr>
<tr>
<td>4 O 1 {3,S}</td>
<td>C3H7</td>
</tr>
<tr>
<td>1 C 2 {2,S}</td>
<td></td>
</tr>
<tr>
<td>C4H2</td>
<td>2 CO {1,S} {3,T}</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>1 CO {2,T}</td>
<td>3 CO {2,T}</td>
</tr>
<tr>
<td>2 CO {1,T} {3,S}</td>
<td>3 CO {2,T}</td>
</tr>
<tr>
<td>3 CO {2,S} {4,T}</td>
<td>H2CCCH</td>
</tr>
<tr>
<td>4 CO {3,T}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>O2</th>
<th>1 O 0 {2,D}</th>
<th>2 CO {1,S}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 O 0 {2,D}</td>
<td>2 CO {1,S}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td></td>
<td>H2CCCH</td>
<td>4 O 0 {2,S}</td>
</tr>
<tr>
<td></td>
<td>1 CO {2,D}</td>
<td>5 O 1 {4,S}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H2O</th>
<th>1 O 0 {2,S}</th>
<th>2 CO {1,S}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 O 0 {2,S}</td>
<td>2 CO {1,S}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td></td>
<td>H2CCCH</td>
<td>4 O 0 {2,S}</td>
</tr>
<tr>
<td></td>
<td>1 CO {2,D}</td>
<td>5 O 1 {4,S}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CO</th>
<th>1 C 2T {2,D}</th>
<th>2 CO {1,D}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 C 2T {2,D}</td>
<td>2 CO {1,D}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 CO {1,D}</td>
<td>H02</td>
<td>4 O 0 {2,S}</td>
</tr>
<tr>
<td>3 CO {3,D}</td>
<td>1 O 2T</td>
<td>5 O 1 {4,S}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>H2O2</th>
<th>1 O 0 {2,S}</th>
<th>2 CO {1,S}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 O 0 {2,S}</td>
<td>2 CO {1,S}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 O 0 {1,D}</td>
<td>HCO</td>
<td>4 O 0 {2,S}</td>
</tr>
<tr>
<td>3 CO {2,S} {4,T}</td>
<td>1 CO {2,D}</td>
<td>5 O 1 {4,S}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH2O</th>
<th>1 CO {2,D}</th>
<th>2 CO {1,D}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CO {2,D}</td>
<td>2 CO {1,D}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 CO {1,D}</td>
<td>CH3O</td>
<td>4 O 0 {3,S}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH2CO</th>
<th>1 CO {2,D}</th>
<th>2 CO {1,D}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CO {2,D}</td>
<td>2 CO {1,D}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 CO {1,D} {3,D}</td>
<td>CH2OH</td>
<td>4 O 0 {3,S}</td>
</tr>
<tr>
<td>3 CO {2,D}</td>
<td>1 CO {2,S}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 O 0 {1,S}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CH20H</th>
<th>1 CO {2,S}</th>
<th>2 CO {1,S}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CO {2,S}</td>
<td>2 CO {1,S}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 O 0 {1,S}</td>
<td>HCCO</td>
<td>4 O 0 {3,S}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HCCO</th>
<th>1 CO {2,D}</th>
<th>2 CO {1,D}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CO {2,D}</td>
<td>2 CO {1,D}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 CO {1,D} {3,D}</td>
<td>3 CO {2,S}</td>
<td>4 O 0 {3,S}</td>
</tr>
<tr>
<td>3 CO {2,D}</td>
<td>2 CO {1,S}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2 O 0 {1,S}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C4H9</th>
<th>1 CO {2,S}</th>
<th>2 CO {1,S}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 CO {2,S}</td>
<td>2 CO {1,S}</td>
<td>3 CO {2,S}</td>
</tr>
<tr>
<td>2 CO {1,D} {3,D}</td>
<td>3 CO {2,S}</td>
<td>4 O 0 {3,S}</td>
</tr>
</tbody>
</table>
Reactions definition is shown in the following table:

<table>
<thead>
<tr>
<th>Reaction</th>
<th>A (mol/cm³/s)</th>
<th>n</th>
<th>E (kJ/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₂H₂ + O = CH₂ + CO</td>
<td>2.17E+06</td>
<td>2.1</td>
<td>6.57</td>
</tr>
<tr>
<td>C₂H₂ + O = HCCO + H</td>
<td>5.06E+06</td>
<td>2.1</td>
<td>6.57</td>
</tr>
<tr>
<td>C₄H₂ + O = C₃H₂ + CO</td>
<td>7.89E+12</td>
<td>0</td>
<td>5.64</td>
</tr>
<tr>
<td>O₂ + CO = CO₂ + O</td>
<td>1.26E+13</td>
<td>0</td>
<td>196.9</td>
</tr>
<tr>
<td>O₂ + CH = CO + OH</td>
<td>1.66E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O₂ + CH = CO₂ + H</td>
<td>1.66E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O₂ + CH₂ = CO₂ + H₂</td>
<td>5.43E+12</td>
<td>0</td>
<td>6.24</td>
</tr>
<tr>
<td>O₂ + CH₂ = CO + OH + H</td>
<td>8.15E+12</td>
<td>0</td>
<td>6.24</td>
</tr>
<tr>
<td>O₂ + CH₂ = CO + H₂O</td>
<td>1.48E+12</td>
<td>0</td>
<td>6.24</td>
</tr>
<tr>
<td>O₂ + CH₂ = CH₂O + O</td>
<td>4.20E+12</td>
<td>0</td>
<td>6.24</td>
</tr>
<tr>
<td>O₂ + CH₂(S) = CO + OH + H</td>
<td>3.13E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O₂ + HCO = H₂O + CO</td>
<td>3.01E+12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>O₂ + HCCO = CO + CO + OH</td>
<td>1.63E+12</td>
<td>0</td>
<td>3.58</td>
</tr>
<tr>
<td>CO + OH = CO₂ + H</td>
<td>1.66E+07</td>
<td>1.3</td>
<td>-3.2</td>
</tr>
<tr>
<td>CO + H₂O = CO₂ + OH</td>
<td>1.51E+14</td>
<td>0</td>
<td>99.02</td>
</tr>
<tr>
<td>CO + CH = HCCO</td>
<td>2.77E+11</td>
<td>0</td>
<td>-7.15</td>
</tr>
<tr>
<td>CO₂ + CH = HCO + CO</td>
<td>3.43E+12</td>
<td>0</td>
<td>2.87</td>
</tr>
<tr>
<td>CO₂ + CH₂ = CH₂O + CO</td>
<td>2.35E+10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH₂CO + H = CH₃ + CO</td>
<td>1.81E+13</td>
<td>0</td>
<td>14.13</td>
</tr>
<tr>
<td>CH₂CO + O = CH₂ + CO₂</td>
<td>1.33E+12</td>
<td>0</td>
<td>5.65</td>
</tr>
<tr>
<td>CH₂CO + O = CH₂O + CO</td>
<td>4.58E+11</td>
<td>0</td>
<td>5.65</td>
</tr>
<tr>
<td>CH₂CO + O = HCO + H + CO</td>
<td>2.52E+11</td>
<td>0</td>
<td>5.65</td>
</tr>
<tr>
<td>CH₂CO + O = HCO + HCO</td>
<td>2.52E+11</td>
<td>0</td>
<td>5.65</td>
</tr>
<tr>
<td>CH₂CO + OH = CH₃ + CO₂</td>
<td>2.52E+12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH₂CO + OH = CH₂OH + CO</td>
<td>4.68E+12</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H + HCO = CO + H₂</td>
<td>9.03E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>H + HCCO = CH₂ + H₂</td>
<td>1.51E+14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH + O = CO + H</td>
<td>3.97E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>CH₃ + HCO = CH₄ + CO</td>
<td>1.20E+14</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C₂H + O = CH + CO</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C₂H + OH = CH₂ + CO</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>C₂H₃ + O = CO + CH₃</td>
<td>3.00E+13</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
\[
\begin{align*}
\text{H}_2\text{CCCH} + \text{O} & = \text{C}_2\text{H}_2 + \text{CO} + \text{H} & 1.39 \times 10^{14} & 0 & 0 \\
\text{O} + \text{HCO} & = \text{CO} + \text{OH} & 3.01 \times 10^{13} & 0 & 0 \\
\text{O} + \text{HCCO} & = \text{H} + \text{CO} + \text{CO} & 9.64 \times 10^{13} & 0 & 0 \\
\text{OH} + \text{HCO} & = \text{H}_2\text{O} + \text{CO} & 1.02 \times 10^{14} & 0 & 0 \\
\text{OH} + \text{HCCO} & = \text{HCO} + \text{HCO} & 1.00 \times 10^{13} & 0 & 0 \\
\text{OH} + \text{HCCO} & = \text{CH}_2\text{O} + \text{CO} & 1.00 \times 10^{13} & 0 & 0 \\
\text{HCO} + \text{HCO} & = \text{CH}_2\text{O} + \text{CO} & 3.01 \times 10^{13} & 0 & 0 \\
\text{HCCO} + \text{HCCO} & = \text{C}_2\text{H}_2 + \text{CO} + \text{CO} & 1.00 \times 10^{13} & 0 & 0 \\
\text{CH} + \text{HCCO} & = \text{C}_2\text{H}_2 + \text{CO} & 5.00 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{O} & = \text{CO} + \text{H} + \text{H} & 7.20 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{O} & = \text{CO} + \text{H}_2 & 4.80 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{HCO} & = \text{CH}_3 + \text{CO} & 1.81 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{HCCO} & = \text{C}_2\text{H}_3 + \text{CO} & 3.00 \times 10^{13} & 0 & 0 \\
\text{O}_2 + \text{C}_2\text{H} & = \text{CO}_2 + \text{CH} & 9.05 \times 10^{12} & 0 & 0 \\
\text{O} + \text{HCO} & = \text{CO}_2 + \text{H} & 3.01 \times 10^{13} & 0 & 0 \\
\text{CH}_4 + \text{CH} & = \text{C}_2\text{H}_4 + \text{H} & 3.01 \times 10^{13} & 0 & -1.66 \\
\text{C}_2\text{H}_2 + \text{CH} & = \text{C}_2\text{H} + \text{CH}_2 & 2.11 \times 10^{14} & 0 & -0.51 \\
\text{C}_2\text{H}_4 + \text{CH} & = \text{C}_3\text{H}_4 + \text{H} & 1.32 \times 10^{14} & 0 & -1.44 \\
\text{C}_2\text{H}_6 + \text{CH} & = \text{C}_2\text{H}_4 + \text{CH}_3 & 1.08 \times 10^{14} & 0 & -1.1 \\
\text{CH}_2\text{O} + \text{CH} & = \text{CH}_2 + \text{HCO} & 9.64 \times 10^{13} & 0 & -2.16 \\
\text{H} + \text{CH}_2 & = \text{CH} + \text{H}_2 & 6.02 \times 10^{12} & 0 & -7.48 \\
\text{CH} + \text{CH}_2 & = \text{C}_2\text{H}_2 + \text{H} & 4.00 \times 10^{13} & 0 & 0 \\
\text{CH} + \text{CH}_3 & = \text{C}_2\text{H}_3 + \text{H} & 3.00 \times 10^{13} & 0 & 0 \\
\text{CH} + \text{C}_2\text{H}_3 & = \text{CH}_2 + \text{C}_2\text{H}_2 & 5.00 \times 10^{13} & 0 & 0 \\
\text{CH} + \text{OH} & = \text{HCO} + \text{H} & 3.00 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{CH}_2 & = \text{C}_2\text{H}_2 + \text{H}_2 & 1.20 \times 10^{13} & 0 & 3.33 \\
\text{CH}_2 + \text{CH}_2 & = \text{C}_2\text{H}_2 + \text{H} + \text{H} & 1.08 \times 10^{14} & 0 & 3.33 \\
\text{CH}_2 + \text{CH}_3 & = \text{C}_2\text{H}_4 + \text{H} & 4.22 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{C}_2\text{H}_3 & = \text{C}_2\text{H}_2 + \text{CH}_3 & 1.81 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{OH} & = \text{CH}_2\text{O} + \text{H} & 1.81 \times 10^{13} & 0 & 0 \\
\text{CH}_2 + \text{HCCO} & = \text{C}_2\text{H} + \text{CH}_2\text{O} & 1.00 \times 10^{13} & 0 & 8.37 \\
\text{CH}_4 + \text{CH}_2 & = \text{CH}_3 + \text{CH}_3 & 4.30 \times 10^{12} & 0 & 42 \\
\text{CH}_4 + \text{CH}_2\text{(S)} & = \text{CH}_3 + \text{CH}_3 & 7.00 \times 10^{13} & 0 & 0 \\
\text{C}_2\text{H}_2 + \text{CH}_2 & = \text{C}_3\text{H}_4 & 1.20 \times 10^{13} & 0 & 27.69 \\
\text{C}_2\text{H}_2 + \text{CH}_2\text{(S)} & = \text{H}_2\text{CCCH} + \text{H} & 1.75 \times 10^{14} & 0 & 0 \\
\end{align*}
\]
\[
\begin{align*}
H + CH_2(S) & = CH_2 + H & 2.00E+14 & 0 & 0 \\
H_2 + CH_2(S) & = CH_3 + H & 7.23E+13 & 0 & 0 \\
C_2H_4 + CH_2(S) & = C_3H_6 & 9.64E+13 & 0 & 0 \\
C_2H_6 + CH_2(S) & = CH_3 + C_2H_5 & 2.40E+14 & 0 & 0 \\
CH_3 + OH & = CH_2(S) + H_2O & 7.23E+13 & 0 & 11.64 \\
C_2H_2 + C_2H_2 & = H_2CCCH + H & 2.00E+09 & 0 & 242 \\
C_2H_2 + C_2H & = C_4H_2 + H & 9.03E+13 & 0 & 0 \\
C_2H_4 + O & = H + CH_2HCO & 4.74E+06 & 1.88 & 0.75 \\
C_2H_4 + O & = CH_3 + HCO & 8.13E+06 & 1.88 & 0.75 \\
C_2H_4 + O & = CH_2CO + H_2 & 6.80E+05 & 1.88 & 0.75 \\
C_4H_2 + OH & = C_3H_2 + HCO & 6.68E+12 & 0 & -1.71 \\
O_2 + H + H_2O & = HO_2 + H_2O & 6.89E+15 & 0 & -8.73 \\
O_2 + H & = OH + O & 9.76E+13 & 0 & 62.11 \\
O_2 + CH_3 & = CH_2O + OH & 3.31E+11 & 0 & 37.42 \\
O_2 + C_2H & = HCCO + O & 9.05E+12 & 0 & 0 \\
O_2 + C_3H_2 & = HCO + HCCO & 1.00E+13 & 0 & 0 \\
O_2 + H_2CCCH & = CH_2CO + HCO & 3.01E+10 & 0 & 12 \\
H_2O_2 + H & = OH + H_2O & 1.02E+13 & 0 & 14.97 \\
CH_3 + CH_3 & = C_2H_5 + H & 3.01E+13 & 0 & 56.54 \\
H + HO_2 & = OH + OH & 1.69E+14 & 0 & 3.66 \\
H + HO_2 & = H_2O + O & 3.01E+13 & 0 & 7.2 \\
H + CH_2OH & = CH_3 + OH & 1.02E+13 & 0 & 0 \\
CH_3 + O & = CH_2O + H & 8.43E+13 & 0 & 0 \\
CH_3 + HO_2 & = CH_3O + OH & 1.80E+13 & 0 & 0 \\
C_2H + OH & = HCCO + H & 2.00E+13 & 0 & 0 \\
C_2H_5 + O & = CH_2O + CH_3 & 6.62E+13 & 0 & 0 \\
O_2 + CH_3 & = CH_3O + O & 4.40E+13 & 0 & 131.37 \\
OH + OH & = O + H_2O & 1.51E+09 & 1.14 & 0.42 \\
H_2 + O & = OH + H & 5.12E+04 & 2.67 & 26.27 \\
H_2O + H & = H_2 + OH & 4.52E+08 & 1.6 & 77.08 \\
CH_4 + O & = CH_3 + OH & 7.23E+08 & 1.56 & 35.5 \\
CH_4 + OH & = CH_3 + H_2O & 1.57E+07 & 1.83 & 11.64 \\
C_2H_2 + OH & = C_2H + H_2O & 6.00E+13 & 0 & 54.04 \\
C_2H_4 + H & = C_2H_3 + H_2 & 5.42E+14 & 0 & 62.36 \\
C_2H_4 + OH & = C_2H_3 + H_2O & 2.05E+13 & 0 & 24.86 \\
\end{align*}
\]
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Products</th>
<th>Rate Constant</th>
<th>Collision Efficiency</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{C}_2\text{H}_6 + \text{H}$</td>
<td>$\text{C}_2\text{H}_5 + \text{H}_2$</td>
<td>$1.45 \times 10^9$</td>
<td>1.5</td>
<td>31.01</td>
</tr>
<tr>
<td>$\text{C}_2\text{H}_6 + \text{CH}_3$</td>
<td>$\text{C}_2\text{H}_5 + \text{CH}_4$</td>
<td>$1.51 \times 10^{-7}$</td>
<td>6</td>
<td>25.3</td>
</tr>
<tr>
<td>$\text{C}_2\text{H}_6 + \text{O}$</td>
<td>$\text{C}_2\text{H}_5 + \text{OH}$</td>
<td>$1.00 \times 10^{9}$</td>
<td>1.5</td>
<td>24.28</td>
</tr>
<tr>
<td>$\text{C}_2\text{H}_6 + \text{OH}$</td>
<td>$\text{C}_2\text{H}_5 + \text{H}_2\text{O}$</td>
<td>$7.23 \times 10^{-6}$</td>
<td>2</td>
<td>3.62</td>
</tr>
<tr>
<td>$\text{C}_2\text{H}_6 + \text{HO}_2$</td>
<td>$\text{H}_2\text{O}_2 + \text{C}_2\text{H}_5$</td>
<td>$1.32 \times 10^{13}$</td>
<td>0</td>
<td>85.63</td>
</tr>
<tr>
<td>$\text{O}_2 + \text{CH}_2\text{O}$</td>
<td>$\text{HCO} + \text{HO}_2$</td>
<td>$6.02 \times 10^{13}$</td>
<td>0</td>
<td>170.11</td>
</tr>
<tr>
<td>$\text{O}_2 + \text{CH}_3\text{O}$</td>
<td>$\text{CH}_2\text{O} + \text{HO}_2$</td>
<td>$2.17 \times 10^{10}$</td>
<td>0</td>
<td>7.32</td>
</tr>
<tr>
<td>$\text{H}_2\text{O}_2 + \text{H}$</td>
<td>$\text{HO}_2 + \text{H}_2$</td>
<td>$1.69 \times 10^{12}$</td>
<td>0</td>
<td>15.71</td>
</tr>
<tr>
<td>$\text{H}_2\text{O}_2 + \text{O}$</td>
<td>$\text{OH} + \text{HO}_2$</td>
<td>$6.62 \times 10^{11}$</td>
<td>0</td>
<td>16.63</td>
</tr>
<tr>
<td>$\text{H}_2\text{O}_2 + \text{OH}$</td>
<td>$\text{H}_2\text{O} + \text{HO}_2$</td>
<td>$7.83 \times 10^{12}$</td>
<td>0</td>
<td>5.57</td>
</tr>
<tr>
<td>$\text{CH}_2\text{O} + \text{H}$</td>
<td>$\text{HCO} + \text{H}_2$</td>
<td>$1.26 \times 10^{08}$</td>
<td>1.62</td>
<td>9.06</td>
</tr>
<tr>
<td>$\text{CH}_2\text{O} + \text{CH}_3$</td>
<td>$\text{CH}_4 + \text{HCO}$</td>
<td>$7.83 \times 10^{-8}$</td>
<td>6.1</td>
<td>8.23</td>
</tr>
<tr>
<td>$\text{CH}_2\text{O} + \text{O}$</td>
<td>$\text{HCO} + \text{OH}$</td>
<td>$4.16 \times 10^{11}$</td>
<td>0.57</td>
<td>11.56</td>
</tr>
<tr>
<td>$\text{CH}_2\text{O} + \text{OH}$</td>
<td>$\text{HCO} + \text{H}_2\text{O}$</td>
<td>$3.43 \times 10^{09}$</td>
<td>1.18</td>
<td>-1.87</td>
</tr>
<tr>
<td>$\text{H} + \text{HO}_2$</td>
<td>$\text{H}_2 + \text{O}_2$</td>
<td>$4.28 \times 10^{13}$</td>
<td>0</td>
<td>5.9</td>
</tr>
<tr>
<td>$\text{H} + \text{CH}_3\text{O}$</td>
<td>$\text{CH}_2\text{O} + \text{H}_2$</td>
<td>$1.81 \times 10^{13}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\text{O} + \text{HO}_2$</td>
<td>$\text{O}_2 + \text{OH}$</td>
<td>$3.19 \times 10^{13}$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$\text{OH} + \text{HO}_2$</td>
<td>$\text{H}_2\text{O} + \text{O}_2$</td>
<td>$2.89 \times 10^{13}$</td>
<td>0</td>
<td>-2.08</td>
</tr>
</tbody>
</table>
Chapter 8 Generating n-Butane Low-Temperature Oxidation Mechanism

8.1 Introduction

Understanding n-butane partial oxidation at low temperature (500K – 1000K) is very important for modeling many combustion processes. N-butane is the smallest hydrocarbon that oxidizes rapidly to cyclic ethers, an important class of compounds with very particular kinetics. Autoignition phenomena involved in such process has been studied experimentally by many groups, [Wilk, et al, 1995 and 1990; Chandraratna, et al., 1994; Chakir et al., 1992, 1991, and 1990; Henig, et al., 1985; Green, et al., 1987], and its reaction mechanism has been investigated and developed by many groups to understand the dynamic transition [Warth, et al., 1998; Ranzi, et al., 1994; Bounaceur, et al., 1996; Marinov, et al., 1998; Pitz, et al., 1986; Dente, et al., 2003; Kojima, 1994].

In this work, the author applied RMG as an automated model generation tool into this complex reaction process to test the quality of the model generated from RMG.

8.2 Generate n-butane low-temperature oxidation

To generate an n-butane low-temperature oxidation mechanism, the author choose the reaction conditions described in the experimental investigation of n-butane oxidation provided by Wilk et al., (1995). The conditions are summarized as following: $T = 715K$, $P_{n\text{-butane}} = 23.6$ Torr, $P_{\text{oxygen}} = 47.1$ Torr, $P_{\text{nitrogen}} = 479.3$ Torr. Because this is a fuel-rich mixture, the kinetics change from oxidation to pyrolysis during the process. DASPK [Li and Petzold, 1999] was used to simulate the dynamic profiles of species during the oxidation. The small molecule oxidation library described in last chapter was used as the initial core to generate this model. Pressure dependent rates are calculated to capture the fall off and chemical activation, which is important for some pathways.

Figure 8.1 shows the initialization file input to RMG for generating this n-butane model.
TemperatureModel: Constant 715 (K)

PressureModel: Constant 0.73 (atm)

InitialStatus:
(1) C4H10 5.3E-7 (mol/cm3)
  1 C 0 {2,S}
  2 C 0 {1,S} {3,S}
  3 C 0 {2,S} {4,S}
  4 C 0 {3,S}

(2) O2 1.1E-6 (mol/cm3)
  1 O 0 {2,D}
  2 O 0 {1,D}

END

InertGas:
N2 1.08E-5 (mol/cm3)
END

ReactionModelEnlarger: RateBasedPDepModelEnlarger

FinishController: RateBasedPDepFinishController
(1) Goal Conversion: C4H10 0.9
(2) Error Tolerance: 0.2

DynamicSimulator: DASPK
TimeStep: 100 (sec)
Atol: 1e-25
Rtol: 1e-4

SA: Off

PrimaryReactionLibrary: on
Name: Leeds
Location: database\primaryReactionLibrary\combustion_core\version4

Figure 8.1 Initialization file for n-butane low temperature oxidation
8.3 Results

Given the input information, RMG generated a model with 68 species and 835 reactions. The model edge includes 1891 species and 3637 reactions. The total running time required to generate this model is \(~30\) min on a 1.70GHz PC with 512Mb memory.

8.3.1 N-butane low-temperature oxidation model

This core model includes 68 Species and 835 Reactions. The author lists all the species each with its ID number, name, chemical formula, and simplified adjacency list representation, (i.e., all the hydrogen atoms are omitted). Each reaction is listed with its reactants and products, direction, type, and kinetics parameters. Notice that there are two types of reactions: pressure-independent reactions, listed in Table 8.1, whose kinetics rates are calculated from Arrhenius parameters, \(A, n, E\); and pressure-dependent reactions, listed in Table 8.2, whose rates are estimated by CHEMDIS [Chang et al., 2000] using the algorithm of Matheu, et al.[2003]. The resulting \(k(T,P)\) of those reactions from CHEMDIS are listed.
Species List

Species 1  Name: C4H10
ChemFormula: C4H10
1 C 0 {2,S}
2 C 0 {3,S} {1,S}
3 C 0 {2,S} {4,S}
4 C 0 {3,S}

Species 2  Name: O2
ChemFormula: O2
1 O 0 {2,D}
2 O 0 {1,D}

Species 3  Name: H2
ChemFormula: H2
1 H 0 {2,S}
2 H 0 {1,S}

Species 4  Name: CH4
ChemFormula: CH4
1 C 0

Species 5  Name: C2H2
ChemFormula: C2H2
1 C 0 {2,T}
2 C 0 {1,T}

Species 6  Name: C2H4
ChemFormula: C2H4
1 C 0 {2,D}
2 C 0 {1,D}

Species 7  Name: C2H6
ChemFormula: C2H6
1 C 0 {2,S}
2 C 0 {1,S}

Species 8  Name: C3H4
ChemFormula: C3H4
1 C 0 {2,D}
2 C 0 {3,D} {1,D}
3 C 0 {2,D}

Species 9  Name: C3H6
ChemFormula: C3H6
1 C 0 {2,D}
2 C 0 {1,D} {3,S}
3 C 0 {2,S}

Species 10 Name: C4H2
ChemFormula: C4H2
1 C 0 {2,T}
2 C 0 {1,T} {3,S}
3 C 0 {4,T} {2,S}
4 C 0 {3,T}
<table>
<thead>
<tr>
<th>Species</th>
<th>Name</th>
<th>ChemFormula</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>H2O</td>
<td>H2O</td>
</tr>
<tr>
<td>12</td>
<td>H2O2</td>
<td>H2O2</td>
</tr>
<tr>
<td>13</td>
<td>CO</td>
<td>CO</td>
</tr>
<tr>
<td>14</td>
<td>CO2</td>
<td>CO2</td>
</tr>
<tr>
<td>15</td>
<td>CH2O</td>
<td>CH2O</td>
</tr>
<tr>
<td>16</td>
<td>CH2CO</td>
<td>C2H2O</td>
</tr>
<tr>
<td>17</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>18</td>
<td>CH</td>
<td>CH</td>
</tr>
<tr>
<td>19</td>
<td>CH2</td>
<td>CH2</td>
</tr>
<tr>
<td>20</td>
<td>CH2(S)</td>
<td>CH2</td>
</tr>
<tr>
<td>21</td>
<td>CH3</td>
<td>CH3</td>
</tr>
<tr>
<td>22</td>
<td>C2H</td>
<td>C2H</td>
</tr>
</tbody>
</table>
Species 23  Name: C2H3  
ChemFormula: C2H3.  
1  C 1 {2,D}  
2  C 0 {1,D}

Species 24  Name: C2H5  
ChemFormula: C2H5.  
1  C 1 {2,S}  
2  C 0 {1,S}

Species 25  Name: C3H2  
ChemFormula: C3H2.  
1  C 1 {2,D}  
2  C 0 {1,D} {3,D}  
3  C 1 {2,D}  
isomer2:  
ChemFormula: C3H2.  
1  C 2 {2,S}  
2  C 0 {3,T} {1,S}  
3  C 0 {2,T}

Species 26  Name: H2CCCH  
ChemFormula: C3H3.  
1  C 1 {2,S}  
2  C 0 {1,S} {3,T}  
3  C 0 {2,T}  
isomer1:  
ChemFormula: C3H3.  
1  C 0 {2,D}  
2  C 0 {1,D} {3,D}  
3  C 1 {2,D}

Species 27  Name: H2CCCH  
ChemFormula: C4H3.  
1  C 0 {2,D}  
2  C 0 {1,D} {3,D}  
3  C 0 {4,D} {2,D}  
4  C 1 {3,D}  
isomer2:  
ChemFormula: C4H3.  
1  C 0 {2,D}  
2  C 1 {1,D} {3,S}  
3  C 0 {4,T} {2,S}  
4  C 0 {3,T}

Species 28  Name: O  
ChemFormula: O..  
1  O 2T

Species 29  Name: OH  
ChemFormula: HO.  
1  O 1
Species 30  Name: HO2
ChemFormula: HO2.
1 O 1 {2,S}
2 O 1 {1.S}

Species 31  Name: HCO
ChemFormula: CHO.
1 C 1 {2,D}
2 O 0 {1.D}

Species 32  Name: CH3O
ChemFormula: CH3O.
1 C 1 {2,S}
2 O 1 {1.S}

Species 33  Name: CH2OH
ChemFormula: CH3O.
1 C 1 {2,S}
2 O 0 {1.S}

Species 34  Name: HCCO
ChemFormula: C2HO.
1 C 1 {2,D}
2 C 0 {3,D} {1,D}
3 O 0 {2,D}

isomer1:
ChemFormula: C2HO.
1 C 0 {2,T}
2 C 0 {1,T} {3,S}
3 O 1 {2,S}

Species 35  Name: CH2HCO
ChemFormula: C2H3O.
1 C 0 {2,D}
2 C 0 {3,S} {1,D}
3 O 1 {2,S}

isomer2:
ChemFormula: C2H3O.
1 C 1 {2,S}
2 C 0 {1,S} {3,D}
3 O 0 {2,D}

Species 36  Name: C3H5
ChemFormula: C3H5.
1 C 0 {3,S} {2,D}
2 C 0 {1,D}
3 C 1 {1.S}

Species 37  Name: C4H9_1
1 C 0 {2,S}
2 C 0 {3,S} {1.S}
3 C 0 {2,S} {4,S}
4 C 1 {3,S}
Species 38  Name: C4H9_2
  1 C 0 {2,S}
  2 C 0 {3,S} {1,S}
  3 C 1 {2,S} {4,S}
  4 C 0 {3,S}

Species 39  Name: C3H7.
ChemFormula: C3H7.
  1 C 0 {7,S} {6,S}
  6 C 0 {1,S}
  7 C 1 {1,S}

Species 44  Name: C4H8
ChemFormula: C4H8
  1 C 0 {7,S} {8,S}
  2 C 0 {8,D}
  7 C 0 {1,S}
  8 C 0 {2,D} {1,S}

Species 45  Name: C4H8
ChemFormula: C4H8
  1 C 0 {8,S} {9,D}
  2 C 0 {9,S}
  8 C 0 {1,S}
  9 C 0 {2,S} {1,D}

Species 49  Name: C4H9O2.
ChemFormula: C4H9O2.
  2 O 0 {15,S} {9,S}
  4 C 0 {11,S} {15,S}
  8 C 0 {15,S}
  9 O 1 {2,S}
  11 C 0 {4,S}
  15 C 0 {2,S} {8,S} {4,S}

Species 52  Name: C2H5O2.
ChemFormula: C2H5O2.
  1 O 0 {7,S} {6,S}
  2 C 0 {7,S}
  6 O 1 {1,S}
  7 C 0 {1,S} {2,S}

Species 54  Name: CH3O2.
ChemFormula: CH3O2.
  1 O 0 {5,S} {4,S}
  4 O 1 {1,S}
  5 C 0 {1,S}

Species 58  Name: C4H9O2.
ChemFormula: C4H9O2.
  2 O 0 {9,S} {8,S}
  4 C 0 {11,S} {15,S}
  8 C 0 {2,S} {15,S}
  9 O 1 {2,S}
Species 61  Name: C2H3O.
ChemFormula: C2H3O.
1 C 1 [5,D] {4,S}
4 C 0 {1,S}
5 O 0 {1,D}

Species 121 Name: C2H4O2
ChemFormula: C2H4O2
1 C 0 [6,S] {5,D}
5 C 0 {1,D}
6 O 0 {1,S} {7,S}
7 O 0 [6,S]

Species 168 Name: C2H6O2
ChemFormula: C2H6O2
1 C 0 [6,S]
3 O 0 [6,S] {4,S}
4 O 0 [3,S]
6 C 0 [3,S] {1,S}

Species 240 Name: CH4O2
ChemFormula: CH4O2
2 O 0 [5,S] {4,S}
4 O 0 [2,S]
5 C 0 [2,S]

Species 278 Name: C4H6
ChemFormula: C4H6
1 C 0 [6,D]
5 C 0 [9,D] {6,S}
6 C 0 [5,S] {1,D}
9 C 0 [5,D]

Species 400 Name: C3H5O2.
ChemFormula: C3H5O2.
1 C 0 [7,S] {6,D} {3,S}
3 O 0 [4,S] {1,S}
4 O 0 [3,S]
6 C 0 [1,D]
7 C 1 {1,S}

Species 505 Name: C2H3O3.
ChemFormula: C2H3O3.
1 C 0 [6,D] {4,S} {5,S}
4 O 0 [7,S] {1,S}
5 C 1 {1,S}
6 O 0 [1,D]
7 O 0 {4,S}

isomer2:
ChemFormula: C2H3O3.
1 C 0 [6,S] {4,S} {5,D}
4 O 0 {1,S} {7,S}
Species 592  Name: C2H5O.
ChemFormula: C2H5O.
1  O 1 [5,S]
5  C 0 [1,S] {8,S}
8  C 0 [5,S]

Species 599  Name: C4H7O.
ChemFormula: C4H7O.
1  O 1 [7,S]
6  C 0 [11,D]
7  C 0 [1,S] {9,S}
9  C 0 [11,S] {7,S}
11  C 0 [6,D] {9,S}

Species 600  Name: C2H3O3.
ChemFormula: C2H3O3.
1  O 0 [5,S] [6,S]
2  C 0 [6,S] {8,D}
5  O 1 [1,S]
6  C 0 [1,S] {2,S}
8  O 0 [2,D]

Species 605  Name: C3H3O2.
ChemFormula: C3H3O2.
1  O 0 [8,S] [5,S]
2  C 0 [8,D] {6,D}
5  O 1 [1,S]
6  C 0 [2,D]
8  C 0 [2,D] {1,S}

Species 610  Name: C2H3O2.
1  O 0 [4,S] [5,S]
2  C 0 [5,D]
4  O 1 [1,S]
5  C 0 [2,D] {1,S}

Species 611  Name: C3H5O2.
ChemFormula: C3H5O2.
1  O 0 [10,S] [6,S]
2  C 0 [7,D]
6  O 1 [1,S]
7  C 0 [10,S] {2,D}
10  C 0 [7,S] {1,S}

Species 626  Name: C4H9O2.
ChemFormula: C4H9O2.
1  O 0 [5,S] [6,S]
2  C 0 [5,S] [3,S]
3  C 1 [8,S] {2,S}
5  C 0 [2,S] {1,S}
6  O 0 [1,S]
Species 711  Name: C4H7.
ChemFormula: C4H7.
1 C 1 {2,S}
2 C 0 {1,S} {6,D}
6 C 0 {8,S} {2,D}
8 C 0 {6,S} 

isomer2:
ChemFormula: C4H7.
1 C 0 {2,D}
2 C 0 {6,S} {1,D}
6 C 1 {2,S} {8,S}
8 C 0 {6,S} 

Species 789  Name: C4H8O2
ChemFormula: C4H8O2
1 C 0 {13,S} {9,S}
4 C 0 {9,D}
6 O 0 {8,S}
8 O 0 {13,S} {6,S}
9 C 0 {4,D} {1,S}
13 C 0 {1,S} {8,S} 

Species 901  Name: C4H9O4.
2 C 0 {17,S} {7,S}
3 O 0 {6,S}
6 O 0 {7,S} {3,S}
7 C 0 {6,S} {2,S}
10 O 0 {17,S} {14,S}
14 O 1 {10,S}
16 C 0 {17,S}
17 C 0 {2,S} {16,S} {10,S} 

Species 1133  Name: C3H5O4.
ChemFormula: C3H5O4.
1 O 0 {7,S} {3,S}
3 C 0 {8,S} {1,S}
4 C 0 {8,D}
6 O 0 {12,S}
7 O 1 {1,S}
8 C 0 {3,S} {4,D} {12,S}
12 O 0 {6,S} {8,S} 

Species 1294  Name: C2H3O5.
ChemFormula: C2H3O5.
1 O 0 {6,S} {8,S}
3 O 0 {5,S}
4 O 0 {7,D}
5 O 0 {7,S} {3,S}
6 O 1 {1,S}
7 C 0 {8,S} {5,S} {4,D}
8 C 0 {7,S} {1,S}
Species 1422  Name: C5H7O2.
ChemFormula: C5H7O2.
1  O 0 {8,D}
3  C 0 {14,D}
5  C 0 {14,S} {10,S} {7,S}
7  C 0 {5,S}
8  C 1 {10,S} {1,D}
10 O 0 {5,S} {8,S}
14 C 0 {5,S} {3,D}

Species 1433  Name: C5H9O.
ChemFormula: C5H9O.
2  O 0 {13,S} {9,S}
8  C 0 {13,S} {14,D}
9  C 1 {2,S}
13 C 0 {8,S} {15,S} {2,S}
14 C 0 {8,D}
15 C 0 {13,S}

Species 1435  Name: C4H7O2.
ChemFormula: C4H7O2.
1  O 0 {9,S} {8,S}
3  C 0 {5,D} {9,S}
5  C 0 {11,S} {3,D}
8  O 1 {1,S}
9  C 0 {1,S} {3,S}
11 C 0 {5,S}

Species 1436  Name: C4H7O2.
ChemFormula: C4H7O2.
1  O 0 {8,S} {5,S}
3  C 0 {5,S} {9,D}
5  C 0 {3,S} {1,S} {11,S}
8  O 1 {1,S}
9  C 0 {3,D}
11 C 0 {5,S}
Table 8.1 Non pressure-dependent reaction list

<table>
<thead>
<tr>
<th>Reaction</th>
<th>For/Back</th>
<th>Type</th>
<th>A (cm³-mol⁻¹-s⁻¹)</th>
<th>n</th>
<th>alpha</th>
<th>E (Kcal/mol)</th>
<th>k (T=715K, P=0.73atm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>C₂H(22) + C₂H(22) → C₄H₂(10)</td>
<td>Direction = 1</td>
<td>R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0</td>
<td>0.292</td>
<td>4.557E+12</td>
</tr>
<tr>
<td>C₂H(22) + CH₂(19) → C₂H₂(5) + CH(18)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>2.11E+14</td>
<td>0</td>
<td>0</td>
<td>-0.122</td>
<td>3.058E+22</td>
</tr>
<tr>
<td>C₂H(22) + CH₂O(15) → CH₂(19) + HCO(34)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
<td>2.000</td>
<td>1.385E+03</td>
</tr>
<tr>
<td>C₂H(22) + H₂O(11) → C₂H₂(5) + OH(29)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>6.00E+13</td>
<td>0</td>
<td>0</td>
<td>12.910</td>
<td>8.886E+12</td>
</tr>
<tr>
<td>C₂H(22) + O(28) → CH(18) + CO(13)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.000E+13</td>
</tr>
<tr>
<td>C₂H(22) + OH(29) → CH₂(19) + CO(13)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.810E+13</td>
</tr>
<tr>
<td>C₂H(22) + OH(29) → HCCO(34) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>2.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>2.000E+13</td>
</tr>
<tr>
<td>C₂H(22) + H₂O(11) → C₂H₂(5) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>9.03E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>9.030E+13</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₂(5) → H₂CCCH(27) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>2.00E+09</td>
<td>0</td>
<td>0</td>
<td>57.812</td>
<td>4.252E-09</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₃(23) → C₂H₂(22) + C₂H₄(6)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>8.85E+09</td>
<td>0.6961</td>
<td>0</td>
<td>9.953</td>
<td>1.354E+02</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₃O₂(610) → C₂H(22) + C₂H₄O₂(121)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.902E-01</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₅(24) → C₂H₂(22) + C₂H₆(7)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>3.61E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>3.941E-02</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₅O₂(52) → C₂H(22) + C₂H₆O₂(168)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.902E-01</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₃H₂(25) → C₂H₂(22) + H₂CCCH(26)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>3.72E+09</td>
<td>1.51</td>
<td>0</td>
<td>21.428</td>
<td>2.139E+07</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₅(36) → C₂H₂(22) + C₃H₆(9)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.81E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>5.553E-01</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₄H₇(71) → C₂H₄H₈(44) + C₂H₂(22)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>8.029E-02</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₄H₇(71) → C₂H₄H₈(45) + C₂H₂(22)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>3.61E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>5.553E-01</td>
</tr>
<tr>
<td>C₂H₂(5) + C₂H₄H₉_1(37) → C₄H₁₀(1) + C₂H₂(22)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>3.61E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>4.024E+02</td>
</tr>
<tr>
<td>C₂H(22) + C₂H₄9_2(38) → C₄H₁₀(1) + C₂H₂(22)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>2.42E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>2.119E+01</td>
</tr>
<tr>
<td>C₂H(22) + CH(18) → C₂H₂(22) + CH₂(19)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>2.11E+14</td>
<td>0</td>
<td>0</td>
<td>-0.122</td>
<td>2.299E+14</td>
</tr>
<tr>
<td>C₂H₂(5) + CH₂(19) → C₂H₂(22) + CH₃(21)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>2.35E+10</td>
<td>0.8291</td>
<td>0</td>
<td>28.328</td>
<td>1.199E+04</td>
</tr>
<tr>
<td>C₂H₂(5) + CH₂(19) → C₃H₄(8)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.20E+13</td>
<td>0</td>
<td>0</td>
<td>6.615</td>
<td>1.140E+11</td>
</tr>
<tr>
<td>C₂H(22) + CH₂(8)(20) → H₂CCCH(26) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.75E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.750E+14</td>
</tr>
<tr>
<td>C₂H(22) + CH₃(21) → C₂H₂(22) + CH₄(4)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.81E+12</td>
<td>0</td>
<td>0</td>
<td>0.500</td>
<td>6.619E+03</td>
</tr>
<tr>
<td>C₂H(22) + CH₃(21) → CH₂(19) + C₂H₃(23)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.051E-08</td>
</tr>
<tr>
<td>C₂H(22) + CH₃O₂(54) → C₂H₂(22) + CH₄O₂(240)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.902E-01</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Location</td>
<td>k</td>
<td>E_m</td>
<td>A</td>
<td>E_Ab</td>
<td>A_Ab</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-----------</td>
<td>----------</td>
<td>-------</td>
<td>----------</td>
<td>---------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>C2H2(5) + CO(13) + CO(13) -&gt; HCCO(34) + HCCO(34)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>3.106E-17</td>
</tr>
<tr>
<td>C2H2(5) + CO(13) + H(17) -&gt; H2CCCH(26) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.39E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>3.551E-06</td>
</tr>
<tr>
<td>C2H2(5) + CO(13) -&gt; CH(18) + HCCO(34)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>5.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>5.709E-35</td>
</tr>
<tr>
<td>C2H2(5) + H(17) + H(17) -&gt; CH2(19) + CH2(19)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.08E+14</td>
<td>0</td>
<td>0</td>
<td>0.796</td>
<td>7.854E+07</td>
</tr>
<tr>
<td>C2H2(5) + H(17) -&gt; C2H2(22) + H(3)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.08E+13</td>
<td>0</td>
<td>0</td>
<td>2.170</td>
<td>1.913E+05</td>
</tr>
<tr>
<td>C2H2(5) + H(17) -&gt; C2H3(23)</td>
<td>Direction = 1</td>
<td>R Addition MultipleBond</td>
<td>5.50E+12</td>
<td>0</td>
<td>0</td>
<td>2.420</td>
<td>1.001E+12</td>
</tr>
<tr>
<td>C2H2(5) + H(17) -&gt; CH(18) + CH2(19)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>4.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>2.924E-23</td>
</tr>
<tr>
<td>C2H2(5) + H2(3) -&gt; C2H4(6)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>9.97E+16</td>
<td>0</td>
<td>0</td>
<td>71.505</td>
<td>2.071E+06</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>5.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>5.709E-35</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.20E+13</td>
<td>0</td>
<td>0</td>
<td>0.796</td>
<td>4.617E-25</td>
</tr>
<tr>
<td>C2H2(5) + O2(28) + CO(13) -&gt; CH2(19) + CO(13)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>2.17E+06</td>
<td>2.1</td>
<td>0</td>
<td>1.570</td>
<td>7.091E+11</td>
</tr>
<tr>
<td>C2H2(5) + O2(28) + CO(13) -&gt; CH2(19) + CO(13)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>5.06E+06</td>
<td>0</td>
<td>1</td>
<td>1.570</td>
<td>1.654E+12</td>
</tr>
<tr>
<td>C2H2(5) + O2(28) + CO(13) -&gt; CH2(19) + CO(13)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.44E+08</td>
<td>0</td>
<td>12.910</td>
<td>6.789E+09</td>
<td></td>
</tr>
<tr>
<td>C2H2(5) + O2(28) + CO(13) -&gt; CH2(19) + CO(13)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.40E+08</td>
<td>0</td>
<td>1.5</td>
<td>6.600</td>
<td>7.046E+03</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>8.85E+09</td>
<td>0</td>
<td>2.420</td>
<td>2.505E-05</td>
<td></td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.40E+08</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>7.230E+13</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>8.478E-16</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>3.778E-02</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>4.33E+10</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>4.740E+05</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>3.000E+13</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>4.52E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.987E+06</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.44E+01</td>
<td>0</td>
<td>0</td>
<td>6.940</td>
<td>6.941E+01</td>
</tr>
<tr>
<td>C2H2(5) + H2CCCH(26) + C2H2(22) + C3H4(8)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>2.974E+07</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>R_Accession MultipleBond</td>
<td>Energetic Barriers</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>--------------------------</td>
<td>-------------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>C2H4(6) + C2H5(24) → C4H9(1)(37)</td>
<td>Direction = 1</td>
<td>R_Accession MultipleBond</td>
<td>3.98E+03</td>
<td>2.44</td>
<td>0</td>
<td>5.370</td>
<td>8.372E+08</td>
</tr>
<tr>
<td>C2H4(6) + C2H5O2(52) → C2H6O2(168) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
<td>6.941E+01</td>
</tr>
<tr>
<td>C2H4(6) + C3H2(25) → C2H3(23) + H2CCH(26)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.18E+14</td>
<td>0</td>
<td>0</td>
<td>13.100</td>
<td>1.164E+10</td>
</tr>
<tr>
<td>C2H4(6) + C3H5(36) → C2H3(23) + C2H6(9)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>2.32E+13</td>
<td>0</td>
<td>0</td>
<td>7.500</td>
<td>2.091E+05</td>
</tr>
<tr>
<td>C2H4(6) + C4H7(71) → C4H8(44) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.02E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
<td>5.528E+02</td>
</tr>
<tr>
<td>C2H4(6) + C4H9(1)(37) → C4H10(1) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>4.64E+13</td>
<td>0</td>
<td>0</td>
<td>7.500</td>
<td>2.091E+05</td>
</tr>
<tr>
<td>C2H4(6) + C4H9(2)(38) → C4H10(1) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>3.037E+07</td>
</tr>
<tr>
<td>C2H4(6) + C3H2(25) → C2H3(23) + C2H6(9)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
<td>1.459E+05</td>
</tr>
<tr>
<td>C2H4(6) + C4H7(71) → C4H8(44) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
<td>6.941E+01</td>
</tr>
<tr>
<td>C2H4(6) + C3H2(25) → C2H3(23) + C2H6(9)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>9.64E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>9.640E+13</td>
</tr>
<tr>
<td>C2H4(6) + C4H7(71) → C4H8(44) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>2.32E+13</td>
<td>0</td>
<td>0</td>
<td>7.500</td>
<td>2.091E+05</td>
</tr>
<tr>
<td>C2H4(6) + C4H9(1)(37) → C4H10(1) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>3.037E+07</td>
</tr>
<tr>
<td>C2H4(6) + C4H9(2)(38) → C4H10(1) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
<td>1.459E+05</td>
</tr>
<tr>
<td>C2H4(6) + C3H2(25) → C2H3(23) + C2H6(9)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
<td>6.941E+01</td>
</tr>
<tr>
<td>C2H4(6) + C4H7(71) → C4H8(44) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>4.64E+13</td>
<td>0</td>
<td>0</td>
<td>7.500</td>
<td>2.091E+05</td>
</tr>
<tr>
<td>C2H4(6) + C4H9(1)(37) → C4H10(1) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>3.037E+07</td>
</tr>
<tr>
<td>C2H4(6) + C4H9(2)(38) → C4H10(1) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
<td>1.459E+05</td>
</tr>
<tr>
<td>C2H4(6) + C3H2(25) → C2H3(23) + C2H6(9)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
<td>6.941E+01</td>
</tr>
<tr>
<td>C2H4(6) + C4H7(71) → C4H8(44) + C2H3(23)</td>
<td>Direction = -1</td>
<td>H_Abstraction</td>
<td>4.64E+13</td>
<td>0</td>
<td>0</td>
<td>7.500</td>
<td>2.091E+05</td>
</tr>
</tbody>
</table>

229
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Direction</th>
<th>Type</th>
<th>E0</th>
<th>E1</th>
<th>E2</th>
<th>E3</th>
</tr>
</thead>
<tbody>
<tr>
<td>C2H4O2(121) + C2H3(23) -&gt; C2H3O2(610) + C2H4(6)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>C2H4O2(121) + C2H5(24) -&gt; C2H6(7) + C2H3O2(610)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H4O2(121) + C2H5O2(52) -&gt; C2H6O2(168) + C2H3O2(610)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>C2H4O2(121) + C3H2(25) -&gt; C2H3O2(610) + H2CCCH(26)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.88E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>C2H4O2(121) + C3H5(36) -&gt; C3H6(9) + C2H3O2(610)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H4O2(121) + C4H7(121) -&gt; C4H8(44) + C2H3O2(610)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>C2H5(24) + C2H5(24) -&gt; C4H10(1)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.15E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C2H5(24) + CH4(4) -&gt; C2H6(7) + CH3(21)</td>
<td>-1</td>
<td>R_Recombination</td>
<td>1.00E-07</td>
<td>6</td>
<td>0</td>
<td>6.044</td>
</tr>
<tr>
<td>C2H5(24) + H(17) -&gt; C2H6(7)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.00E-14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Energy</td>
<td>Activation</td>
<td>Products</td>
<td>Products</td>
<td>Reaction</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>--------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>C2H5(24) + H(17) -&gt; CH3(21) + CH3(21)</td>
<td>-1 Leeds</td>
<td>3.01E+13</td>
<td>0</td>
<td>0</td>
<td>13.507</td>
<td>9.159E+13</td>
</tr>
<tr>
<td>C2H5(24) + H2(3) -&gt; C2H6(7) + H(17)</td>
<td>-1 Leeds</td>
<td>1.45E+09</td>
<td>1.5</td>
<td>0</td>
<td>7.408</td>
<td>2.016E+08</td>
</tr>
<tr>
<td>C2H5(24) + H2O(11) -&gt; C2H6(7) + OH(29)</td>
<td>-1 Leeds</td>
<td>7.23E+06</td>
<td>2</td>
<td>0</td>
<td>0.865</td>
<td>2.872E+05</td>
</tr>
<tr>
<td>C2H5(24) + HO2(30) -&gt; C2H6O2(168)</td>
<td>1 R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0</td>
<td>0.292</td>
<td>4.557E+12</td>
</tr>
<tr>
<td>C2H5(24) + O(28) -&gt; CH2O(15) + CH3(21)</td>
<td>-1 Leeds</td>
<td>6.62E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>6.620E+13</td>
</tr>
<tr>
<td>C2H5(24) + OH(29) -&gt; C2H6(7) + O(28)</td>
<td>-1 Leeds</td>
<td>1.00E+09</td>
<td>1.5</td>
<td>0</td>
<td>5.800</td>
<td>7.219E+08</td>
</tr>
<tr>
<td>C2H5O(392) + OH(29) -&gt; C2H6O2(168)</td>
<td>-1 Leeds</td>
<td>2.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>2.000E+13</td>
</tr>
<tr>
<td>C2H6(7) + C2H(22) -&gt; C2H2(5) + C2H5(24)</td>
<td>1 H_Abstraction</td>
<td>3.61E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>3.612E+12</td>
</tr>
<tr>
<td>C2H6(7) + C2H3(23) -&gt; C2H5(24) + C2H4(6)</td>
<td>-1 H_Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>4.737E+10</td>
</tr>
<tr>
<td>C2H6(7) + C2H3O2(610) -&gt; C2H4O2(121) + C2H5(24)</td>
<td>-1 H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
<td>1.510E+04</td>
</tr>
<tr>
<td>C2H6(7) + C2H5O2(52) -&gt; C2H6O2(168) + C2H5(24)</td>
<td>-1 H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
<td>1.510E+04</td>
</tr>
<tr>
<td>C2H6(7) + C3H2(25) -&gt; C2H5(24) + H2CCCH(26)</td>
<td>1 H_Abstraction</td>
<td>6.04E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>9.475E+10</td>
</tr>
<tr>
<td>C2H6(7) + C3H5(36) -&gt; C3H6(9) + C2H5(24)</td>
<td>-1 H_Abstraction</td>
<td>1.68E+12</td>
<td>0</td>
<td>0</td>
<td>12.400</td>
<td>7.668E+05</td>
</tr>
<tr>
<td>C2H6(7) + C4H7(711) -&gt; C4H8(44) + C2H5(24)</td>
<td>-1 H_Abstraction</td>
<td>3.04E+06</td>
<td>1.885</td>
<td>0</td>
<td>9.448</td>
<td>5.772E+05</td>
</tr>
<tr>
<td>C2H6(7) + C4H9(1) -&gt; C4H1O(1) + C2H5(24)</td>
<td>H A bstracti o n</td>
<td>1.49E+03</td>
<td>2.71</td>
<td>0</td>
<td>12.920</td>
<td>2.414E+07</td>
</tr>
<tr>
<td>C2H6(7) + H(17) -&gt; CH3(21) + H(17)</td>
<td>1 Leeds</td>
<td>1.45E+09</td>
<td>1.5</td>
<td>0</td>
<td>7.408</td>
<td>5.08E+11</td>
</tr>
<tr>
<td>C2H6(7) + H2CCCH(26) -&gt; C2H5(24) + C3H4(8)</td>
<td>-1 H_Abstraction</td>
<td>3.13E+13</td>
<td>0</td>
<td>0</td>
<td>19.700</td>
<td>8.762E+03</td>
</tr>
<tr>
<td>C2H6(7) + HCCO(34) -&gt; C2H5(24) + CH2CO(16)</td>
<td>-1 H_Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
<td>4.737E+10</td>
</tr>
<tr>
<td>C2H6(7) + HCO(31) -&gt; C2H5(24) + CH2O(28)</td>
<td>-1 H_Abstraction</td>
<td>5.50E+03</td>
<td>2.81</td>
<td>0</td>
<td>5.860</td>
<td>6.582E+06</td>
</tr>
<tr>
<td>C2H6(7) + O(28) -&gt; C2H5(24) + OH(29)</td>
<td>-1 Leeds</td>
<td>1.32E+13</td>
<td>0</td>
<td>0</td>
<td>20.456</td>
<td>7.367E+06</td>
</tr>
<tr>
<td>C2H6(7) + OH(29) -&gt; C2H5(24) + H2O(11)</td>
<td>-1 Leeds</td>
<td>7.23E+06</td>
<td>2</td>
<td>0</td>
<td>0.865</td>
<td>2.011E+12</td>
</tr>
<tr>
<td>C2H6O2(168) + C2H(22) -&gt; C2H5O2(52) + C2H2(5)</td>
<td>-1 H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.210E+12</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>------</td>
<td>----------------</td>
<td>-----</td>
<td>-------------</td>
<td>-------------</td>
</tr>
<tr>
<td>C2H6O2(168) + C2H3(23) -&gt; C2H4(6) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>C2H6O2(168) + C2H3O2(610) -&gt; C2H4O2(121) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>C2H6O2(168) + C2H5(24) -&gt; C2H6(7) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H6O2(168) + C2H2(25) -&gt; H2CCCH(26) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.88E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>C2H6O2(168) + C3H5(36) -&gt; C3H6(9) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H6O2(168) + C4H7(711) -&gt; C4H8(44) + C2H5O2(52)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>0</td>
<td>5.105</td>
</tr>
<tr>
<td>C2H6O2(168) + C4H7(711) -&gt; C4H8(45) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H6O2(168) + C4H9_1(37) -&gt; C4H10(1) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H6O2(168) + C4H9_2(38) -&gt; C4H10(1) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H6O2(168) + CH2(19) -&gt; C2H5O2(52) + CH3(21)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>C2H6O2(168) + CH3(21) -&gt; CH4(4) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C2H6O2(168) + CH3O2(54) -&gt; CH4O2(240) + C2H5O2(52)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>C2H6O2(168) + CH2O(15) -&gt; C2H5O2(52) + H2(3)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.06E+10</td>
<td>1.105</td>
<td>0</td>
<td>8.910</td>
</tr>
<tr>
<td>C2H6O2(168) + H2CCCH(26) -&gt; C3H4(8) + C2H5O2(52)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>0</td>
<td>0</td>
<td>5.940</td>
</tr>
<tr>
<td>C2H6O2(168) + HCCO(34) -&gt; C2H5O2(52) + CH2CO(16)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>C2H6O2(168) + HCO(31) -&gt; C2H5O2(52) + CH2O(15)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>4.12E+04</td>
<td>2.5</td>
<td>0</td>
<td>10.210</td>
</tr>
<tr>
<td>C2H6O2(168) + HO(30) -&gt; C2H5O2(52) + H2O(12)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>4.82E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>C2H6O2(168) + O(28) -&gt; C2H5O2(52) + OH(29)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
<td>4.690</td>
</tr>
<tr>
<td>C2H6O2(168) + OH(29) -&gt; C2H5O2(52) + H2O(11)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>C3H2(25) + CO(13) -&gt; C4H2(10) + O(28)</td>
<td>-1</td>
<td>Leeds</td>
<td>7.89E+12</td>
<td>0</td>
<td>0</td>
<td>1.347</td>
</tr>
<tr>
<td>C3H2(25) + HCO(31) -&gt; C4H2(10) + OH(29)</td>
<td>-1</td>
<td>Leeds</td>
<td>6.68E+12</td>
<td>0</td>
<td>0</td>
<td>-0.409</td>
</tr>
<tr>
<td>C3H3O2(605) -&gt; H2CCCH(26) + O2(2)</td>
<td>-1</td>
<td>Beta_Scission</td>
<td>1.65E+12</td>
<td>0.1474</td>
<td>0</td>
<td>2.533</td>
</tr>
<tr>
<td>C3H4(8) + C2H2(22) -&gt; H2CCCH(26) + C2H2(5)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>8.85E+09</td>
<td>0.6961</td>
<td>0</td>
<td>9.953</td>
</tr>
<tr>
<td>C3H4(8) + C2H3(23) -&gt; H2CCCH(26) + C2H4(6)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.88E+13</td>
<td>0</td>
<td>0</td>
<td>13.100</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>H_Abstraction</td>
<td>E in kcal/mol</td>
<td>Error in kcal/mol</td>
<td>E in Hz</td>
<td>Error in Hz</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------------</td>
<td>---------------</td>
<td>------------------</td>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>C3H4(8) + C2H3O2(610) -&gt; H2CCCH(26) + C2H4O2(121)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>5.940</td>
<td>3.136E+11</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + C2H5(24) -&gt; H2CCCH(26) + C2H6(7)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.13E+13</td>
<td>19.700</td>
<td>2.973E+07</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + C2H5O2(52) -&gt; C2H6O2(168) + H2CCCH(26)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>5.940</td>
<td>3.136E+11</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + C3H2(25) -&gt; H2CCCH(26) + C2H6(7)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.13E+13</td>
<td>19.700</td>
<td>2.973E+07</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + C3H5(36) -&gt; C2H6O2(168) + H2CCCH(26)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>5.940</td>
<td>3.136E+11</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + CH2(19) -&gt; H2CCCH(26) + CH3(21)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.91E+09</td>
<td>11.780</td>
<td>1.137E+10</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + HCCO(34) -&gt; H2CCCH(26) + C2H4O2(121)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>5.940</td>
<td>3.136E+11</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + CH3(21) -&gt; H2CCCH(26) + CH4(4)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.30E+04</td>
<td>10.040</td>
<td>1.531E+07</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + CH302(54) -&gt; H2CCCH(26) + CH4O2(168)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>5.940</td>
<td>3.136E+11</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + CH5(17) -&gt; C3H5(36)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>2.40E+11</td>
<td>3.000</td>
<td>2.708E+12</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + H(17) -&gt; H2(3) + H2CCCH(26)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.01E+08</td>
<td>11.780</td>
<td>1.137E+10</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + HCCO(31) -&gt; H2CCCH(26) + H2O(15)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.88E+13</td>
<td>13.100</td>
<td>5.819E+09</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + HO2(30) -&gt; H2CCCH(26) + H2O2(12)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>8.85E+09</td>
<td>9.953</td>
<td>7.791E+08</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + CH2O(28) -&gt; H2CCCH(26) + OH(29)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.88E+01</td>
<td>6.940</td>
<td>1.474E+08</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + OH(29) -&gt; H2CCCH(26) + H2O(11)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.51E+07</td>
<td>3.740</td>
<td>3.076E+11</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + C2H5(24) -&gt; H2CCCH(26) + CH2(19)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.20E+13</td>
<td>6.615</td>
<td>8.383E-17</td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + C2H6(17) -&gt; C3H5(36)</td>
<td>1</td>
<td>Bond_Dissociation</td>
<td>1.21E+14</td>
<td>0.000</td>
<td>1.624E-11</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C2H2(22) -&gt; C3H5(36) + C2H2(5)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.81E+12</td>
<td>0.000</td>
<td>1.806E+12</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C2H3(23) -&gt; C3H5(36) + C2H4(6)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.32E+13</td>
<td>7.500</td>
<td>1.182E+11</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C2H3O2(610) -&gt; C2H4O2(121) + C3H5(36)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>9.690</td>
<td>5.358E+06</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C2H5(24) -&gt; C2H6(7) + C3H5(36)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.68E+12</td>
<td>12.400</td>
<td>2.721E+08</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C2H5O2(52) -&gt; C2H6O2(168) + C3H5(36)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>9.690</td>
<td>5.358E+06</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C3H2(25) -&gt; C3H5(36) + H2CCCH(26)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>4.64E+13</td>
<td>7.500</td>
<td>2.364E+11</td>
<td></td>
</tr>
<tr>
<td>C3H6(9) + C4H7(711) -&gt; C4H8(44) + C3H5(36)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>9.448</td>
<td>2.048E+08</td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Type</td>
<td>Energy (kcal/mol)</td>
<td>Enthalpy (kcal/mol)</td>
<td>Entropy (cal/mol K)</td>
<td>Gas Constant (kcal/mol K)</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>---------------------</td>
<td>---------------------</td>
<td>---------------------------</td>
</tr>
<tr>
<td>C3H6(9) + C4H7._1(37) -&gt; C4H10(1) + C3H5(36)</td>
<td>1</td>
<td>H Abstraction</td>
<td>1.68E+12</td>
<td>0</td>
<td>0</td>
<td>12.400</td>
</tr>
<tr>
<td>C3H6(9) + C4H9 (2,38) -&gt; C4H10(1) + C3H5(36)</td>
<td>1</td>
<td>H Abstraction</td>
<td>8.61E+11</td>
<td>0</td>
<td>0</td>
<td>12.300</td>
</tr>
<tr>
<td>C3H6(9) + CH2(19) -&gt; C3H5(36) + CH3(21)</td>
<td>1</td>
<td>H Abstraction</td>
<td>5.00E+05</td>
<td>2.326</td>
<td>0</td>
<td>3.700</td>
</tr>
<tr>
<td>C3H6(9) + CH3(21) -&gt; C3H5(36) + CH4(4)</td>
<td>1</td>
<td>H Abstraction</td>
<td>2.41E+02</td>
<td>2.92</td>
<td>0</td>
<td>7.160</td>
</tr>
<tr>
<td>C3H6(9) + CH3(21) -&gt; C4H9 _2(38)</td>
<td>1</td>
<td>R Addition_MultipleBond</td>
<td>1.28E+05</td>
<td>2.28</td>
<td>0</td>
<td>6.600</td>
</tr>
<tr>
<td>C3H6(9) + CH3O_2(54) -&gt; C3H5(36) + H2(3)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C3H6(9) + + H(17) -&gt; C3H5(36) + H2(3)</td>
<td>1</td>
<td>R Addition_MultipleBond</td>
<td>1.30E+06</td>
<td>2.38</td>
<td>0</td>
<td>2.800</td>
</tr>
<tr>
<td>C3H6(9) + HCCCH(26) -&gt; C3H5(36) + H2(3)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>2.14E+14</td>
<td>0</td>
<td>0</td>
<td>30.700</td>
</tr>
<tr>
<td>C3H6(9) + HCCO(34) -&gt; C3H5(36) + CH2CO(16)</td>
<td>1</td>
<td>H Abstraction</td>
<td>2.32E+13</td>
<td>0</td>
<td>0</td>
<td>7.500</td>
</tr>
<tr>
<td>C3H6(9) + HO2(30) -&gt; C3H5(36) + H2O2(12)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>2.97E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C3H6(9) + O(28) -&gt; C3H5(36) + OH(29)</td>
<td>1</td>
<td>H Abstraction</td>
<td>2.57E+04</td>
<td>3.05</td>
<td>0</td>
<td>3.123</td>
</tr>
<tr>
<td>C3H6(9) + OH(29) -&gt; C3H5(36) + H2O(11)</td>
<td>1</td>
<td>H Abstraction</td>
<td>1.08E+09</td>
<td>1.3333</td>
<td>0</td>
<td>4.999</td>
</tr>
<tr>
<td>C3H6(9) -&gt; C2H4(6) + CH2(S)(20)</td>
<td>-1</td>
<td>Leeds</td>
<td>9.64E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C3H7.(39) + CH3(21) -&gt; C4H10(1)</td>
<td>1</td>
<td>R Recombination</td>
<td>3.37E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H10(1) + C2H2 .(22) -&gt; C2H2(5) + C4H9 _1(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.61E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H10(1) + C2H2(22) -&gt; C2H2(5) + C4H9 _2(38)</td>
<td>1</td>
<td>H Abstraction</td>
<td>2.42E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H10(1) + C3H2(3) -&gt; C2H4(6) + C4H9 _1(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
</tr>
<tr>
<td>C4H10(1) + C3H2(3) -&gt; C2H4(6) + C4H9 _2(38)</td>
<td>1</td>
<td>H Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H10(1) + C2H3O2.(610) -&gt; C2H4O2(121) + C4H9 _1(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H10(1) + C2H3O2.(610) -&gt; C2H4O2(121) + C4H9 _2(38)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H10(1) + C2H5(24) -&gt; C2H6(7) + C4H9 _1(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.95E+03</td>
<td>2.71</td>
<td>0</td>
<td>12.920</td>
</tr>
<tr>
<td>C4H10(1) + C2H5(24) -&gt; C2H6(7) + C4H9 _2(38)</td>
<td>1</td>
<td>H Abstraction</td>
<td>6.16E+03</td>
<td>2.66</td>
<td>0</td>
<td>10.100</td>
</tr>
<tr>
<td>C4H10(1) + C2H5O2(52) -&gt; C2H6O2(168) + C4H9 _1(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H10(1) + C2H5O2(52) -&gt; C2H6O2(168) + C4H9 _2(38)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Type</td>
<td>Energy</td>
<td>Delta E</td>
<td>Delta S</td>
<td>Delta H</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------------</td>
<td>--------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>C4H10(1) + C3H2(25) -&gt; H2CCCH(26) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>6.04E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
</tr>
<tr>
<td>C4H10(1) + C3H2(25) -&gt; H2CCCH(26) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>4.08E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H10(1) + C3H5(36) -&gt; C3H6(9) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.68E+12</td>
<td>0</td>
<td>0</td>
<td>12.400</td>
</tr>
<tr>
<td>C4H10(1) + C3H5(36) -&gt; C3H6(9) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>8.61E+11</td>
<td>0</td>
<td>0</td>
<td>12.300</td>
</tr>
<tr>
<td>C4H10(1) + C4H7(711) -&gt; C4H8(44) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H10(1) + C4H7(711) -&gt; C4H8(44) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H10(1) + C4H7(711) -&gt; C4H8(45) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>3.36E+12</td>
<td>0</td>
<td>0</td>
<td>12.400</td>
</tr>
<tr>
<td>C4H10(1) + C4H7(711) -&gt; C4H8(45) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.72E+12</td>
<td>0</td>
<td>0</td>
<td>12.300</td>
</tr>
<tr>
<td>C4H10(1) + C4H9(11) -&gt; C4H10(1) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>6.16E+03</td>
<td>2.66</td>
<td>0</td>
<td>10.100</td>
</tr>
<tr>
<td>C4H10(1) + C4H9(27) -&gt; C4H10(1) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>6.16E+03</td>
<td>2.66</td>
<td>0</td>
<td>10.100</td>
</tr>
<tr>
<td>C4H10(1) + CH3(19) -&gt; CH3(21) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>1.00E+06</td>
<td>2.326</td>
<td>0</td>
<td>3.700</td>
</tr>
<tr>
<td>C4H10(1) + CH3(19) -&gt; CH3(21) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.02E+00</td>
<td>3.46</td>
<td>0</td>
<td>7.470</td>
</tr>
<tr>
<td>C4H10(1) + CH3(21) -&gt; CH4(4) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>1.67E+06</td>
<td>1.9</td>
<td>0</td>
<td>11.050</td>
</tr>
<tr>
<td>C4H10(1) + CH3(21) -&gt; CH4(4) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>5.80E+06</td>
<td>1.77</td>
<td>0</td>
<td>8.530</td>
</tr>
<tr>
<td>C4H10(1) + CH302(54) -&gt; CH4O2(240) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H10(1) + CH302(54) -&gt; CH4O2(240) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H10(1) + H(17) -&gt; H2(3) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.77E+08</td>
<td>1.75</td>
<td>0</td>
<td>7.510</td>
</tr>
<tr>
<td>C4H10(1) + H(17) -&gt; H2(3) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>5.20E+08</td>
<td>1.69</td>
<td>0</td>
<td>4.780</td>
</tr>
<tr>
<td>C4H10(1) + H2CCCH(26) -&gt; C3H4(8) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>3.13E+13</td>
<td>0</td>
<td>0</td>
<td>19.700</td>
</tr>
<tr>
<td>C4H10(1) + H2CCCH(26) -&gt; C3H4(8) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.30E+13</td>
<td>0</td>
<td>0</td>
<td>20.200</td>
</tr>
<tr>
<td>C4H10(1) + HCCO(34) -&gt; CH2CO(16) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>0</td>
<td>9.175</td>
</tr>
<tr>
<td>C4H10(1) + HCCO(34) -&gt; CH2CO(16) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H10(1) + HCO(31) -&gt; CH2O(15) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>5.50E+03</td>
<td>2.81</td>
<td>0</td>
<td>5.860</td>
</tr>
<tr>
<td>C4H10(1) + HCO(31) -&gt; CH2O(15) + C4H9(37)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.08E+11</td>
<td>0</td>
<td>0</td>
<td>6.960</td>
</tr>
<tr>
<td>C4H10(1) + HO2(30) -&gt; C4H9(37) + H2O(12)</td>
<td>1</td>
<td>Leeds</td>
<td>1.68E+13</td>
<td>0</td>
<td>0</td>
<td>20.425</td>
</tr>
<tr>
<td>C4H10(1) + HO2(30) -&gt; C4H9(37) + H2O(12)</td>
<td>1</td>
<td>Leeds</td>
<td>1.12E+13</td>
<td>0</td>
<td>0</td>
<td>17.678</td>
</tr>
<tr>
<td>C4H10(1) + O(28) -&gt; OH(29) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>5.13E+04</td>
<td>3.05</td>
<td>0</td>
<td>3.123</td>
</tr>
<tr>
<td>C4H10(1) + O(28) -&gt; OH(29) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>1.91E+05</td>
<td>2.71</td>
<td>0</td>
<td>2.106</td>
</tr>
<tr>
<td>C4H10(1) + OH(29) -&gt; H2O(11) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>1.58E+08</td>
<td>1.8</td>
<td>0</td>
<td>0.278</td>
</tr>
<tr>
<td>C4H10(1) + OH(29) -&gt; H2O(11) + C4H9(37)</td>
<td>1</td>
<td>H Abstraction</td>
<td>3.60E+06</td>
<td>2</td>
<td>0</td>
<td>-1.133</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Pathway</td>
<td>Rate Constant</td>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
<td>---------------</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>C4H2(10) + H(17) -&gt; C2H2(5) + C2H(22)</td>
<td>-1</td>
<td>Leeds</td>
<td>9.03E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H2(10) + H(17) -&gt; H2CCCH(27)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>3.72E+13</td>
<td>0</td>
<td>0</td>
<td>2.300</td>
</tr>
<tr>
<td>C4H2(10) + O(28) -&gt; C3H2(25) + CO(13)</td>
<td>1</td>
<td>Leeds</td>
<td>7.89E+12</td>
<td>0</td>
<td>0</td>
<td>1.347</td>
</tr>
<tr>
<td>C4H2(10) + OH(29) -&gt; C3H2(25) + HCO(31)</td>
<td>1</td>
<td>Leeds</td>
<td>6.68E+12</td>
<td>0</td>
<td>0</td>
<td>-0.409</td>
</tr>
<tr>
<td>C4H6(278) + H(17) -&gt; C4H7.(711)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>7.42E+13</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
</tr>
<tr>
<td>C4H7.(711) + H(17) -&gt; C4H8(44)</td>
<td>1</td>
<td>R_Recombination</td>
<td>5.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H7.(711) + H(17) -&gt; C4H8(45)</td>
<td>1</td>
<td>R_Recombination</td>
<td>5.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H7.(711) -&gt; C4H6(278) + H(17)</td>
<td>-1</td>
<td>Beta_Scission</td>
<td>7.42E+13</td>
<td>0</td>
<td>0</td>
<td>1.000</td>
</tr>
<tr>
<td>C4H7O.(599) + OH(29) -&gt; C4H8O2(789)</td>
<td>1</td>
<td>R_Recombination</td>
<td>2.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H7O2.(1436) -&gt; C4H7.(711) + O2(2)</td>
<td>-1</td>
<td>Beta_Scission</td>
<td>7.54E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H8(44) + C2H2(22) -&gt; C2H2(5) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H8(44) + C2H3(23) -&gt; C2H4(6) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.02E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H8(44) + C2H3O2(610) -&gt; C2H4O2(121) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>0</td>
<td>5.105</td>
</tr>
<tr>
<td>C4H8(44) + C2H5(24) -&gt; C2H6(7) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H8(44) + C2H5O2(52) -&gt; C2H6O2(168) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>0</td>
<td>5.105</td>
</tr>
<tr>
<td>C4H8(44) + C3H2(25) -&gt; H2CCCH(26) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H8(44) + C3H5(36) -&gt; C3H6(9) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H8(44) + C4H7.(711) -&gt; C4H8(45) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H8(44) + C4H9.(137) -&gt; C4H10(1) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H8(44) + C4H9.(238) -&gt; C4H10(1) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H8(44) + CH2(19) -&gt; CH3(21) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.51E+00</td>
<td>3.46</td>
<td>0</td>
<td>7.470</td>
</tr>
<tr>
<td>C4H8(44) + CH3(21) -&gt; CH4(4) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.46E+13</td>
<td>0</td>
<td>0</td>
<td>10.400</td>
</tr>
<tr>
<td>C4H8(44) + CH3O2(54) -&gt; CH4O2(240) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>0</td>
<td>5.105</td>
</tr>
<tr>
<td>C4H8(44) + H(17) -&gt; C4H9.(711) + H2(3)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.40E+06</td>
<td>2.36</td>
<td>0</td>
<td>1.110</td>
</tr>
<tr>
<td>C4H8(44) + Hr(17) -&gt; C4H9.(137)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>1.30E+13</td>
<td>0</td>
<td>0</td>
<td>3.260</td>
</tr>
<tr>
<td>C4H8(44) + Hr(17) -&gt; C4H9.(238)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>1.30E+13</td>
<td>0</td>
<td>0</td>
<td>1.560</td>
</tr>
<tr>
<td>C4H8(44) + H2CCCH(26) -&gt; C3H4(8) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.02E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H8(44) + HCCO(34) -&gt; CH2CO(16) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.02E+03</td>
<td>3.1</td>
<td>0</td>
<td>8.820</td>
</tr>
<tr>
<td>C4H8(44) + HCO(31) -&gt; CH2O(15) + C4H7.(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>7.57E+06</td>
<td>1.95</td>
<td>0</td>
<td>16.625</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Description</td>
<td>Rate (s⁻¹)</td>
<td>Energy (kJ/mol)</td>
<td>Predicted (kJ/mol)</td>
<td>Water (kJ/mol)</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>-----------</td>
<td>-----------------------</td>
<td>------------</td>
<td>-----------------</td>
<td>-------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>C4H8(44) + H2O(30) -&gt; C4H7(711) + H2O2(12)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>2.97E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H8(44) + O(28) -&gt; C4H7(711) + OH(29)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>9.54E+04</td>
<td>2.71</td>
<td>0</td>
<td>2.106</td>
</tr>
<tr>
<td>C4H8(44) + OH(29) -&gt; C4H7(711) + H2O(11)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>5.105</td>
<td></td>
</tr>
<tr>
<td>C4H8(44) -&gt; C4H7(711) + H(17)</td>
<td>-1</td>
<td>Bond_Dissociation</td>
<td>5.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H8(45) + C2H(22) -&gt; C2H2(5) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.61E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>C4H8(45) + C2H3(23) -&gt; C2H4(6) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>4.64E+13</td>
<td>0</td>
<td>7.500</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + C2H3O2(610) -&gt; C2H4O2(121) + C4H7(711)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H8(45) + C2H5(24) -&gt; C2H6(7) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.36E+12</td>
<td>0</td>
<td>0</td>
<td>12.400</td>
</tr>
<tr>
<td>C4H8(45) + C2H5O2(52) -&gt; C2H6O2(168) + C4H7(711)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H8(45) + C3H2(25) -&gt; H2CCCH(26) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>9.28E+13</td>
<td>0</td>
<td>7.500</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + C3H5(36) -&gt; C3H6(9) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.47E+13</td>
<td>0</td>
<td>21.400</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + C4H7(711) -&gt; C4H8(44) + C4H7(711)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>3.04E+06</td>
<td>1.8858</td>
<td>0</td>
<td>9.448</td>
</tr>
<tr>
<td>C4H8(45) + C4H9_1(37) -&gt; C4H10(1) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>3.36E+12</td>
<td>0</td>
<td>12.400</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + C4H9_2(38) -&gt; C4H10(1) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.72E+12</td>
<td>0</td>
<td>12.300</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + CH2(19) -&gt; CH3(21) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.00E+06</td>
<td>2.326</td>
<td>0</td>
<td>3.700</td>
</tr>
<tr>
<td>C4H8(45) + CH3(21) -&gt; CH4(4) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>4.82E+02</td>
<td>2.92</td>
<td>0</td>
<td>7.160</td>
</tr>
<tr>
<td>C4H8(45) + CH3O2(54) -&gt; CH4O2(240) + C4H7(711)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H8(45) + H(17) -&gt; C4H7(711) + H2(3)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.60E+06</td>
<td>2.38</td>
<td>0</td>
<td>2.800</td>
</tr>
<tr>
<td>C4H8(45) + H2CCCH(26) -&gt; C3H4(8) + C4H7(711)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>2.00E+13</td>
<td>0</td>
<td>2.900</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + HCCO(34) -&gt; CH2CO(16) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.14E+14</td>
<td>0</td>
<td>30.700</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + HCO(31) -&gt; CH2O(15) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>4.64E+13</td>
<td>0</td>
<td>7.500</td>
<td></td>
</tr>
<tr>
<td>C4H8(45) + HO2(30) -&gt; H2O(12) + C4H7(711)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.97E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>C4H8(45) + O(28) -&gt; C4H7(711) + OH(29)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.13E+04</td>
<td>3.05</td>
<td>0</td>
<td>3.123</td>
</tr>
<tr>
<td>C4H8(45) + OH(29) -&gt; C4H7(711) + H2O(11)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>2.17E+09</td>
<td>1.3333</td>
<td>0</td>
<td>4.999</td>
</tr>
<tr>
<td>C4H8O2(789) + H(17) -&gt; C4H9O2(626)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>1.30E+13</td>
<td>0</td>
<td>1.560</td>
<td></td>
</tr>
<tr>
<td>C4H9_1(37) + H(17) -&gt; C4H8O2(626)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.00E+14</td>
<td>0</td>
<td>0.000</td>
<td></td>
</tr>
<tr>
<td>C4H9_1(37) + H2O(12) -&gt; C4H10(1) + HO2(30)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.68E+13</td>
<td>0</td>
<td>20.425</td>
<td></td>
</tr>
<tr>
<td>C4H9_1(37) -&gt; C4H9_2(38)</td>
<td>1</td>
<td>intra_H_migration</td>
<td>3.88E+10</td>
<td>0.89</td>
<td>35.800</td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>R_Recombination</td>
<td>Rate Constant</td>
<td>Preexponent</td>
<td>Temperature</td>
<td>Energy</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>----------------</td>
<td>---------------</td>
<td>-------------</td>
<td>-------------</td>
<td>--------</td>
</tr>
<tr>
<td>$C_4H_9_2(38) + H(17) \rightarrow C_4H_1O(1)$</td>
<td>1</td>
<td>Leeds</td>
<td>2.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$C_4H_9_2(38) + H_2O_2(12) \rightarrow C_4H_1O(1) + HO_2(30)$</td>
<td>-1</td>
<td>Leeds</td>
<td>1.12E+13</td>
<td>0</td>
<td>0</td>
<td>17.678</td>
</tr>
<tr>
<td>$C_4H_9_2(38) \rightarrow C_4H_9_1(37)$</td>
<td>-1</td>
<td>intra_H_migration</td>
<td>3.88E+10</td>
<td>0.89</td>
<td>0</td>
<td>35.800</td>
</tr>
<tr>
<td>$C_5H_7O_2.(1422) \rightarrow C_4H_7.(711) + CO_2(14)$</td>
<td>-1</td>
<td>Beta_Scission</td>
<td>1.65E+12</td>
<td>0.1474</td>
<td>0</td>
<td>2.533</td>
</tr>
<tr>
<td>$C_5H_9O.(1433) \rightarrow CH_2O(15) + C_4H_7(711)$</td>
<td>-1</td>
<td>Beta_Scission</td>
<td>8.26E+11</td>
<td>0.1474</td>
<td>0</td>
<td>2.533</td>
</tr>
<tr>
<td>$CH(18) + CH_2(19) \rightarrow CH_2(19) + C_2H_2(5)$</td>
<td>1</td>
<td>Leeds</td>
<td>5.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + CH_3(21) \rightarrow C_2H_4(6) + H(17)$</td>
<td>1</td>
<td>Leeds</td>
<td>4.22E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + O(28)$</td>
<td>-1</td>
<td>Leeds</td>
<td>6.57E+15</td>
<td>0</td>
<td>0</td>
<td>57.580</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + H(17)$</td>
<td>-1</td>
<td>Leeds</td>
<td>1.51E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + H(17) + HCCO(34) \rightarrow CH_2CO(16) + O(28)$</td>
<td>-1</td>
<td>Leeds</td>
<td>1.33E+12</td>
<td>0</td>
<td>0</td>
<td>1.350</td>
</tr>
<tr>
<td>$CH(18) + HCCO(34) \rightarrow C_2H_2(5) + CO(13)$</td>
<td>1</td>
<td>Leeds</td>
<td>1.08E+14</td>
<td>0</td>
<td>0</td>
<td>0.796</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + H(17)$</td>
<td>-1</td>
<td>Leeds</td>
<td>1.20E+13</td>
<td>0</td>
<td>0</td>
<td>0.796</td>
</tr>
<tr>
<td>$CH(18) + CH_2(19) \rightarrow CH_2(19) + C_2H_2(5) + H_2(3)$</td>
<td>1</td>
<td>Leeds</td>
<td>4.22E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + CH_2(19) \rightarrow CH_2(19) + C_2H_2(5) + H(17)$</td>
<td>1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + CH_2(19) \rightarrow CH_2(19) + C_2H_2(5) + H(17)$</td>
<td>1</td>
<td>Leeds</td>
<td>1.08E+14</td>
<td>0</td>
<td>0</td>
<td>0.796</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + H(17)$</td>
<td>-1</td>
<td>Leeds</td>
<td>2.17E+06</td>
<td>2.1</td>
<td>0</td>
<td>1.570</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + H(17)$</td>
<td>-1</td>
<td>Leeds</td>
<td>6.57E+15</td>
<td>0</td>
<td>0</td>
<td>57.580</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + O(28)$</td>
<td>-1</td>
<td>Leeds</td>
<td>1.51E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + H(17)$</td>
<td>-1</td>
<td>Leeds</td>
<td>1.33E+12</td>
<td>0</td>
<td>0</td>
<td>1.350</td>
</tr>
<tr>
<td>$CH(18) + H(17) + HCCO(34) \rightarrow CH_2CO(16) + O(28)$</td>
<td>-1</td>
<td>Leeds</td>
<td>2.91E+16</td>
<td>0</td>
<td>0</td>
<td>90.573</td>
</tr>
<tr>
<td>$CH(18) + H(17) \rightarrow H(17) + CH_2(19) + H_2(3)$</td>
<td>1</td>
<td>Leeds</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
<td>2.000</td>
</tr>
<tr>
<td>$CH(18) + HCCO(34) \rightarrow C_2H_2(5) + H_2(3)$</td>
<td>1</td>
<td>Leeds</td>
<td>3.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + H_2O_2(12) \rightarrow C_4H_1O(1) + H(17)$</td>
<td>1</td>
<td>Leeds</td>
<td>9.64E+13</td>
<td>0</td>
<td>0</td>
<td>-0.516</td>
</tr>
<tr>
<td>$CH(18) + HCO(13) \rightarrow CH_3(21) + CO(13)$</td>
<td>1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>$CH(18) + CO(13) \rightarrow CH_2CO(16) + H(17)$</td>
<td>-1</td>
<td>Leeds</td>
<td>7.20E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Energy (kcal/mol)</td>
<td>ΔE (kcal/mol)</td>
<td>ΔG (kcal/mol)</td>
<td>ΔH (kcal/mol)</td>
<td>Leeds</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>--------------</td>
<td>--------------</td>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>CH2(19) + O(28) -&gt; CO(13) + H2(3)</td>
<td>Direction = 1</td>
<td>4.80E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>4.800E+13</td>
</tr>
<tr>
<td>CH2(19) + OH(29) -&gt; CH2O(15) + H(17)</td>
<td>Direction = 1</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.810E+13</td>
</tr>
<tr>
<td>CH2(19) -&gt; CH2(S)(20)</td>
<td>Direction = -1</td>
<td>1.51E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.510E+13</td>
</tr>
<tr>
<td>CH2(S)(20) + H2O(11) -&gt; CH3(21) + OH(29)</td>
<td>Direction = -1</td>
<td>7.23E+13</td>
<td>0</td>
<td>2.781</td>
<td>4.737E+12</td>
<td></td>
</tr>
<tr>
<td>CH2(S)(20) -&gt; CH2(19)</td>
<td>Direction = 1</td>
<td>1.51E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.510E+13</td>
</tr>
<tr>
<td>CH2CO(16) + C2H(22) -&gt; HCCO(34) + C2H2(5)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>4.43E+09</td>
<td>0.6961</td>
<td>9.953</td>
<td>3.895E+08</td>
</tr>
<tr>
<td>CH2CO(16) + C2H3(23) -&gt; HCCO(34) + C2H4(6)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>5.88E+13</td>
<td>0</td>
<td>13.100</td>
<td>2.909E+09</td>
</tr>
<tr>
<td>CH2CO(16) + C2H3O2 (610) -&gt; HCCO(34) + C2H4O2 (121)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>6.940</td>
<td>3.471E+01</td>
</tr>
<tr>
<td>CH2CO(16) + C2H5(24) -&gt; HCCO(34) + C2H6(7)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>9.175</td>
<td>1.487E+07</td>
</tr>
<tr>
<td>CH2CO(16) + C2H5O2 (52) -&gt; HCCO(34) + C2H6O2 (168)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>6.940</td>
<td>3.471E+01</td>
</tr>
<tr>
<td>CH2CO(16) + C3H2(25) -&gt; HCCO(34) + H2CCCH (26)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>5.88E+13</td>
<td>0</td>
<td>13.100</td>
<td>5.819E+09</td>
</tr>
<tr>
<td>CH2CO(16) + C3H5 (36) -&gt; HCCO(34) + C3H6(9)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.32E+13</td>
<td>0</td>
<td>7.500</td>
<td>1.045E+05</td>
</tr>
<tr>
<td>CH2CO(16) + C4H7 (711) -&gt; C4H8 (44) + HCCO (34)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.02E+03</td>
<td>3.1</td>
<td>8.820</td>
<td>2.764E+02</td>
</tr>
<tr>
<td>CH2CO(16) + C4H7 (711) -&gt; C4H8 (45) + HCCO (34)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>4.64E+13</td>
<td>0</td>
<td>7.500</td>
<td>1.045E+05</td>
</tr>
<tr>
<td>CH2CO(16) + C4H9 (137) -&gt; C4H10(1) + HCCO(34)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>3.02E+13</td>
<td>0</td>
<td>9.175</td>
<td>1.518E+07</td>
</tr>
<tr>
<td>CH2CO(16) + C4H9 (238) -&gt; C4H10(1) + HCCO(34)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.04E+03</td>
<td>3.1</td>
<td>8.820</td>
<td>7.296E+04</td>
</tr>
<tr>
<td>CH2CO(16) + CH2 (19) -&gt; HCCO(34) + C3H (21)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>9.54E+08</td>
<td>1.305</td>
<td>5.685</td>
<td>9.260E+07</td>
</tr>
<tr>
<td>CH2CO(16) + CH3 (21) -&gt; HCCO(34) + CH4 (4)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>6.36E+14</td>
<td>0</td>
<td>13.700</td>
<td>6.164E+08</td>
</tr>
<tr>
<td>CH2CO(16) + CH3O2 (54) -&gt; HCCO(34) + C4H4O2 (240)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>6.940</td>
<td>3.471E+01</td>
</tr>
<tr>
<td>CH2CO(16) + H(17) -&gt; CH2HCO (35)</td>
<td>Direction = 1</td>
<td>R Addition _MultipleBond</td>
<td>1.00E+13</td>
<td>0</td>
<td>2.333</td>
<td>1.935E+12</td>
</tr>
<tr>
<td>CH2CO(16) + H(17) -&gt; CH2HCO (35)</td>
<td>Direction = 1</td>
<td>R Addition _MultipleBond</td>
<td>3.40E+12</td>
<td>0.23</td>
<td>3.133</td>
<td>1.701E+12</td>
</tr>
<tr>
<td>CH2CO(16) + H(17) -&gt; CH2HCO (35)</td>
<td>Direction = 1</td>
<td>R Addition _MultipleBond</td>
<td>3.96E+12</td>
<td>0.046</td>
<td>4.898</td>
<td>1.707E+11</td>
</tr>
<tr>
<td>CH2CO(16) + H(17) -&gt; CH3(21) + CO (13)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>3.766</td>
<td>1.682E+12</td>
</tr>
<tr>
<td>CH2CO(16) + H(17) -&gt; HCCO(34) + H2 (3)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>8.98E+12</td>
<td>0</td>
<td>10.300</td>
<td>1.497E+09</td>
</tr>
<tr>
<td>CH2CO(16) + H2 (3) -&gt; C2H4 (6) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>6.80E+05</td>
<td>1.88</td>
<td>0.179</td>
<td>1.823E+14</td>
</tr>
<tr>
<td>CH2CO(16) + H2CCCH (26) -&gt; HCCO(34) + C3H4 (8)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>5.88E+13</td>
<td>0</td>
<td>13.100</td>
<td>5.383E+02</td>
</tr>
<tr>
<td>CH2CO(16) + HCO (31) -&gt; HCCO(34) + CH2O (15)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>5.42E+03</td>
<td>2.81</td>
<td>5.860</td>
<td>2.036E+03</td>
</tr>
<tr>
<td>CH2CO(16) + HCO (31) -&gt; O2 (2) + H2CCCH (26)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.01E+10</td>
<td>0</td>
<td>2.867</td>
<td>1.451E+16</td>
</tr>
<tr>
<td>CH2CO(16) + HO2 (30) -&gt; C2H3O3 (505)</td>
<td>Direction = 1</td>
<td>R Addition _MultipleBond</td>
<td>6.72E+11</td>
<td>0.0737</td>
<td>3.071</td>
<td>1.257E+11</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Type</td>
<td>Rate Constant</td>
<td>Rate Constant</td>
<td>Rate Constant</td>
<td>Rate Constant</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>------------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>CH2CO(16) + O2(30) -&gt; HCCO(34) + H2O2(12)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>2.88E+01</td>
<td>3.10</td>
<td>6.940</td>
<td>1.364E+01</td>
</tr>
<tr>
<td>CH2CO(16) + O(28) -&gt; CH2(19) + CO2(14)</td>
<td>1</td>
<td>Leeds</td>
<td>1.33E+12</td>
<td>0.00</td>
<td>1.350</td>
<td>5.143E+11</td>
</tr>
<tr>
<td>CH2CO(16) + O(28) -&gt; HCCO(34) + OH(29)</td>
<td>1</td>
<td>Leeds</td>
<td>4.58E+11</td>
<td>0.00</td>
<td>1.350</td>
<td>1.771E+11</td>
</tr>
<tr>
<td>CH2CO(16) + O(28) -&gt; HCO(31) + H(17) + CO(13)</td>
<td>1</td>
<td>Leeds</td>
<td>7.56E+06</td>
<td>1.910</td>
<td>3.740</td>
<td>1.538E+11</td>
</tr>
<tr>
<td>CH2CO(16) + OH(29) -&gt; CH2O(33) + CO(13)</td>
<td>1</td>
<td>Leeds</td>
<td>4.68E+12</td>
<td>0.00</td>
<td>0.000</td>
<td>4.680E+12</td>
</tr>
<tr>
<td>CH2CO(16) + OH(29) -&gt; CH3(21) + CO2(14)</td>
<td>1</td>
<td>Leeds</td>
<td>2.52E+12</td>
<td>0.00</td>
<td>0.000</td>
<td>2.520E+12</td>
</tr>
<tr>
<td>CH2CO(16) + OH(29) -&gt; HCO(31) + H2O(11)</td>
<td>1</td>
<td>Leeds</td>
<td>1.03E+13</td>
<td>0.000</td>
<td>5.940</td>
<td>1.576E+11</td>
</tr>
<tr>
<td>CH2CO(16) + OH(29) -&gt; CH2(19) + CO(13)</td>
<td>1</td>
<td>Leeds</td>
<td>6.57E+15</td>
<td>0.000</td>
<td>57.580</td>
<td>1.644E-02</td>
</tr>
<tr>
<td>CH2HCO(35) + OH(29) -&gt; C2H4O2(121)</td>
<td>1</td>
<td>R_Recombination</td>
<td>2.00E+13</td>
<td>0.000</td>
<td>0.000</td>
<td>2.000E+13</td>
</tr>
<tr>
<td>CH2HCO(35) -&gt; CH2CO(16) + H(17)</td>
<td>-1</td>
<td>Beta_Scission</td>
<td>3.40E+12</td>
<td>0.230</td>
<td>3.133</td>
<td>7.710E+01</td>
</tr>
<tr>
<td>CH2O(15) + C2H(22) -&gt; HCO(31) + C2H2(5)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.44E+08</td>
<td>1.3197</td>
<td>5.129</td>
<td>2.281E+10</td>
</tr>
<tr>
<td>CH2O(15) + C2H3(23) -&gt; HCO(31) + C2H4(6)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.42E+03</td>
<td>2.810</td>
<td>5.860</td>
<td>9.189E+09</td>
</tr>
<tr>
<td>CH2O(15) + C2H3O2(610) -&gt; HCO(31) + C2H4O2(121)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>4.12E+04</td>
<td>2.500</td>
<td>10.210</td>
<td>4.262E+08</td>
</tr>
<tr>
<td>CH2O(15) + C2H5(24) -&gt; CH2H(7) + HCO(31)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.50E+03</td>
<td>2.810</td>
<td>5.860</td>
<td>9.324E+09</td>
</tr>
<tr>
<td>CH2O(15) + C2H5O2(52) -&gt; HCO(31) + C2H6O2(168)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>4.12E+04</td>
<td>2.500</td>
<td>10.210</td>
<td>4.262E+08</td>
</tr>
<tr>
<td>CH2O(15) + C3H2(25) -&gt; HCO(31) + C2HCCCH(26)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.08E+04</td>
<td>2.810</td>
<td>5.860</td>
<td>1.838E+10</td>
</tr>
<tr>
<td>CH2O(15) + C3H5(36) -&gt; C3H6(9) + HCO(31)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.50E+03</td>
<td>2.810</td>
<td>5.860</td>
<td>9.324E+09</td>
</tr>
<tr>
<td>CH2O(15) + C4H7(35) -&gt; C4H8(44) + HCO(31)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>7.94E+10</td>
<td>0.000</td>
<td>6.700</td>
<td>7.107E+08</td>
</tr>
<tr>
<td>CH2O(15) + C4H7(711) -&gt; C4H8(44) + HCO(31)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>7.57E+06</td>
<td>1.950</td>
<td>16.625</td>
<td>1.986E+07</td>
</tr>
<tr>
<td>CH2O(15) + C4H7(711) -&gt; C4H8(45) + HCO(31)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.50E+03</td>
<td>2.810</td>
<td>5.860</td>
<td>9.324E+09</td>
</tr>
<tr>
<td>CH2O(15) + C4H7(711) -&gt; C5H9O(1433)</td>
<td>1</td>
<td>R_Addition_MultipleBond</td>
<td>8.26E+11</td>
<td>0.1474</td>
<td>2.533</td>
<td>3.659E+11</td>
</tr>
<tr>
<td>CH2O(15) + C4H9(137) -&gt; C4H10(1) + HCO(31)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>5.50E+03</td>
<td>2.810</td>
<td>5.860</td>
<td>9.324E+09</td>
</tr>
<tr>
<td>CH2O(15) + C4H9(238) -&gt; C4H10(1) + HCO(31)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.08E+11</td>
<td>0.000</td>
<td>6.960</td>
<td>8.051E+08</td>
</tr>
<tr>
<td>CH2O(15) + CH(18) -&gt; CH2(19) + HCO(31)</td>
<td>1</td>
<td>Leeds</td>
<td>9.64E+13</td>
<td>0.000</td>
<td>-0.516</td>
<td>1.386E+14</td>
</tr>
<tr>
<td>CH2O(15) + CH2(19) -&gt; HCO(31) + CH3(21)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>6.04E+09</td>
<td>0.000</td>
<td>0.000</td>
<td>6.040E+09</td>
</tr>
<tr>
<td>CH2O(15) + CH3(21) -&gt; CH2H5(24) + O(28)</td>
<td>-1</td>
<td>Leeds</td>
<td>6.62E+13</td>
<td>0.000</td>
<td>0.000</td>
<td>3.367E-11</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Type</td>
<td>Rate Constant</td>
<td>Energy</td>
<td>Frequency</td>
<td>Change (kJ/mol)</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>-----------------------------</td>
<td>---------------</td>
<td>--------</td>
<td>-----------</td>
<td>----------------</td>
</tr>
<tr>
<td>CH2O(15) + CH3(21) -&gt; C2H5O(592)</td>
<td>Direction = 1</td>
<td>R_Recombination</td>
<td>7.23E+13</td>
<td>0.000</td>
<td>7.230E+13</td>
<td>0.000</td>
</tr>
<tr>
<td>CH3(21) + C2H5(24) -&gt; C2H6(7) + CH2(S)(20)</td>
<td>Direction = -1</td>
<td>R_Recombination</td>
<td>2.40E+14</td>
<td>0.000</td>
<td>2.390E+13</td>
<td>0.000</td>
</tr>
<tr>
<td>CH3(21) + CH3(21) -&gt; C4H8(44)</td>
<td>Direction = 1</td>
<td>R_Recombination</td>
<td>2.49E+15</td>
<td>-0.90</td>
<td>2.500E+12</td>
<td>0.250</td>
</tr>
<tr>
<td>CH3(21) + CH3(21) -&gt; C2H5(24) + H(17)</td>
<td>Direction = 1</td>
<td>R_Recombination</td>
<td>3.01E+13</td>
<td>0.000</td>
<td>3.150E+12</td>
<td>0.000</td>
</tr>
<tr>
<td>CH3(21) + CH3(21) -&gt; C2H6(7)</td>
<td>Direction = 1</td>
<td>R_Recombination</td>
<td>8.26E+17</td>
<td>-1.40</td>
<td>8.123E+13</td>
<td>1.000</td>
</tr>
<tr>
<td>CH3(21) + CH3(21) -&gt; CH4(4) + CH2(19)</td>
<td>Direction = -1</td>
<td>R_Recombination</td>
<td>4.30E+12</td>
<td>0.000</td>
<td>10.033E+12</td>
<td>0.000</td>
</tr>
<tr>
<td>CH3(21) + CH3(21) -&gt; CH4(4) + CH2(S)(20)</td>
<td>Direction = -1</td>
<td>R_Recombination</td>
<td>7.00E+13</td>
<td>0.000</td>
<td>1.026E+09</td>
<td>0.000</td>
</tr>
</tbody>
</table>

### Additional Reactions

- **CH2O(15)** + H(17) → CH2OH(33)
  - Direction = -1
  - H Abstraction
  - Rate Constant: 1.26E+16
  - Energy: 0.000
  - Change (kJ/mol): 30.005
  - Change (eV): 9.151E+15

- **CH2O(15)** + H(17) → CH3(21) + O(28)
  - Direction = -1
  - Rate Constant: 8.43E+13
  - Energy: 0.000
  - Change (kJ/mol): 1.756E+06

- **CH2O(15)** + H(17) → CH3O(32)
  - Direction = -1
  - Rate Constant: 1.55E+14
  - Energy: 0.000
  - Change (kJ/mol): 5.54E+16
  - Change (eV): 3.611E+16

- **CH2O(15)** + H2O(2) → HCO(31) + H2(3)
  - Direction = -1
  - Rate Constant: 4.12E+04
  - Energy: 0.000
  - Change (kJ/mol): 3.262E+08

- **CH2O(15)** + CO(13) → CO2(14) + CH2(19)
  - Direction = -1
  - Rate Constant: 4.29E+08
  - Energy: 0.000
  - Change (kJ/mol): 1.350E+12

- **CH2O(15)** + CO(13) → HCO(31) + HCO(31)
  - Direction = -1
  - Rate Constant: 8.827E+09
  - Energy: 0.000
  - Change (kJ/mol): 3.029E-20

- **CH2O(15)** + CH3O2(54) → HCO(31) + CH4O2(240)
  - Direction = -1
  - Rate Constant: 10.210
  - Energy: 0.000
  - Change (kJ/mol): 4.262E+08

- **CH2O(15)** + H(17) → CH2(19) + OH(29)
  - Direction = -1
  - Rate Constant: 1.81E+13
  - Energy: 0.000
  - Change (kJ/mol): 3.510E-08

- **CH2O(15)** + H2(3) → H(17) + CH3O(32)
  - Direction = -1
  - Rate Constant: 1.26E+16
  - Energy: 0.000
  - Change (kJ/mol): 9.151E+15

- **CH2O(15)** + H2O2 + H2O2(26) → HCO(31) + C3H4(8)
  - Direction = -1
  - Rate Constant: 8.85E+09
  - Energy: 0.8691
  - Change (kJ/mol): 3.253E+08

- **CH2O(15)** + HCCO(34) → HCO(31) + CH2CO(16)
  - Direction = -1
  - Rate Constant: 5.42E+03
  - Energy: 0.000
  - Change (kJ/mol): 9.189E+09

- **CH2O(15)** + H2O(30) → HCO(31) + H2O2(12)
  - Direction = -1
  - Rate Constant: 4.11E+04
  - Energy: 2.810
  - Change (kJ/mol): 5.860

- **CH2O(15)** + O2(2) → CH3O(32)
  - Direction = -1
  - Rate Constant: 2.17E+10
  - Energy: 0.000
  - Change (kJ/mol): 1.749E-05

- **CH2O(15)** + H2O2(30) → O2(2) + CH3O(32)
  - Direction = -1
  - Rate Constant: 2.17E+10
  - Energy: 0.000
  - Change (kJ/mol): 1.749E-05

- **CH2O(15)** + O(28) → HCO(31) + OH(29)
  - Direction = -1
  - Rate Constant: 4.16E+11
  - Energy: 0.000
  - Change (kJ/mol): 2.762

- **CH2O(15)** + O(28) → O2(2) + CH2(19)
  - Direction = -1
  - Rate Constant: 4.20E+12
  - Energy: 0.000
  - Change (kJ/mol): 1.491

- **CH2O(15)** + OH(29) → HCO(31) + H2O(11)
  - Direction = -1
  - Rate Constant: 3.43E+09
  - Energy: 1.180
  - Change (kJ/mol): -0.447

- **CH2O(15)** + OH(29) → O2(2) + CH3(21)
  - Direction = -1
  - Rate Constant: 3.31E+11
  - Energy: 0.000
  - Change (kJ/mol): 8.939

- **CH2O(15)** + CO(13) → CH2CO(16) + OH(29)
  - Direction = -1
  - Rate Constant: 4.68E+12
  - Energy: 0.000
  - Change (kJ/mol): 1.057E+03

- **CH2O(15)** + H(17) → CH2O(15) + H2(3)
  - Direction = 1
  - Rate Constant: 4.09E+12
  - Energy: 8.827
  - Change (kJ/mol): 8.192E+09

- **CH2O(15)** + CH3(21) → CH4(4) + CH2(19)
  - Direction = -1
  - Rate Constant: 3.253E+08
  - Energy: 0.000
  - Change (kJ/mol): 10.210

- **CH2O(15)** + CH3(21) → C2H5(24) + CH2(S)(20)
  - Direction = -1
  - Rate Constant: 4.262E+08
  - Energy: 0.000
  - Change (kJ/mol): 10.210

- **CH2O(15)** + CO(13) → CH4(4) + CH2(S)(20)
  - Direction = -1
  - Rate Constant: 4.262E+08
  - Energy: 0.000
  - Change (kJ/mol): 10.210
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Direction</th>
<th>Leeds</th>
<th>0</th>
<th>0</th>
<th>0.000</th>
<th>6.815E-15</th>
</tr>
</thead>
<tbody>
<tr>
<td>CH₃(21) + CO(13) -&gt; CH₂(19) + HCO(31)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>3.736</td>
</tr>
<tr>
<td>CH₃(21) + CO(13) -&gt; CH₂CO(16) + H(17)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>2.52E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃(21) + H(17) -&gt; CH₂CO(16) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.93E+14</td>
<td>0</td>
<td>0</td>
<td>0.270</td>
</tr>
<tr>
<td>CH₃(21) + H₂O(11) -&gt; CH₄(4) + CO(13)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃(21) + OH(29) -&gt; CH₂CO(16) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.20E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃(21) + H₂O(30) -&gt; CH₄O₂(240)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.21E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃(21) + O(28) -&gt; CH₂O(15) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>8.43E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃(21) + OH(29) -&gt; CH₂(S)(20) + H₂O(11)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>7.23E+13</td>
<td>0</td>
<td>0</td>
<td>2.781</td>
</tr>
<tr>
<td>CH₃(21) + OH(29) -&gt; CH₄(4) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>7.23E+08</td>
<td>1.56</td>
<td>0</td>
<td>8.481</td>
</tr>
<tr>
<td>CH₃(21) + OH(29) -&gt; CH₂H(19) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.02E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃O(32) + O(28) -&gt; O₂(2) + CH₃(21)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>4.40E+13</td>
<td>0</td>
<td>0</td>
<td>31.383</td>
</tr>
<tr>
<td>CH₃O(32) + OH(29) -&gt; CH₃(21) + HO₂(30)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.80E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH₃O(32) -&gt; CH₂O(33)</td>
<td>Direction = 1</td>
<td>intra H migration</td>
<td>1.79E+10</td>
<td>0.6507</td>
<td>0</td>
<td>38.184</td>
</tr>
<tr>
<td>CH₄(4) + C₂H(22) -&gt; CH₃(21) + C₂H₂(5)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>1.81E+12</td>
<td>0</td>
<td>0</td>
<td>0.500</td>
</tr>
<tr>
<td>CH₄(4) + C₂H₃(23) -&gt; CH₃(21) + C₂H₄(6)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>6.36E+14</td>
<td>0</td>
<td>0</td>
<td>13.700</td>
</tr>
<tr>
<td>CH₄(4) + C₂H₃O₂(610) -&gt; CH₃(21) + C₂H₄O₂(121)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH₄(4) + C₂H₅O₂(52) -&gt; CH₃(21) + C₂H₆O₂(168)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH₄(4) + C₃H₂(25) -&gt; CH₃(21) + H₂CCCH(26)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>1.27E+15</td>
<td>0</td>
<td>0</td>
<td>13.700</td>
</tr>
<tr>
<td>CH₄(4) + C₃H₅(36) -&gt; CH₃(21) + C₃H₆(9)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.41E+02</td>
<td>2.92</td>
<td>0</td>
<td>7.160</td>
</tr>
<tr>
<td>CH₄(4) + C₄H₇(711) -&gt; C₄H₈(44) + CH₃(21)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>5.46E+13</td>
<td>0</td>
<td>0</td>
<td>10.400</td>
</tr>
<tr>
<td>CH₄(4) + C₄H₇(711) -&gt; C₄H₈(45) + CH₃(21)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>4.82E+02</td>
<td>2.92</td>
<td>0</td>
<td>7.160</td>
</tr>
<tr>
<td>CH₄(4) + C₄H₉(1_37) -&gt; C₄H₁₀(1) + CH₃(21)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.67E+06</td>
<td>1.9</td>
<td>0</td>
<td>11.050</td>
</tr>
<tr>
<td>CH₄(4) + C₄H₉(2_38) -&gt; C₄H₁₀(1) + CH₃(21)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>5.80E+06</td>
<td>1.77</td>
<td>0</td>
<td>8.530</td>
</tr>
<tr>
<td>CH₄(4) + CH(18) -&gt; C₂H₄(6) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>3.01E+13</td>
<td>0</td>
<td>0</td>
<td>-0.397</td>
</tr>
<tr>
<td>CH₄(4) + CH₂(19) -&gt; CH₃(21) + CH₃(21)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>4.30E+12</td>
<td>0</td>
<td>0</td>
<td>10.033</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Product</td>
<td>E(0/0)</td>
<td>A</td>
<td>B</td>
<td>E(0/0)</td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
<td>--------</td>
<td>---</td>
<td>---</td>
<td>--------</td>
</tr>
<tr>
<td>CH4(4) + CH3O2(54) -&gt; CH3(21) + CH4O2(240)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4(4) + CO(13) -&gt; CH3(21) + HCO(31)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.20E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH4(4) + H(17) -&gt; CH3(21) + H2(3)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.22E+08</td>
<td>1.87</td>
<td>0</td>
<td>10.590</td>
</tr>
<tr>
<td>CH4(4) + H2CCCH(26) -&gt; CH3(21) + C3H4(8)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.30E+04</td>
<td>2.58</td>
<td>0</td>
<td>14.040</td>
</tr>
<tr>
<td>CH4(4) + HCCO(34) -&gt; CH3(21) + H2CO(16)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>6.36E+14</td>
<td>0</td>
<td>0</td>
<td>13.700</td>
</tr>
<tr>
<td>CH4(4) + HCO(31) -&gt; CH2O(15) + CH3(21)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>4.09E+12</td>
<td>0</td>
<td>0</td>
<td>8.827</td>
</tr>
<tr>
<td>CH4(4) + HO2(30) -&gt; CH3(21) + H2O2(12)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>2.97E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4(4) + O(28) -&gt; CH3(21) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>7.23E+08</td>
<td>1.56</td>
<td>0</td>
<td>8.481</td>
</tr>
<tr>
<td>CH4(4) + OH(29) -&gt; CH3(21) + H2O(11)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.57E+07</td>
<td>1.83</td>
<td>0</td>
<td>2.781</td>
</tr>
<tr>
<td>CH4O2(240) + C2H(22) -&gt; CH3O2(54) + C2H2(5)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.21E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>CH4O2(240) + C2H3(23) -&gt; CH3O2(54) + C2H4(6)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>CH4O2(240) + C2H3O2(610) -&gt; CH3O2(54) + C2H4O2(121)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>CH4O2(240) + C2H5(24) -&gt; CH3O2(54) + C2H6(7)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4O2(240) + C2H5O2(52) -&gt; C2H6O2(168) + CH3O2(54)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>CH4O2(240) + C3H2(25) -&gt; CH3O2(54) + H2CCCH(26)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>2.88E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>CH4O2(240) + C3H5(36) -&gt; C3H6(9) + CH3O2(54)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4O2(240) + C4H7(711) -&gt; C4H8(44) + CH3O2(54)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>0</td>
<td>5.105</td>
</tr>
<tr>
<td>CH4O2(240) + C4H7 (711) -&gt; C4H8(45) + CH3O2(54)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4O2(240) + C4H9(137) -&gt; C4H10(1) + CH3O2(54)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4O2(240) + C4H9(238) -&gt; C4H10(1) + CH3O2(54)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4O2(240) + CH2(19) -&gt; CH3O2(54) + CH3(21)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>CH4O2(240) + CH3(21) -&gt; CH3O2(54) + CH4(4)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.49E+03</td>
<td>2.386</td>
<td>0</td>
<td>9.690</td>
</tr>
<tr>
<td>CH4O2(240) + H(17) -&gt; CH3O2(54) + H2(3)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.06E+10</td>
<td>1.105</td>
<td>0</td>
<td>8.910</td>
</tr>
<tr>
<td>CH4O2(240) + H2CCCH(26) -&gt; CH3O2(54) + C3H4(8)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>2.05E+13</td>
<td>0</td>
<td>0</td>
<td>5.940</td>
</tr>
<tr>
<td>CH4O2(240) + HCCO(34) -&gt; CH3O2(54) + CH2CO(16)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>1.44E+01</td>
<td>3.1</td>
<td>0</td>
<td>6.940</td>
</tr>
<tr>
<td>CH4O2(240) + HCO(31) -&gt; CH3O2(54) + CH2O(15)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>4.12E+04</td>
<td>2.5</td>
<td>0</td>
<td>10.210</td>
</tr>
<tr>
<td>CH4O2(240) + HO2(30) -&gt; CH3O2(54) + H2O2(12)</td>
<td>Direction = 1</td>
<td>H_Abstraction</td>
<td>4.82E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Product</td>
<td>Energy (E)</td>
<td>Intensity (cm²)</td>
<td>Temperature (K)</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-----------</td>
<td>---------</td>
<td>------------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>CH₄O₂(240) + O(28) -&gt; CH₃O₂(54) + OH(29)</td>
<td>Direction = 1 H Abstraction</td>
<td>1.00E+13</td>
<td>0</td>
<td>4.690</td>
<td>3.684E+11</td>
<td></td>
</tr>
<tr>
<td>CH₄O₂(240) + OH(29) -&gt; CH₃O₂(54) + H₂O(11)</td>
<td>Direction = 1 H Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>9.940</td>
<td>2.205E+09</td>
<td></td>
</tr>
<tr>
<td>CO(13) + CH₃(21) -&gt; HCO(34)</td>
<td>Direction = 1 Leeds</td>
<td>2.77E+11</td>
<td>0</td>
<td>-1.708</td>
<td>9.218E+11</td>
<td></td>
</tr>
<tr>
<td>CO(13) + CH₃(21) -&gt; H₂H₂(23) + O(28)</td>
<td>Direction = 1 Leeds</td>
<td>3.00E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.914E-24</td>
<td></td>
</tr>
<tr>
<td>CO(13) + CH₃(21) -&gt; C₂H₃O(61)</td>
<td>Direction = 1 R Addition COM</td>
<td>5.06E+11</td>
<td>0</td>
<td>6.880</td>
<td>3.990E+09</td>
<td></td>
</tr>
<tr>
<td>CO(13) + CO(13) + OH(29) -&gt; O₂(2) + HCO(34)</td>
<td>Direction = 1 Leeds</td>
<td>1.63E+12</td>
<td>0</td>
<td>0.855</td>
<td>1.282E-18</td>
<td></td>
</tr>
<tr>
<td>CO(13) + H(17) + H(17) -&gt; O₂(2) + CH₂(19)</td>
<td>Direction = 1 Leeds</td>
<td>7.20E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.007E-08</td>
<td></td>
</tr>
<tr>
<td>CO(13) + CO(13) + O(28) -&gt; CO₂(14) + OH(29)</td>
<td>Direction = 1 Leeds</td>
<td>1.54E+15</td>
<td>0</td>
<td>3.000</td>
<td>2.153E-22</td>
<td></td>
</tr>
<tr>
<td>CO(13) + O(28) + CH₃(21) -&gt; HCO(34) + CH₂(19)</td>
<td>Direction = 1 Leeds</td>
<td>1.51E+14</td>
<td>0</td>
<td>23.655</td>
<td>8.869E+06</td>
<td></td>
</tr>
<tr>
<td>CO(13) + OH(29) + H(17) -&gt; O₂(2) + CH₂(19)</td>
<td>Direction = 1 Leeds</td>
<td>8.15E+12</td>
<td>0</td>
<td>1.491</td>
<td>2.692E-42</td>
<td></td>
</tr>
<tr>
<td>CO(13) + OH(29) + H(17) -&gt; O₂(2) + CH₂(19) + O(28)</td>
<td>Direction = 1 Leeds</td>
<td>3.13E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.863E+14</td>
<td></td>
</tr>
<tr>
<td>CO(13) + CH₃(21) -&gt; HCO(34) + H(17)</td>
<td>Direction = 1 Leeds</td>
<td>1.66E+13</td>
<td>0</td>
<td>-0.764</td>
<td>1.460E+11</td>
<td></td>
</tr>
<tr>
<td>CO(13) + OH(29) -&gt; O₂(2) + HCO(31)</td>
<td>Direction = 1 Leeds</td>
<td>9.03E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.217E-13</td>
<td></td>
</tr>
<tr>
<td>CO(13) + OH(29) -&gt; O₂(2) + CH₂(19)</td>
<td>Direction = 1 Leeds</td>
<td>1.66E+13</td>
<td>0</td>
<td>0.000</td>
<td>8.761E+36</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + C₂H₇(711) -&gt; C₅H₇O₂(1422)</td>
<td>Direction = 1 R Addition MultipleBond</td>
<td>1.65E+12</td>
<td>0.1474</td>
<td>2.533</td>
<td>7.319E+11</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + CH₃(21) + H(17) -&gt; HCO(31) + CO(13)</td>
<td>Direction = 1 Leeds</td>
<td>3.43E+12</td>
<td>0</td>
<td>0.686</td>
<td>2.117E+12</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + CH₃(21) -&gt; H₂H₂(23) + C₂H₂(22)</td>
<td>Direction = 1 Leeds</td>
<td>9.05E+12</td>
<td>0</td>
<td>0.000</td>
<td>6.408E+13</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + CH₂(19) -&gt; H₂O(15) + CO(13)</td>
<td>Direction = 1 Leeds</td>
<td>2.35E+10</td>
<td>0</td>
<td>0.000</td>
<td>2.350E+10</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + H(17) + H(17) -&gt; O₂(2) + CH₂(19)</td>
<td>Direction = 1 Leeds</td>
<td>5.43E+12</td>
<td>0</td>
<td>1.491</td>
<td>2.280E-11</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + H(17) + CO(13) + OH(29)</td>
<td>Direction = 1 Leeds</td>
<td>1.66E+07</td>
<td>1.3</td>
<td>-0.764</td>
<td>1.904E+06</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + H(17) + O(28) + HCO(31)</td>
<td>Direction = 1 Leeds</td>
<td>3.01E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.588E+18</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + H(17) + O₂(2) + CH(18)</td>
<td>Direction = 1 Leeds</td>
<td>1.66E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.143E-40</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + H₂(3) + O₂(2) + CH₂(19)</td>
<td>Direction = 1 Leeds</td>
<td>5.43E+12</td>
<td>0</td>
<td>1.491</td>
<td>1.206E+42</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + O(28) + O₂(2) + CO(13)</td>
<td>Direction = 1 Leeds</td>
<td>1.26E+13</td>
<td>0</td>
<td>47.038</td>
<td>4.513E-03</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) + OH(29) + CO(13) + H₂O(30)</td>
<td>Direction = 1 Leeds</td>
<td>1.51E+14</td>
<td>0</td>
<td>23.655</td>
<td>2.792E+10</td>
<td></td>
</tr>
<tr>
<td>CO₂(14) -&gt; CO(13) + O(28)</td>
<td>Direction = 1 Leeds</td>
<td>1.54E+15</td>
<td>0</td>
<td>3.000</td>
<td>2.153E-22</td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Type</td>
<td>Rate Constant</td>
<td>Efficiency</td>
<td>Energy</td>
<td>Product</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-----------</td>
<td>--------------------</td>
<td>---------------</td>
<td>------------</td>
<td>----------</td>
<td>--------------</td>
</tr>
<tr>
<td>H(17) + C2H(22) → C2H2(5)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.81E+14</td>
<td>0</td>
<td>0.000</td>
<td>1.810E+14</td>
</tr>
<tr>
<td>H(17) + C2H3O2 (610) → C2H4O2 (121)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0.292</td>
<td>4.557E+12</td>
</tr>
<tr>
<td>H(17) + C2H5O2 (52) → C2H6O2 (168)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0.292</td>
<td>4.557E+12</td>
</tr>
<tr>
<td>H(17) + C3H2 (25) → H2CCCH (26)</td>
<td>1</td>
<td>R_Recombination</td>
<td>2.42E+14</td>
<td>0</td>
<td>0.000</td>
<td>2.420E+14</td>
</tr>
<tr>
<td>H(17) + C3H5 (36) → C3H6 (9)</td>
<td>1</td>
<td>R_Recombination</td>
<td>5.00E+13</td>
<td>0</td>
<td>0.000</td>
<td>5.000E+13</td>
</tr>
<tr>
<td>H(17) + CH2 (19) → CH2HCO (35)</td>
<td>-1</td>
<td>Leeds</td>
<td>4.74E+06</td>
<td>1.88</td>
<td>0.179</td>
<td>1.089E+08</td>
</tr>
<tr>
<td>H(17) + CH2OH (33) → CH3 (21) + OH (29)</td>
<td>1</td>
<td>Leeds</td>
<td>1.02E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.020E+13</td>
</tr>
<tr>
<td>H(17) + CH3O (32) → CH2O (15) + H2 (3)</td>
<td>1</td>
<td>Leeds</td>
<td>1.81E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.810E+13</td>
</tr>
<tr>
<td>H(17) + CH3O2 (54) → CH4O2 (240)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0.292</td>
<td>4.557E+12</td>
</tr>
<tr>
<td>H(17) + CO (13) → HCO (31)</td>
<td>-1</td>
<td>Leeds</td>
<td>9.64E+13</td>
<td>0</td>
<td>0.000</td>
<td>2.107E+20</td>
</tr>
<tr>
<td>H(17) + H (17) → H2 (3)</td>
<td>1</td>
<td>Leeds</td>
<td>1.87E+18</td>
<td>-1</td>
<td>0.000</td>
<td>2.615E+15</td>
</tr>
<tr>
<td>H(17) + H2CCCH (26) → C3H4 (8)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.21E+14</td>
<td>0</td>
<td>0.000</td>
<td>1.210E+14</td>
</tr>
<tr>
<td>H(17) + HCCO (34) → CH2 (19) + CO (13)</td>
<td>1</td>
<td>Leeds</td>
<td>1.51E+14</td>
<td>0</td>
<td>0.000</td>
<td>1.510E+14</td>
</tr>
<tr>
<td>H(17) + HCCO (34) → CH2CO (16)</td>
<td>1</td>
<td>Leeds</td>
<td>1.21E+14</td>
<td>0</td>
<td>0.000</td>
<td>1.210E+14</td>
</tr>
<tr>
<td>H(17) + CO (31) → CO (13) + H2 (3)</td>
<td>1</td>
<td>Leeds</td>
<td>9.03E+13</td>
<td>0</td>
<td>0.000</td>
<td>9.030E+13</td>
</tr>
<tr>
<td>H(17) + HO2 (30) → H2 (3) + O2 (2)</td>
<td>1</td>
<td>Leeds</td>
<td>4.28E+13</td>
<td>0</td>
<td>1.409</td>
<td>1.587E+13</td>
</tr>
<tr>
<td>H(17) + HO2 (30) → H2O (11) + O (28)</td>
<td>1</td>
<td>Leeds</td>
<td>3.01E+13</td>
<td>0</td>
<td>1.720</td>
<td>8.970E+12</td>
</tr>
<tr>
<td>H(17) + HO2 (30) → H2O2 (12)</td>
<td>1</td>
<td>R_Recombination</td>
<td>1.50E+13</td>
<td>-0.15</td>
<td>0.292</td>
<td>4.557E+12</td>
</tr>
<tr>
<td>H(17) + HO2 (30) → OH (29) + OH (29)</td>
<td>1</td>
<td>Leeds</td>
<td>1.69E+14</td>
<td>0</td>
<td>0.874</td>
<td>9.133E+13</td>
</tr>
<tr>
<td>H(17) + O (28) → OH (29)</td>
<td>1</td>
<td>Leeds</td>
<td>1.18E+19</td>
<td>-1</td>
<td>0.000</td>
<td>1.650E+16</td>
</tr>
<tr>
<td>H(17) + OH (29) → H2O (11)</td>
<td>1</td>
<td>Leeds</td>
<td>5.53E+22</td>
<td>-2</td>
<td>0.000</td>
<td>1.082E+17</td>
</tr>
<tr>
<td>H2 (3) + C2H (22) → H (17) + C2H2 (5)</td>
<td>1</td>
<td>H_Abstraction</td>
<td>1.08E+13</td>
<td>0</td>
<td>0.000</td>
<td>2.345E+12</td>
</tr>
<tr>
<td>H2 (3) + C2H3O2 (610) → H (17) + C2H4O2 (121)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.06E+10</td>
<td>1.105</td>
<td>0.000</td>
<td>5.517E+04</td>
</tr>
<tr>
<td>H2 (3) + C2H5O2 (52) → H (17) + C2H6O2 (168)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.06E+10</td>
<td>1.105</td>
<td>0.000</td>
<td>5.517E+04</td>
</tr>
<tr>
<td>H2 (3) + C3H2 (25) → H (17) + H2CCCH (26)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.80E+13</td>
<td>0</td>
<td>0.000</td>
<td>1.276E+10</td>
</tr>
<tr>
<td>H2 (3) + C3H5 (36) → H (17) + C3H6 (9)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.30E+06</td>
<td>2.38</td>
<td>0.000</td>
<td>4.237E+06</td>
</tr>
<tr>
<td>H2 (3) + C4H7 (711) → H (17) + C4H8 (44)</td>
<td>-1</td>
<td>H_Abstraction</td>
<td>1.40E+06</td>
<td>2.36</td>
<td>0.000</td>
<td>2.835E+06</td>
</tr>
</tbody>
</table>

245
<table>
<thead>
<tr>
<th>Reaction Details</th>
<th>Direction</th>
<th>H Abstraction</th>
<th>E (kJ/mol)</th>
<th>ZPE (kJ/mol)</th>
<th>G (kJ/mol)</th>
<th>S (kJ/mol)</th>
<th>barrier (kJ/mol)</th>
<th>E (kJ/mol)</th>
<th>ZPE (kJ/mol)</th>
<th>G (kJ/mol)</th>
<th>S (kJ/mol)</th>
<th>barrier (kJ/mol)</th>
</tr>
</thead>
<tbody>
<tr>
<td>H₂(3) + C₄H₇_.(711) → H(17) + C₄H₈(45)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.60E+06</td>
<td>2.38</td>
<td>0</td>
<td>2.800</td>
<td>4.237E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + C₄H₉_1(37) → C₄H₁₀(1) + H(17)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>3.77E+08</td>
<td>1.75</td>
<td>0</td>
<td>7.510</td>
<td>2.575E+08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + C₄H₉_2(38) → C₄H₁₀(1) + H(17)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>5.20E+08</td>
<td>1.69</td>
<td>0</td>
<td>4.780</td>
<td>1.286E+08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + CH₂(19) → H(17) + CH₃(21)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>2.35E+10</td>
<td>0.8291</td>
<td>0</td>
<td>28.328</td>
<td>1.199E+04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + CH₂₋₁(20) → CH₃(21) + H₁(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>7.23E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>7.230E+13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + CH₃(21) → H(17) + CH₄(4)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.22E+08</td>
<td>1.87</td>
<td>0</td>
<td>10.590</td>
<td>9.814E+08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + CH₃O₂(54) → H(17) + CH₄O₂(240)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.06E+10</td>
<td>1.105</td>
<td>0</td>
<td>8.910</td>
<td>5.517E+04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + CO(13) → CH₂O(15)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>3.26E+36</td>
<td>-5.54</td>
<td>0</td>
<td>96.651</td>
<td>6.159E-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + C₂H₂CC(26) → H(17) + C₃H₄(8)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.01E+08</td>
<td>1.98</td>
<td>0</td>
<td>11.780</td>
<td>4.480E+03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + HCCO(34) → H(17) + CH₂CO(16)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>8.98E+12</td>
<td>0</td>
<td>0</td>
<td>10.300</td>
<td>6.378E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + O(28) → OH(29) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>5.12E+04</td>
<td>2.67</td>
<td>0</td>
<td>6.276</td>
<td>2.581E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + O₂(2) → H(17) + HO₂(30)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>4.28E+13</td>
<td>0</td>
<td>0</td>
<td>1.409</td>
<td>3.484E-03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) + OH(29) → H₂O(11) + H(17)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>4.52E+08</td>
<td>1.6</td>
<td>0</td>
<td>18.414</td>
<td>3.669E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂(3) → H(17) + H₁(17)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.87E+18</td>
<td>-1</td>
<td>0</td>
<td>0.000</td>
<td>1.384E-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂CCCH(27) + H(17) → C₂H₂(5) + C₂H₂(5)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>2.00E+09</td>
<td>0</td>
<td>0</td>
<td>57.812</td>
<td>3.686E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂CCCH(27) → C₄H₂(10) + H(17)</td>
<td>Direction = -1</td>
<td>Beta Scission</td>
<td>3.72E+13</td>
<td>0</td>
<td>0</td>
<td>2.300</td>
<td>2.727E+00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂CCCH(26) + H(17) → C₂H₂(5) + CH₂(S)(20)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.75E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.415E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂CCCH(26) + O(28) → C₂H₂(5) + CO(13) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.39E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.390E+14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + CH₂O₂(610) → CH₂O₂(121) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
<td>4.536E-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + CH₂O₂(52) → CH₂H₂O₂(168) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
<td>4.536E-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + CH₂H₂(25) → H₂CCCH(26) + OH(29)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>9.68E+02</td>
<td>2.9</td>
<td>0</td>
<td>14.860</td>
<td>5.258E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + C₂H₅(36) → C₂H₆(9) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.08E+09</td>
<td>1.3333</td>
<td>0</td>
<td>4.999</td>
<td>8.259E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + C₄H₇_.(711) → C₄H₈(44) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.68E+09</td>
<td>1</td>
<td>0</td>
<td>5.105</td>
<td>2.871E+00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + C₄H₇_.(711) → C₄H₈(45) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.17E+09</td>
<td>1.3333</td>
<td>0</td>
<td>4.999</td>
<td>8.259E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + C₄H₉_1(37) → C₄H₁₀(1) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>1.58E+08</td>
<td>1.8</td>
<td>0</td>
<td>0.278</td>
<td>2.599E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + C₄H₉_2(38) → C₄H₁₀(1) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>3.60E+06</td>
<td>2</td>
<td>0</td>
<td>-1.133</td>
<td>4.684E+04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + CH₂₋₁(20) → CH₃(21) + OH(29)</td>
<td>Direction = 1</td>
<td>H Abstraction</td>
<td>1.45E+08</td>
<td>1.4333</td>
<td>0</td>
<td>9.820</td>
<td>1.777E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + CH₃O₂(54) → CH₄O₂(240) + OH(29)</td>
<td>Direction = -1</td>
<td>H Abstraction</td>
<td>2.41E+12</td>
<td>0</td>
<td>0</td>
<td>9.940</td>
<td>4.536E-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) + CO(13) → OH(29) + HCO(31)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.02E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>2.631E-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(11) → H(17) + H₂(3) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>4.52E+08</td>
<td>1.6</td>
<td>0</td>
<td>18.414</td>
<td>3.919E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Type</td>
<td>Energy</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + H₂CCCH(26) → C₃H₄(8) + OH(29)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>2.05E+13</td>
<td>5.940</td>
<td>1.320E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + HCCO(34) → CH₂CO(16) + OH(29)</td>
<td>-1</td>
<td>H Abstraction</td>
<td>1.03E+13</td>
<td>5.940</td>
<td>7.135E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + H₀₂(30) → H₂O₂(12) + OH(29)</td>
<td>Leeds</td>
<td></td>
<td>7.83E+12</td>
<td>1.331</td>
<td>1.240E+02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + O(28) → H(17) + H₂O₂(30)</td>
<td>Leeds</td>
<td></td>
<td>3.01E+13</td>
<td>1.720</td>
<td>1.381E-03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + O(2) → OH(29) + H₂O₂(30)</td>
<td>Leeds</td>
<td></td>
<td>2.89E+13</td>
<td>-0.497</td>
<td>9.607E-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + C₂H(22) → C₂H₂(5) + H₀₂(30)</td>
<td>Leeds</td>
<td></td>
<td>5.53E+13</td>
<td>0.000</td>
<td>2.420E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H₂O(1) + O₂(2) → C₂H₂(5) + O(28)</td>
<td>Leeds</td>
<td></td>
<td>5.06E+06</td>
<td>1.570</td>
<td>1.163E+08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCCO(34) + H(17) → C₂H₂(5) + O(28)</td>
<td>Leeds</td>
<td></td>
<td>1.00E+13</td>
<td>0.000</td>
<td>1.000E+13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCCO(34) + OH(29) → H₂O(1) + H₂O₂(30)</td>
<td>Leeds</td>
<td></td>
<td>7.83E+12</td>
<td>1.331</td>
<td>3.069E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCCO(34) + C₂H₂(5) + CO(13) + CO(13)</td>
<td>Leeds</td>
<td></td>
<td>2.00E+13</td>
<td>0.000</td>
<td>6.855E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCCO(34) + O(28) → O₂(2) + C₂H(22)</td>
<td>Leeds</td>
<td></td>
<td>5.06E+06</td>
<td>2.10</td>
<td>1.163E+08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCCO(34) → CO(13) + CH(18)</td>
<td>Leeds</td>
<td></td>
<td>2.77E+11</td>
<td>-1.708</td>
<td>3.388E-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

247
<table>
<thead>
<tr>
<th>Reaction</th>
<th>Direction</th>
<th>Leeds</th>
<th>0</th>
<th>0</th>
<th>0.686</th>
<th>3.224E-09</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHCO(18) + CO(13) -&gt; CO2(14) + CH(18)</td>
<td>Direction = -1 Leeds 3.43E+12 0 0 1.350 3.757E+04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + CO(13) -&gt; CH2CO(16) + O(28)</td>
<td>Direction = -1 Leeds 2.52E+11 0 0 0.000 5.959E+13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + H(17) -&gt; CH(18) + OH(29)</td>
<td>Direction = -1 Leeds 3.00E+13 0 0 0.000 5.959E+13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + H(17) -&gt; CH2O(15)</td>
<td>Direction = -1 Leeds 1.40E+36 -5.54 0 96.651 1.095E+16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + H2(3) -&gt; CH2O(15) + H(17)</td>
<td>Direction = -1 Leeds 1.26E+08 1.62 0 2.164 1.090E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + H2O(11) -&gt; CH2O(15) + OH(29)</td>
<td>Direction = -1 Leeds 3.43E+09 1.18 0 -0.447 1.105E+03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + H2O2(12) -&gt; CH2O(15) + HO2(30)</td>
<td>Direction = -1 Leeds 4.11E+04 2.5 0 10.196 1.071E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + HCCO(34) -&gt; O2(2) + C3H2(25)</td>
<td>Direction = -1 Leeds 1.00E+13 0 0 0.000 5.233E-07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + HCO(31) -&gt; CH2CO(16) + O(28)</td>
<td>Direction = -1 Leeds 2.52E+11 0 0 1.350 8.231E-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + HCO(31) -&gt; CH2O(15) + CO(13)</td>
<td>Direction = 1 Leeds 3.01E+13 0 0 0.000 3.010E+13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + HCO(31) -&gt; OH(29) + HCCO(34)</td>
<td>Direction = -1 Leeds 1.00E+13 0 0 0.000 1.184E+01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + HO2(30) -&gt; O2(2) + CH2O(15)</td>
<td>Direction = -1 Leeds 6.02E+13 0 0 40.638 9.789E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) + OH(29) -&gt; CH2O(15) + O(28)</td>
<td>Direction = -1 Leeds 4.16E+11 0.57 0 2.762 3.987E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HCO(31) -&gt; H(17) + CO(13)</td>
<td>Direction = -1 Leeds 4.49E+14 0 0 15.750 6.880E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C2H3(23) -&gt; CH4O2(121)</td>
<td>Direction = 1 HO2 Addition 1.50E+13 -0.15 0 0.292 4.557E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C2H4(6) -&gt; O2(2) + C2H5(24)</td>
<td>Direction = -1 HO2 Addition 7.56E+11 0 0 0.000 2.940E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C3H4(8) -&gt; O2(2) + C3H5(36)</td>
<td>Direction = -1 HO2 Addition 2.42E+12 0 0 13.550 5.294E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C3H6(9) -&gt; O2(2) + C3H7(39)</td>
<td>Direction = -1 HO2 Addition 3.62E+11 0 0 0.000 7.029E+05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C4H6(278) -&gt; O2(2) + C4H7(711)</td>
<td>Direction = -1 HO2 Addition 7.56E+11 0 0 0.000 3.161E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C4H8(44) -&gt; C4H9_1(37) + O2(2)</td>
<td>Direction = -1 HO2 Addition 3.62E+11 0 0 0.000 6.650E+05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C4H8(44) -&gt; C4H9_2(38) + O2(2)</td>
<td>Direction = -1 HO2 Addition 7.56E+11 0 0 0.000 1.769E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C4H8(45) -&gt; C4H9_2(38) + O2(2)</td>
<td>Direction = -1 HO2 Addition 3.62E+11 0 0 0.000 3.651E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C4H8O2(789) -&gt; C4H9O2(626) + O2(2)</td>
<td>Direction = -1 HO2 Addition 7.56E+11 0 0 0.000 1.769E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + C2HCO(16) -&gt; O2(2) + C2H3O(61)</td>
<td>Direction = -1 HO2 Addition 7.56E+11 0 0 0.000 5.018E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + CH2CO(16) -&gt; O2(2) + CH2HCO(35)</td>
<td>Direction = -1 HO2 Addition 6.56E+11 0 0 2.710 5.180E+05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + CH2CO(16) -&gt; O2(2) + CH2HCO(35)</td>
<td>Direction = -1 HO2 Addition 2.42E+12 0 0 13.550 9.285E+02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + CH2O(15) -&gt; O2(2) + CH2OH(33)</td>
<td>Direction = -1 HO2 Addition 2.28E+13 0 0 0.000 5.936E+06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + CO(13) -&gt; O2(2) + HCO(31)</td>
<td>Direction = -1 Leeds 1.52E+13 0 0 -1.694 5.512E+02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HO2(30) + H2(3) -&gt; H2O2(12) + H(17)</td>
<td>Direction = -1 Leeds 1.69E+12 0 0 3.753 4.556E+04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Energy (E)</td>
<td>Rate Constant (k)</td>
<td>Temperature (T)</td>
<td>Energy (D)</td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------</td>
<td>------------</td>
<td>------------------</td>
<td>----------------</td>
<td>------------</td>
<td></td>
</tr>
<tr>
<td>H₂O₂(30) + H₂O₂(30) -&gt; H₂O₂(12) + O₂(2)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.87E+12</td>
<td>0</td>
<td>0</td>
<td>1.541</td>
</tr>
<tr>
<td>H₂O₂(30) -&gt; O₂(2) + H(17)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>2.10E+18</td>
<td>-0.8</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O(28) + H₂O(11) -&gt; O₂(29) + OH(29)</td>
<td>Direction = -1</td>
<td>Leeds</td>
<td>1.51E+09</td>
<td>1.14</td>
<td>0</td>
<td>0.100</td>
</tr>
<tr>
<td>O(28) + HCCO(34) -&gt; H(17) + CO(13) + CO(13)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>9.64E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O(28) + HCO(31) -&gt; CO(13) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>3.01E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O(28) + HCO(31) -&gt; CO₂(14) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>3.01E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O(28) + HO₂(30) -&gt; O₂(2) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>3.19E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O(28) + O(28) -&gt; O₂(2)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>5.40E+13</td>
<td>0</td>
<td>0</td>
<td>-1.787</td>
</tr>
<tr>
<td>O₂(2) + C₂H₂(22) -&gt; CO₂(14) + CH(18)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>9.05E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + C₂H₂(22) -&gt; HCCO(34) + O(28)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>9.05E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + C₂H₃O(61) -&gt; HO₂(30) + CH₂CO(16)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>7.56E+11</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + C₃H₂(25) -&gt; HCO(31) + HCCO(34)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + C₃H₇(39) -&gt; HO₂(30) + C₃H₆(9)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>3.62E+11</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + C₄H₇(71)1 -&gt; C₄H₇O₂(1436)</td>
<td>Direction = 1</td>
<td>RAdditionMultipleBond</td>
<td>7.54E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + C₄H₇(71)1 -&gt; HO₂(30) + C₄H₆(728)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>7.56E+11</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + CH(18) -&gt; HO(13) + O₂(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.66E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + CH(18) -&gt; CO₂(14) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.66E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + CH₂(19) -&gt; CH₂O(15) + O(28)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>4.20E+12</td>
<td>0</td>
<td>0</td>
<td>1.491</td>
</tr>
<tr>
<td>O₂(2) + CH₂(19) -&gt; CO(13) + H₂O(11)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>1.48E+12</td>
<td>0</td>
<td>0</td>
<td>1.491</td>
</tr>
<tr>
<td>O₂(2) + CH₂(19) -&gt; CO(13) + HO(29) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>8.15E+12</td>
<td>0</td>
<td>0</td>
<td>1.491</td>
</tr>
<tr>
<td>O₂(2) + CH₂(19) -&gt; CO₂(14) + H(17) + H(17)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>5.43E+12</td>
<td>0</td>
<td>0</td>
<td>1.491</td>
</tr>
<tr>
<td>O₂(2) + CH₂(19) -&gt; CO₂(14) + H₂(3)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>5.43E+12</td>
<td>0</td>
<td>0</td>
<td>1.491</td>
</tr>
<tr>
<td>O₂(2) + CH₂(5)²(20) -&gt; CO(13) + OH(29) + H(17)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>3.13E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + CH₂HCO(35) -&gt; C₂H₂O₃(600)</td>
<td>Direction = 1</td>
<td>RAdditionMultipleBond</td>
<td>4.52E+12</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + CH₂HCO(35) -&gt; HO₂(30) + CH₂CO(16)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>6.56E+11</td>
<td>0</td>
<td>0</td>
<td>2.710</td>
</tr>
<tr>
<td>O₂(2) + CH₂O(15) -&gt; HCO(31) + HO₂(30)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>6.02E+13</td>
<td>0</td>
<td>0</td>
<td>40.638</td>
</tr>
<tr>
<td>O₂(2) + CH₂OH(33) -&gt; HO₂(30) + CH₂O(15)</td>
<td>Direction = 1</td>
<td>Disproportionation</td>
<td>2.28E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
</tr>
<tr>
<td>O₂(2) + CH₃(21) -&gt; CH₂O(15) + OH(29)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>3.31E+11</td>
<td>0</td>
<td>0</td>
<td>8.939</td>
</tr>
<tr>
<td>O₂(2) + CH₃(21) -&gt; CH₃O(32) + O(28)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>4.40E+13</td>
<td>0</td>
<td>0</td>
<td>31.383</td>
</tr>
<tr>
<td>O₂(2) + CH₃O(32) -&gt; CH₂O(15) + HO₂(30)</td>
<td>Direction = 1</td>
<td>Leeds</td>
<td>2.17E+10</td>
<td>0</td>
<td>0</td>
<td>1.749</td>
</tr>
<tr>
<td>Reaction</td>
<td>Direction</td>
<td>Leeds</td>
<td>Energy</td>
<td>Temp</td>
<td>Energy</td>
<td>Temp</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-----------</td>
<td>-----------</td>
<td>--------</td>
<td>------</td>
<td>------------</td>
<td>------</td>
</tr>
<tr>
<td>O2(2) + CO(13) -&gt; CO2(14) + O(28)</td>
<td>Direction = 1</td>
<td>Leeds 1.26E+13</td>
<td>0</td>
<td>0</td>
<td>47.038</td>
<td>5.265E-02</td>
</tr>
<tr>
<td>O2(2) + H(17) -&gt; HO2(30)</td>
<td>Direction = 1</td>
<td>Leeds 2.10E+18</td>
<td>-0.8</td>
<td>0</td>
<td>0.000</td>
<td>1.093E+16</td>
</tr>
<tr>
<td>O2(2) + H(17) -&gt; OH(29) + O(28)</td>
<td>Direction = 1</td>
<td>Leeds 9.76E+13</td>
<td>0</td>
<td>0</td>
<td>14.838</td>
<td>2.842E+09</td>
</tr>
<tr>
<td>O2(2) + H2CCCH(26) -&gt; C3H3O2(605)</td>
<td>Direction = 1</td>
<td>Leeds 2.10E+18</td>
<td>0.1474</td>
<td>0</td>
<td>2.533</td>
<td>7.319E+11</td>
</tr>
<tr>
<td>O2(2) + H(17) + CH2CO(16) + HCO(31)</td>
<td>Direction = 1</td>
<td>Leeds 3.01E+10</td>
<td>0</td>
<td>0</td>
<td>2.867</td>
<td>4.002E+09</td>
</tr>
<tr>
<td>O2(2) + HCO(31) -&gt; HO2(30) + CO(13)</td>
<td>Direction = 1</td>
<td>Leeds 1.63E+12</td>
<td>0</td>
<td>0</td>
<td>0.855</td>
<td>8.928E+11</td>
</tr>
<tr>
<td>O2(2) + OH(29) -&gt; O(28) + HO2(30)</td>
<td>Direction = -1</td>
<td>Leeds 3.19E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.172E-02</td>
</tr>
<tr>
<td>O2(2) -&gt; O(28) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds 5.40E+13</td>
<td>0</td>
<td>0</td>
<td>-1.787</td>
<td>2.561E-21</td>
</tr>
<tr>
<td>OH(29) + CH3O(32) -&gt; CH4O2(240)</td>
<td>Direction = 1</td>
<td>Leeds 2.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>2.000E+13</td>
</tr>
<tr>
<td>OH(29) + H(17) -&gt; H2(3) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds 5.12E+04</td>
<td>2.67</td>
<td>0</td>
<td>6.276</td>
<td>4.323E+10</td>
</tr>
<tr>
<td>OH(29) + H2O(11) -&gt; H2O2(12) + H(17)</td>
<td>Direction = -1</td>
<td>Leeds 2.41E+13</td>
<td>0</td>
<td>0</td>
<td>3.963</td>
<td>1.445E-10</td>
</tr>
<tr>
<td>OH(29) + HCCO(34) -&gt; CH2O(15) + CO(13)</td>
<td>Direction = 1</td>
<td>Leeds 1.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.000E+13</td>
</tr>
<tr>
<td>OH(29) + HCCO(34) -&gt; HCO(31) + HCO(31)</td>
<td>Direction = 1</td>
<td>Leeds 1.00E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.000E+13</td>
</tr>
<tr>
<td>OH(29) + HCO(31) -&gt; H2O(11) + CO(13)</td>
<td>Direction = 1</td>
<td>Leeds 1.02E+14</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.020E+14</td>
</tr>
<tr>
<td>OH(29) + H2O(11) + O(28)</td>
<td>Direction = 1</td>
<td>Leeds 2.89E+13</td>
<td>0</td>
<td>0</td>
<td>-0.497</td>
<td>4.100E+13</td>
</tr>
<tr>
<td>OH(29) + H2O(11) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds 6.62E+11</td>
<td>0</td>
<td>0</td>
<td>3.973</td>
<td>2.560E+04</td>
</tr>
<tr>
<td>OH(29) + OH(29) -&gt; H(17) + HO2(30)</td>
<td>Direction = -1</td>
<td>Leeds 1.69E+14</td>
<td>0</td>
<td>0</td>
<td>0.874</td>
<td>2.204E+02</td>
</tr>
<tr>
<td>OH(29) + OH(29) -&gt; H2O2(12)</td>
<td>Direction = 1</td>
<td>Leeds 1.57E+13</td>
<td>0</td>
<td>0</td>
<td>0.000</td>
<td>1.570E+13</td>
</tr>
<tr>
<td>OH(29) + OH(29) -&gt; O(28) + H2O(11)</td>
<td>Direction = 1</td>
<td>Leeds 1.51E+09</td>
<td>1.14</td>
<td>0</td>
<td>0.100</td>
<td>2.525E+12</td>
</tr>
<tr>
<td>OH(29) -&gt; H(17) + O(28)</td>
<td>Direction = -1</td>
<td>Leeds 1.18E+19</td>
<td>-1</td>
<td>0</td>
<td>0.000</td>
<td>1.462E-15</td>
</tr>
<tr>
<td>Reaction</td>
<td>k(T=715K, P=0.73atm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>----------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O4.(901) -&gt; C4H9O2.(626) + O2(2)</td>
<td>180000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O4.(901) -&gt; C4H8O2(789) + HO2(30)</td>
<td>2200</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H4(8) -&gt; H2C2CCH(26) + H(17)</td>
<td>1.50E-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3(23) -&gt; C2H2(5) + H(17)</td>
<td>9.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH3(21) + O2(2) -&gt; CH3O.(54)</td>
<td>8.70E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH3(21) + O2(2) -&gt; CH3(21) + O2(2)</td>
<td>4.40E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2(5) -&gt; C2H(22) + H(17)</td>
<td>1.70E-26</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH4O2(240) -&gt; CH3O(32) + OH(29)</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH4O2(240) -&gt; HO2(30) + CH3(21)</td>
<td>2.70E-09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH4O2(240) -&gt; CH3O2.(54) + H(17)</td>
<td>5.40E-19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + HO2(30) -&gt; C3H4(8) + HO2(30)</td>
<td>7.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + HO2(30) -&gt; C3H5O2.(400)</td>
<td>1.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H4(8) + HO2(30) -&gt; C3H4(8) + HO2(30)</td>
<td>6.00E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5(36) + O2(2) -&gt; C3H5O2(611)</td>
<td>1.80E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5(36) + O2(2) -&gt; C3H4(8) + HO2(30)</td>
<td>2.10E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5(36) + O2(2) -&gt; C3H5(36) + O2(2)</td>
<td>4.30E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2O(16) -&gt; HCCO(34) + H(17)</td>
<td>6.60E-19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5(36) + CO2(14) -&gt; C3H5(36) + CO2(14)</td>
<td>7.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H6O2(168) -&gt; C2H5O.(592) + OH(29)</td>
<td>95</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H6O2(168) -&gt; C2H5(24) + HO2(30)</td>
<td>3.50E-08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H6O2(168) -&gt; C2H5O2.(52) + H(17)</td>
<td>1.60E-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2(626) + O2(2) -&gt; C4H9O4.(901)</td>
<td>2.10E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2(626) + O2(2) -&gt; C4H8O2(789) + HO2(30)</td>
<td>2.10E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2(626) + O2(2) -&gt; C4H9O2(626) + O2(2)</td>
<td>5.30E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2O(15) + C3H5(36) -&gt; CH2O(15) + C3H5(36)</td>
<td>3.70E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H7(711) -&gt; C4H6(278) + H(17)</td>
<td>1.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2HCO(35) + O2(2) -&gt; C2H3O3.(600)</td>
<td>6.00E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2HCO(35) + O2(2) -&gt; CH2HCO(35) + O2(2)</td>
<td>3.90E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2HCO(35) + O2(2) -&gt; CH2HCO(16) + HO2(30)</td>
<td>5.00E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2HCO(35) -&gt; CH2HCO(16) + H(17)</td>
<td>42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2HCO(35) -&gt; CH2HCO(16) + H(17)</td>
<td>3.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(137) -&gt; C4H9O2(58)</td>
<td>1.90E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(137) -&gt; HO2(30) + C4H8(44)</td>
<td>9.60E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(137) -&gt; O2(2) + C4H9(137)</td>
<td>2.60E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(238) -&gt; C4H9O2(49)</td>
<td>1.80E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(238) -&gt; HO2(30) + C4H8(44)</td>
<td>2.40E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(238) -&gt; HO2(30) + C4H8(45)</td>
<td>1.30E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(238) -&gt; O2(2) + C4H9(238)</td>
<td>5.70E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>O2(2) + C4H9(238) -&gt; C4H8(45) + HO2(30)</td>
<td>4.90E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9(137) -&gt; C2H5H(24) + C2H4(6)</td>
<td>33000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9(137) -&gt; C4H8(44) + H(17)</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C5H7O2.(1422) -&gt; C4H7(711) + CO2(14)</td>
<td>4.00E+09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Rate Constant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>-----------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2O(15) + CH3(21) -&gt; CH2O(15) + CH3(21)</td>
<td>3.70E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5(24) -&gt; C2H4(6) + H(17)</td>
<td>36</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O3.(505) -&gt; CH2CO(16) + HO2(30)</td>
<td>160000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O3.(505) -&gt; CH2CO(16) + HO2(30)</td>
<td>47000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O2.(610) -&gt; C2H3(23) + O2(2)</td>
<td>60</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O.(61) -&gt; H(17) + CH2CO(16)</td>
<td>1.50E-05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O.(61) -&gt; CH3(21) + CO(13)</td>
<td>1600000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H3O2.(605) -&gt; H2CCCH(26) + O2(2)</td>
<td>6600000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2CCCH(26) + O2(2) -&gt; C3H3O2.(605)</td>
<td>4.40E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2CCCH(26) + O2(2) -&gt; H2CCCH(26) + O2(2)</td>
<td>2.10E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H8(45) -&gt; C4H7.(711) + H(17)</td>
<td>1.40E-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH2O(15) -&gt; HCO(31) + H(17)</td>
<td>3.50E-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3(23) + O2(2) -&gt; C2H3O2.(610)</td>
<td>4.90E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3(23) + O2(2) -&gt; C2H3(23) + O2(2)</td>
<td>4.00E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H7O2.(1436) -&gt; C4H7.(711) + O2(2)</td>
<td>2.00E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H7O2.(1436) -&gt; C4H6(278) + HO2(30)</td>
<td>22</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H8(44) -&gt; C3H5(36) + CH3(21)</td>
<td>4.10E-08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H8(44) -&gt; C2H3(23) + C2H5(24)</td>
<td>3.40E-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H8(44) -&gt; C4H7.(711) + H(17)</td>
<td>3.20E-12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H10(1) -&gt; C4H9.2(38) + H(17)</td>
<td>8.90E-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H10(1) -&gt; C2H5(24) + C2H5(24)</td>
<td>3.30E-09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H10(1) -&gt; C3H7.(39) + CH3(21)</td>
<td>7.00E-10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H10(1) -&gt; C4H9.1(37) + H(17)</td>
<td>3.30E-15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5O2.(400) + O2(2) -&gt; C3H5O4.(1133)</td>
<td>2.50E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5O2.(400) + O2(2) -&gt; C3H5O2.(400) + O2(2)</td>
<td>4.30E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H6(278) + OH(29) -&gt; C4H6(278) + OH(29)</td>
<td>2.80E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H6(278) + OH(29) -&gt; C4H6(278) + OH(29)</td>
<td>1.40E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H2(10) -&gt; C2H2(22) + C2H(22)</td>
<td>4.10E-35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5(24) + O2(2) -&gt; C2H5O2.(52)</td>
<td>4.90E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5(24) + O2(2) -&gt; C2H5(24) + O2(2)</td>
<td>4.00E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5(24) + O2(2) -&gt; C2H4(6) + HO2(30)</td>
<td>2.60E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H4(6) -&gt; C2H3(23) + H(17)</td>
<td>2.20E-18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H6(7) -&gt; CH3(21) + CH3(21)</td>
<td>3.90E-11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H6(7) -&gt; C2H5(24) + H(17)</td>
<td>3.50E-16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O3.(600) -&gt; CH2CO(35) + O2(2)</td>
<td>260000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O3.(600) -&gt; CH2CO(16) + HO2(30)</td>
<td>82</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O5.(1294) -&gt; C2H3O3.(505) + O2(2)</td>
<td>270000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(49) -&gt; HO2(30) + C4H8(45)</td>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Rate Constant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(49) -&gt; HO2(30) + C4H8(44)</td>
<td>2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(49) -&gt; O2(2) + C4H9(2)(38)</td>
<td>150000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(49) -&gt; C4H8(45) + HO2(30)</td>
<td>590</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5O.(52) -&gt; CH2O(15) + CH3(21)</td>
<td>120000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2O2(12) -&gt; OH(29) + OH(29)</td>
<td>7.40E-04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2O2(12) -&gt; HO2(30) + H(17)</td>
<td>3.70E-18</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5O2.(52) -&gt; C2H5(24) + O2(2)</td>
<td>29000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5O2.(52) -&gt; C2H4(6) + HO2(30)</td>
<td>810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9(2)(38) -&gt; C4H8(45) + H(17)</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9(2)(38) -&gt; C4H8(44) + H(17)</td>
<td>28</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9(2)(38) -&gt; C3H6(9) + CH3(21)</td>
<td>18000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5O.(592) -&gt; CH2O(15) + CH3(21)</td>
<td>120000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H2O2(12) -&gt; OH(29) + OH(29)</td>
<td>7.40E-04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H5O2.(611) -&gt; C2H3(9) + C3H5(36) + O2(2)</td>
<td>2.00E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2H5O(1294)</td>
<td>3.80E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2HCO(16) + HO2(30)</td>
<td>8.30E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2HCO(16) + HO2(30)</td>
<td>3.00E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2H3O3(505)</td>
<td>4.30E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2HCO(16) + HO2(30)</td>
<td>2.40E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2HCO(16) + HO2(30)</td>
<td>8.20E+10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H2O(16) + HO2(30) -&gt; C2HCO(16) + HO2(30)</td>
<td>3.50E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH4(4) -&gt; CH3(21) + H(17)</td>
<td>1.20E-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H7O2.(1435) -&gt; C4H7(711) + O2(2)</td>
<td>2.80E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O3.(505) + O2(2) -&gt; C2H3O5.(1294)</td>
<td>7.10E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C2H3O3.(505) + O2(2) -&gt; C2H3O3.(505) + O2(2)</td>
<td>3.80E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C3H5O2.(400) -&gt; C3H4(8) + HO2(30)</td>
<td>260000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(58) -&gt; HO2(30) + C4H8(44)</td>
<td>880</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(58) -&gt; O2(2) + C4H9(1)(37)</td>
<td>110000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(58) -&gt; C4H9O2.(626)</td>
<td>9300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(58) -&gt; C4H8O2(789) + H(17)</td>
<td>0.0018</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; C4H8O2(789) + H(17)</td>
<td>35</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; C4H9O2.(58)</td>
<td>100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; HO2(30) + C4H8(44)</td>
<td>690</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; O2(2) + C4H9(1)(37)</td>
<td>120000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; C4H9O2.(626)</td>
<td>100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; HO2(30) + C4H8(44)</td>
<td>690</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; O2(2) + C4H9(1)(37)</td>
<td>120000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; C4H9O2.(626)</td>
<td>100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; HO2(30) + C4H8(44)</td>
<td>690</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; O2(2) + C4H9(1)(37)</td>
<td>120000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; C4H9O2.(626)</td>
<td>100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; HO2(30) + C4H8(44)</td>
<td>690</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; O2(2) + C4H9(1)(37)</td>
<td>120000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C4H9O2.(626) -&gt; C4H9O2.(626)</td>
<td>100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reaction</td>
<td>Rate Constant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>---------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>H(_2)CCCCH(27) -&gt; C(_4)H(_2)(10) + H(17)</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH(_2)O(15) + C(_2)H(_3)O(_2).(610) -&gt; CH(_2)O(15) + C(_2)H(_3)O(_2).(610)</td>
<td>1.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + CO(_2)(14) -&gt; C(_5)H(_7)O(_2).(1422)</td>
<td>0.16</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + CO(_2)(14) -&gt; C(_4)H(_7).(711) + CO(_2)(14)</td>
<td>7.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + CO(_2)(14) -&gt; C(_4)H(_7).(711) + CO(_2)(14)</td>
<td>7.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_3)H(_6)(9) -&gt; C(_3)H(_5)(36) + H(17)</td>
<td>7.00E-13</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_3)H(_6)(9) -&gt; C(_2)H(_3)(23) + CH(_3)(21)</td>
<td>4.50E-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + O(_2)(2) -&gt; C(_4)H(_7)O(_2).(1436)</td>
<td>3.50E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + O(_2)(2) -&gt; C(_4)H(_6)(278) + HO(_2)(30)</td>
<td>1.10E+08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + O(_2)(2) -&gt; C(_4)H(_7).(711) + O(_2)(2)</td>
<td>7.20E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + O(_2)(2) -&gt; C(_4)H(_7)O(_2).(1435)</td>
<td>2.60E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_7).(711) + O(_2)(2) -&gt; C(_4)H(_7).(711) + O(_2)(2)</td>
<td>4.30E+12</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_2)H(_4)O(_2).(121) -&gt; CH(_2)HCO(35) + OH(29)</td>
<td>3100000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_2)H(_4)O(_2).(121) -&gt; C(_2)H(_3)(23) + HO(_2)(30)</td>
<td>2.50E-14</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_2)H(_4)O(_2).(121) -&gt; C(_2)H(_3)O(_2).(610) + H(17)</td>
<td>7.90E-20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_3)H(_5)(36) -&gt; C(_3)H(_4)(8) + H(17)</td>
<td>2.10E-05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_3)H(_5)O(_4).(1133) -&gt; C(_3)H(_5)O(_2).(400) + O(_2)(2)</td>
<td>2.70E+07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_3)H(_7).(39) -&gt; C(_3)H(_6)(9) + H(17)</td>
<td>100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_3)H(_7).(39) -&gt; C(_2)H(_4)(6) + CH(_3)(21)</td>
<td>4000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C(_4)H(_6)(278) -&gt; C(_2)H(_3)(23) + C(_2)H(_3)(23)</td>
<td>2.60E-17</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH(_2)O(15) + C(_3)H(_3)O(_2).(54) -&gt; CH(_2)O(15) + C(_3)H(_3)O(_2).(54)</td>
<td>1.30E+11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.3.2 N-butane low-temperature oxidation results

The author shows here four figures on major reactants and products dynamic profiles provided by RMG simulation using our 68 species and 835 reactions n-butane oxidation model.

Figure 8.2 and Figure 8.3 shows the n-butane oxidation and major CO and CO$_2$ production profiles from experimental data and from our RMG model simulation. From those figures, we can see that this model has successfully detected the autoignition phenomena in low temperature oxidation of n-butane. This model predicts a delay time of about 120 seconds for ignition. Comparing to the experimental result provided in Wilk, et al.1995, which showed a delay time of 120 seconds, our RMG n-butane model precisely predicts the delay for the ignition. This is a very good prediction result. However, our model predicted oxidation process after the autoignition is slightly slower than the experimental result. In experimental result, after autoignition, it only took about 200 more seconds to burn 70% of n-butane, while, in our model prediction, it took about 300 more seconds to do so. As to the major product CO and CO$_2$ formation, our RMG model predictions qualitatively agree with the experimental results. However, the yield of CO is lower than experimental results by about a factor of 2.5, and the yield of CO$_2$ is also lower than experimental results by about a factor of 10.
Figure 8.2 n-butane oxidation and CO, CO$_2$ formation (Experimental results from Wilk et al. [1995])

Figure 8.3 n-butane oxidation and CO, CO$_2$ formation (Predictions from our RMG model)
Figure 8.4 Other major products formation (Experimental results from Wilk et al. [1995])

Figure 8.5 Other major products formation (Predictions from our RMG model)
Figure 8.4 and 8.5 compares the predicted profiles for other major products, alkanes and alkenes with the experimental results. The yields of alkene products, such as \( \text{C}_4\text{H}_8 \), \( \text{C}_3\text{H}_6 \), and \( \text{C}_2\text{H}_4 \), are in very good agreement with experimental data. The errors of alkene products prediction are within 20%, comparing to the experimental data. However, the methane formation is lower than the experimental data by factor of 10.

### 8.3.3 Discussion

The n-butane low temperature model generated by RMG shows very good quantitative agreement with the experimental data on the ignition delay and major product yields. The model provides very good prediction on ignition delay time and many major products, such as CO and alkenes. However, the predicted \( \text{CO}_2 \) and methane yields differ from the experimental data by factors of 10. Although errors of factors of 10 may sound alarming, one must remember that the RMG model predictions are pure predictions – zero adjustable parameters – and that an error of 2 kcal/mol in an estimated barrier height, or \( \Delta H \), or an error of 3 cal/mol/K in an entropy estimate, or a couple of factor-of-2 errors in a symmetry number computation would be enough to explain the errors in the predictions. Since the model contains 835 reactions and 68 species, and most of the numbers in the model are computer estimates, and most of the estimates are expected to be significantly uncertain, the level of agreement between experiment and this model is remarkably good.

This error might result from several causes. Firstly, the thermodynamic properties calculations for complex radicals, such as, \( \text{ROO} \cdot \), \( \text{\cdot QOOH} \) and \( \text{\cdot OO-Q-OOH} \), and molecules, such as \( \text{H-OOQOO-H} \), certainly have some non-negligible error bars, since so far we don’t have very detailed oxygen group values and oxygen HBI radical correction values for all kinds of different oxygen groups and oxygen radicals. The error in thermodynamic properties, discussed in chapter 5 and 6, might lead to a big error in the backward reactions kinetics calculation, which is based on the forward kinetics rate rule and the reaction thermodynamic equilibrium constants. To solve this problem, we should do more calculation on the thermodynamic group values for oxygen related groups, so
that the thermodynamic properties estimation for oxygen molecules and radicals will provide better values.

Another likely reason causing the error of RMG model might be the imperfect kinetics rate rules in our present library. Although we tried our best to collect all the available kinetics from literature to fill in our kinetics rate rule library, it still has many unoccupied positions. For those missing rates, RMG searches through the library to find a closest rate rule for such reaction. Sometimes, since a reaction family’s kinetics library has so much missing kinetics, the approximation rate found for a reaction could be quite far away from the true value. To solve this problem, we should keep working on collecting more kinetics rate rules to enrich our rate rule libraries.

In summary, the major problems causing n-butane oxidation problems could be the imperfectness of our thermodynamic and kinetic database, although we have tried our best to collect and include as many as possible data from literature, calculation, and experiments. To improve the model generation quality, we should keep working on enriching our databases to accommodate more and more data with good qualities, so that the errors caused from using approximated data parameters could be eliminated gradually. Fortunately, we provide an easy way to extend our tree structured databases, as discussed in chapter 5 and 6, and, therefore, to include new thermodynamic and kinetic data into RMG will be very easy and quick.

8.4 Conclusions

RMG successfully generated an n-butane low temperature oxidation reaction mechanism with 68 species and 835 reactions. So far, this is the first trial of generating the reaction mechanism for such a complex reaction system in a completely automated way among all available reaction model generation software. The generated n-butane oxidation model shows quantitative agreement with experimental data in predicting the ignition time delay and the yields of the major products, such as CO and alkenes. However, the simulation results shows a quantitative disagreements of about a factor of 10 in CO₂ and CH₄ formation compared with the experimental data.
The disagreement between the experimental data and the model predictions may come from the errors of our present imperfect thermodynamic and kinetic databases. With the improvement of those databases by adding more and more precise data, RMG will provide much better predictions. Furthermore, since RMG offers chemistry users an easy and quick way to update all the RMG's databases, this task will be relatively easy to fulfill.
Chapter 9  Valid Parameter Range Analysis

From chapter 3 to chapter 8, the author has discussed the issue of generating a reaction mechanism for defined reaction conditions of a reaction system. For example, as shown in chapter 7, people should specify the temperature, pressure, initial concentrations, etc., at the initialization of model generation process. In this section, the author will discuss an algorithm for identifying a valid parameter range for a reaction mechanism that was developed and studied earlier in her thesis.

When the author developed the algorithm and studied the examples for valid range analysis problem, RMG hasn’t been developed yet; therefore, the first version of model generation software, XMG, was used to generate reaction mechanism examples for this study, and primary results were published in Chemical Engineering Science.

Here, the author used those examples from XMG to illustrate the fundamental algorithms for valid range analysis, which are implemented exactly the same way in our second version reaction model generation software, RMG. Therefore, although in this chapter the author uses old “XMG” notation to introduce valid parameter range analysis algorithm and used all XMG generated reaction mechanisms, the fundamental algorithm is also applicable for RMG.

9.1 Motivations

After we are able to generate a reaction model, it grows ever more important to assess the accuracy of their predictions, and to identify the range of conditions over which the models are expected to be valid. A considerable amount of effort has been put into understanding how uncertainties in the model parameters and inputs propagate through to uncertainties in the model predictions. Here we address the second issue: is the structure of the model valid at these reaction conditions?

Kinetic models are generally only valid over a limited range of reaction conditions, and in fact the person constructing the model usually has a particular set of conditions in mind when he or she decides which processes to include in the model.
However, since it still takes a considerable amount of effort to construct a kinetic model, models are frequently used in conditions far from where their authors intended, and this sometimes leads to very serious errors in the model predictions. Unfortunately, until now it has not been possible to identify the valid range for any given kinetic model, so one generally does not know whether a kinetic model prediction should be accurate. Here we make a first step towards identifying any model’s valid range, and towards constructing chemical kinetic models known to be valid over a user-specified range.

There are two aspects to constructing a chemical kinetic model. The first concerns the structure of the model: enumerating the chemical species and reactions in the model. The second concerns the parameters in the model: technical parameters such as rate constants, and operating parameters that describe the reaction conditions, such as the initial concentrations, temperature, and pressure. It is well known that model predictions depend on these parameter values, but not so widely appreciated that the very structure of the model ought to also depend on these values. Since it is usually impossible to construct a kinetic model that is truly comprehensive and appropriate for all reaction conditions, these two aspects are necessarily linked: the structure of the model should change if the parameters vary enough. For example, one expects a combustion model to differ from a model for low temperature partial oxidation, and a model of styrene polymerization to differ from a model of styrene pyrolysis. Similarly, when it is determined that a reaction previously thought to be negligible actually has a high rate constant, one must add that reaction (and often several daughter reactions) to the model, changing its structure.

In general, it is desirable to use a kinetic model whose structure is tailored to the reaction conditions of interest: these models are smaller, easier to understand, and easier to solve than models based on larger, more comprehensive reaction mechanisms. On the other hand, for a kinetic model to be useful, its structure should be constant over some range of parameters. The invariance of the model structure as the parameters are varied is a critical assumption of conventional parametric sensitivity analysis and uncertainty quantification. It is also an implicit assumption in most simulations, where the same chemical kinetic model is used at all times and spatial positions. (Though since the chemistry can change dramatically as a reaction proceeds, this assumption is not always
necessary or desirable [Green and Schwer, 2001; Green, et al., 2001; Schwer, et al., 2002]. Sometimes the change in the chemistry as a reaction proceeds can be very abrupt, causing numerical problems, see for example [Manca, et al., 2001].) We intuitively expect that any reasonable model’s structure will be unchanged by “small” changes in the parameter values. But how small is “small”? 

In this thesis, that author presents a new method for connecting the model structure with the parameter values, and so to identify a range of conditions and parameter values where a kinetic model is valid; the new method can be coupled with model generation codes to construct models valid over a user-specified parameter range; and it also can provide a measure of the “model truncation error”, i.e. the error due to the fact that the model of interest is not comprehensive. This method provides a new perspective to understand the relations between the structure of a kinetic model, the size of the model’s valid parameter range, and the associated truncation error tolerance.

The problems of identifying the valid ranges of reduced kinetic models and generating a reduced model valid over a user-specified range have recently been addressed by Sirdeshpande et al. [2001]. In that paper, the authors used fairly exhaustive sampling to map out the valid ranges for several illustrative reduced models based on a relatively small (20 reaction) complete model. As they pointed out, it will not usually be practical to map out the complete valid range by sampling, and instead one is looking for simple representations of subsets of the multidimensional valid range, for example polyhedra or hyper-rectangles. In the present work, we also use the hyper-rectangle representation of the valid range, but focus on much more complex kinetic systems (> 1000 reactions). To reduce computing time, we use a local-sensitivity based approach rather than sampling. This approach is much more practical for systems with a large number of parameters.

In the reduced-model case studied by Sirdeshpande et al., one normally has available the original larger model, which is assumed to be valid over the entire of parameter range of interest. This same assumption has recently been made by Bhattacharjee and Oluwole in this research group in order to construct software that
automatically constructs a valid range of a reduced model, recently demonstrated at Supercomputing 2003. Comparison between the reduced model and the original model allows one to directly compute the model truncation error, and so to set unambiguous bounds on the valid range of the reduced model.

The problem of identifying the valid range of a newly generated model, addressed here for the first time, is much more difficult. No comprehensive model is available, and no explicit criteria exist for defining the valid parameter range, so there is no way known to precisely determine the model truncation error. As shown below, we can nonetheless set useful criteria bounding the valid range of the model. The way to set up such criteria is closely related to the rate-based iterative algorithm for model construction [Susnow, et al., 1997] discussed in Chapter 7, which we briefly review in the next section, and in practice our new method is used in concert with computerized model generation codes based on that algorithm, such as XMG and RMG.

9.2 Methodology for valid parameter range analysis

As we have discussed, a reaction model is generally constructed at a fixed parameter point \( \theta_0 \), called the nominal point. The author proposed a method to quantify the relations between the model structure and the valid parameter range around the nominal parameter point.

The methodology for valid parameter range analysis includes two main parts: developing criteria bounding that valid parameter range, and regulating the valid parameter range into a hyper-rectangle expression.

### 9.2.1 Criteria defining valid parameter range

#### 9.2.1.1 Parameter range validity criteria

As stated earlier, a reaction model is complete if the rate-based termination rule (7.3) is satisfied. Noticing that the rate-based termination rule is tested at the nominal parameter point \( \theta_0 \), we can re-write (7.3) into:
For a reaction model, criteria (9.1) is only known to be satisfied at the nominal parameter point \( \theta_0 \); we can imagine that with a small change of parameters \( \Delta \theta \) around their original values \( \theta_0 \), the termination criteria (9.1) might be still satisfied so that the model structure would still be valid. Therefore, testing the rate-based termination rule in a parameter range instead of at the nominal parameter point can easily extend the original rate-based termination rule into a new criteria defining the valid parameter range, as shown below:

\[
r_j(t, \theta_0 + \Delta \theta) < R_{\min}(t, \theta_0 + \Delta \theta) = \varepsilon \cdot R_{\text{char}}(t, \theta_0 + \Delta \theta) \quad \text{for} \quad j = 1, \ldots, m
\]

\[
t_0 \leq t \leq \tau
\]

Inequalities (9.2) are called parameter range validity criteria, and if a kinetic model satisfies (9.2), the model structure is considered valid in the parameter range \( \theta_0 + \Delta \theta \). It should be noted that these criteria have to be tested for all the edge species during the whole reaction time period.

9.2.1.2 Constraints setup

Criteria (9.2) is just an implicit expression, and in order to make it explicitly regulate the valid range, the author expanded about the nominal parameter point \( \theta_0 \) for (9.2). In practice, for convenience, the author used logarithm scale for rates \( r_j \) and \( R_{\min} \) in Taylor Expansion to gain a simple expression, and also omitted the 2\textsuperscript{nd} and higher-order items in the Taylor Expansion to make it a linear approximation. The detailed derivation is shown as following:
\[ \log(r_j(t, \theta_0 + \Delta \theta)) < \log(R_{\text{min}}(t, \theta_0 + \Delta \theta)) \]

\[ \Rightarrow \log(r_j(t, \theta_0)) + \frac{\partial \log(r_j)}{\partial \theta}(t) \bigg|_{\theta_0} \cdot \Delta \theta < \log(R_{\text{min}}(t, \theta_0)) + \frac{\partial \log(R_{\text{min}})}{\partial \theta}(t) \bigg|_{\theta_0} \cdot \Delta \theta \]

\[ \Rightarrow \log(r_j(t, \theta_0)) + \frac{1}{r_j(t, \theta_0)} \cdot \frac{\partial r_j}{\partial \theta}(t) \bigg|_{\theta_0} \cdot \Delta \theta < \log(R_{\text{min}}(t, \theta_0)) + \frac{1}{R_{\text{min}}(t, \theta_0)} \cdot \frac{\partial R_{\text{min}}}{\partial \theta}(t) \bigg|_{\theta_0} \cdot \Delta \theta \]

\[ \Rightarrow \left( \frac{1}{r_j(t, \theta_0)} \cdot \frac{\partial r_j}{\partial \theta}(t) \bigg|_{\theta_0} - \frac{1}{R_{\text{char}}(t, \theta_0)} \cdot \left( \sum_i \frac{\partial R_i}{\partial \theta}(t) \bigg|_{\theta_0} \cdot R_i(t, \theta_0) \right) \right) \cdot \Delta \theta < \log \left( \frac{R_{\text{min}}(t, \theta_0)}{r_j(t, \theta_0)} \right) \]

\[ \cdot: R_{\text{min}} = \varepsilon \cdot R_{\text{char}} = \varepsilon \cdot \sqrt{\sum_i R_i^2(t, \theta_0)} \]

\[ \Rightarrow \left( \frac{1}{r_j(t, \theta_0)} \cdot \frac{\partial r_j}{\partial \theta}(t) \bigg|_{\theta_0} - \frac{1}{R_{\text{char}}(t, \theta_0)} \cdot \left( \sum_i \frac{\partial R_i}{\partial \theta}(t) \bigg|_{\theta_0} \cdot R_i(t, \theta_0) \right) \right) \cdot \Delta \theta < \log \left( \frac{\varepsilon \cdot R_{\text{char}}(t, \theta_0)}{r_j(t, \theta_0)} \right) \]

Defining \( s_j(t, \theta_0) = \frac{\partial r_j}{\partial \theta}(t) \bigg|_{\theta_0} \) and \( S_i(t, \theta_0) = \frac{\partial R_i}{\partial \theta}(t) \bigg|_{\theta_0} \), we obtain an implicit definition of the valid parameter range:

\[ \left( \frac{s_j(t, \theta_0)}{r_j(t, \theta_0)} - \frac{1}{R_{\text{char}}(t, \theta_0)} \cdot \sum_i R_i(t, \theta_0) \cdot S_i(t, \theta_0) \right) \cdot \Delta \theta < \log \left( \frac{\varepsilon \cdot R_{\text{char}}(t, \theta_0)}{r_j(t, \theta_0)} \right) \]

(9.3)

\[ t_0 \leq t \leq \tau \quad \text{and for } j = 1, \ldots, m \]

where \( \theta_0 = [\theta_{10}, \theta_{20}, \ldots, \theta_{p0}]^T \) represents the nominal parameter point at which the kinetic model is generated; \( \Delta \theta = [\Delta \theta_1, \Delta \theta_2, \ldots, \Delta \theta_p]^T \) represents the parameter variation of the system; \( r_j(t, \theta_0) = \frac{dC_j}{dt}(t, \theta_0) \) represents the formation rate of \( j^{th} \) edge species and

\( R_i(t, \theta_0) = \frac{dC_i}{dt}(t, \theta_0) \) represents the rate of change of \( i^{th} \) core species;

\( s_j(t, \theta_0) = \frac{\partial r_j}{\partial \theta}(t) \bigg|_{\theta_0} = [s_{j1}, s_{j2}, \ldots, s_{jp}] = \left[ \frac{\partial r_j}{\partial \theta_1}(t) \bigg|_{\theta_0}, \frac{\partial r_j}{\partial \theta_2}(t) \bigg|_{\theta_0}, \ldots, \frac{\partial r_j}{\partial \theta_p}(t) \bigg|_{\theta_0} \right] \) is a row vector representing the rate sensitivity coefficients of \( j^{th} \) edge species and
\[ S_i(t, \theta_0) = \left. \frac{\partial R_i(t)}{\partial \theta} \right|_{\theta_0} = [S_{i1}, S_{i2}, \ldots, S_{ip}] = \left[ \frac{\partial R_{i1}(t)}{\partial \theta_1} \bigg|_{\theta_0}, \frac{\partial R_{i2}(t)}{\partial \theta_2} \bigg|_{\theta_0}, \ldots, \frac{\partial R_{ip}(t)}{\partial \theta_p} \bigg|_{\theta_0} \right] \]

is a row vector representing the rate sensitivity coefficient vector of \(i^{th}\) core species. \(M\) is the total number of the edge species and \(\tau\) is the reaction time scale.

In order to set up (9.3) into a concrete form, we need the information on the rates and rate sensitivities, \(r_j(t, \theta_0), s_j(t, \theta_0), R_i(t, \theta_0),\) and \(S_i(t, \theta_0),\) which can be directly obtained by solving the initial problem of the reaction system ODEs and doing sensitivity analysis (SA) on it. Solving system ODE/SA problem has been discussed in details in chapter 7, when the author introduced the dynamic simulator for RMG system. Here, we can use same ODE/SA solver, like DASPK [Li, S. and Petzold, 1999] and DSL48S [Feehery et al., 1997; Feehery, 1998] to get the solutions on those rate and rate sensitivities.

It has been stated that inequalities (9.3) should be tested over the whole continuous reaction time range of interest. However, because most ODE/SA problems for complex reaction systems can only be numerically solved at a finite number of time points, it is practically impossible (and unnecessary) to check (9.3) for a continuous time domain. In practice, the author discretizes the time domain, sample the flux-time curves at discretized time points, and test parameter range validity criteria (9.3) at sample time points instead. Although this time-discretizing approximation may introduce error, such error could always be reduced by increasing the number of sampled time points.

After the time discretization, the parameter range validity criteria become:

\[
\left( \frac{s_j(t, \theta_0) - \sum_i R_i(t, \theta_0) \cdot S_i(t, \theta_0)}{r_j(t, \theta_0) - R_{char}^2(t, \theta_0)} \right) \cdot \Delta \theta < \log \left( \frac{e \cdot R_{char}(t, \theta_0)}{r_j(t, \theta_0)} \right)
\]

(9.4)

For all \(j = 1, 2, \ldots, m\)

For all \(t = t_0, t_1, t_2, \ldots, \tau\)
Criteria (9.4) should be satisfied at each sampled time point for each edge species. Therefore, the total number of the constraints in (9.4) is the product of the number of edge species and the number of time points we sampled. If we put together the constraints for all the edge species and all the sampled time points, (9.4) becomes a matrix expression:

\[
A \cdot \Delta \theta < b
\]  

(9.5)

where \( A \) is called the rate sensitivity matrix, \( b \) is called the rate distance vector, and \( \Delta \theta \) is the parameter changes. Notice that \( A \in \mathbb{R}^{n \times p}, \Delta \theta \in \mathbb{R}^p, b \in \mathbb{R}^n \), in which \( n \) is the total number of the constraints, and \( p \) is the number of the parameters of interest.

It is noteworthy that, in the development of parameter range validity criteria, the author only used the first term and omitted the higher terms in the Taylor Expansion. This is equivalent to using the rate sensitivity at the nominal parameter point \( \theta_0 \) to linearly predict the species rates in a parameter range. All the range analyses results discussed in the following sections are based on this linear approximation at the nominal parameter point \( \theta_0 \). Higher-order sensitivities methods can be used to move beyond the linear approximation, but at much greater computational cost.

### 9.2.2 Regulate Valid Parameter Range into Hyper-rectangle

#### 9.2.2.1 Valid parameter range: a characteristic polyhedron

Constraints set (9.5) defines a polyhedron in the parameter space. This polyhedron, \( \Phi \), is uniquely determined by the model and \( \theta_0 \), represented by:

\[
\Phi = \{ \Delta \theta | A \cdot \Delta \theta \leq b \} \quad A \in \mathbb{R}^{n \times p}, \Delta \theta \in \mathbb{R}^p, b \in \mathbb{R}^n
\]  

(9.6)

A schematic example of a characteristic polyhedron with two parameters' variation is shown in Figure 9.1.
The characteristic polyhedron reflects the model's flexibility in parameter space. The larger the characteristic polyhedron, the more flexible the model is. In addition, if the polyhedron could be drawn, we could easily tell its corners and edges corresponding to the extreme parameter conditions, beyond which the model is invalid. Furthermore, the shape of the polyhedron might elucidate the crucial parameter(s) for the model structure's validity.

9.2.2.2 Hyper-rectangle expression of valid parameter range

Although (9.6) has defined a characteristic polyhedron as the valid parameter range, it is difficult to directly visualize and understand such a complex-shaped parameter range, especially in multi-parameter cases. Traditionally, a parameter range is represented as hyper-rectangle that clearly defines the upper and lower bounds of each independent parameter, which can be expressed as below:

$$
\Gamma = \{ \Delta \theta | \Delta \theta_{\text{low}} \leq \Delta \theta \leq \Delta \theta_{\text{up}} \} \quad \Delta \theta, \Delta \theta_{\text{low}}, \Delta \theta_{\text{up}} \in \mathbb{R}^p
$$

where $\Gamma$ is the valid hyper-rectangle parameter range, $\Delta \theta$ is the parameter variation, $\Delta \theta_{\text{low}}$ and $\Delta \theta_{\text{up}}$ are the lower bound and the upper bound of the parameter variation, and $p$ is the number of parameters involved.
Using a hyper-rectangle expression has many benefits. It is straightforward to directly visualize the actual valid parameter range, to quickly understand the primary parameter(s) limiting the usage of a model, and to easily apply such valid range information into other related research.

9.2.2.3 Regulation of the valid parameter range from a polyhedron into a unique hyper-rectangle

The goal of this regulation is to find the unique largest hyper-rectangle inscribed in the characteristic polyhedron; in other words, it is to use a largest hyper-rectangle space to approximate the original characteristic polyhedron. This type of problem, called feasibility analysis, has been thoroughly studied in the work of Swaney and Grossmann [1985a].

The key difficulty in feasibility analysis is that many distinct hyper-rectangles could be inscribed in the characteristic polyhedron, which is illustrated in Figure 9.2. The existence of multiple solutions is due to the undetermined scale of each parameter. If the scale of each parameter is not fixed, there will be infinite number of vertex directions in each quadrant in the parameter space, and the number of inscribed hyper-rectangles will also be infinite.
In order to obtain a unique solution, Swaney and Grossmann [1985a] proposed that user define the expected parameter variation vectors, $\Delta \theta^-$ and $\Delta \theta^+$, which uniquely define an expected hyper-rectangle, and that hyper-rectangle be scaled in size until it just fits within the characteristic polyhedron. This approach is illustrated in Figure 6. After the expected hyper-rectangle had been defined by $\Delta \theta^-$ and $\Delta \theta^+$, any similar hyper-rectangle can be expressed mathematically as:

$$\Gamma = \{ \Delta \theta \mid -d \cdot \Delta \theta^- < \Delta \theta < d \cdot \Delta \theta^+ \}$$

(9.8)

where $d$ was originally named “flexibility index” by Swaney and Grossmann. Here, we call it “flexibility”, and it represents the scale of the hyper-rectangle.

Combining (9.8) with the original constraints (9.5) results in the mathematical expression $\Gamma$ for the hyper-rectangle with flexibility $d$ inside the original characteristic polyhedron:

$$A \cdot \Delta \theta < b$$

$$\forall \Delta \theta \in \Gamma = \{ \Delta \theta \mid -d \cdot \Delta \theta^- < \Delta \theta < d \cdot \Delta \theta^+ \}$$

(9.9)

where $A$ and $b$ are the same definitions as in (9.5).

### 9.2.3 Three types of important range analyses

In Swaney and Grossmann’s paper on feasibility analysis [1985a], the relation of different designs for a chemical plant and their flexibilities was studied, and a method for solving Eqs. (9.9) was proposed. In this thesis, the author is also interested in another important factor of this problem: the effects of the error tolerance. Error tolerance is essential to the range analyses because it is necessary to understand how much error must be tolerated if a model is to be used in a parameter range. Although traditionally error tolerance is set as a given constant, the author treats it as a variable, and its effect for the range analysis can be shown explicitly:
\[ A \cdot \Delta \theta < \log(\varepsilon) + b_{\text{char}} \]
\[ \forall \Delta \theta \in \Gamma = \{ \Delta \theta \mid -d \cdot \Delta \theta^- < \Delta \theta < d \cdot \Delta \theta^+ \} \]

where \( A \) is the same as in (9.9), \( \varepsilon \) is the error tolerance, and \( b_{\text{char}} = \log \left( \frac{R_{\text{char}}}{r_f} \right) \).

In (9.10), there are three important variables, the user-specified tolerance \( \varepsilon \), the valid range \( \Delta \theta \), and the model structure that determines the \( A \) and \( b_{\text{char}} \). Knowing any two of them uniquely bounds the third one. Accordingly, there are three types of analyses from inequalities (9.10):

Type 1: Valid Parameter Range Identification. Given a kinetic model (which determines \( A \) and \( b_{\text{char}} \)) and the tolerance the user requires (\( \varepsilon \)), calculate the valid parameter range (\( \Delta \theta \)) where the model can be used;

Type 2: Robust Model Generation. Generate a kinetic model valid over a specified parameter range (\( \Delta \theta \)) to the specified tolerance (\( \varepsilon \)).

Type 3: Tolerance Estimation. Given a kinetic model (which determines \( A \) and \( b_{\text{char}} \)) and a parameter range (\( \Delta \theta \)), calculate the corresponding tolerance (\( \varepsilon \)) involved, when that model is used over the given parameter range.

All three types of problems are essential in the study on chemical kinetics; however, before this work, very few methods have been proposed to solve these problems. The problem of identifying the valid parameter range for a reduced kinetic model has been studied in Sirdeshpande, et al. [2001], and an automated software package for constructing valid parameter ranges of reduced models has recently been demonstrated by Bhattacharjee and Oluwole, but the valid parameter range identification for a full kinetic model and the problems of robust model generation and error estimation have never been studied before.

The inequalities (9.10), first built in this work, clarify the relation between the tolerance, valid parameter range, and model structure, and make it possible to solve those three types of problems directly.
9.2.4 Type 1: Valid parameter range identification

9.2.4.1 Problem statement and solving procedure

Problem statement

Given a model (with its edge species and reactions) valid at $\theta_0$, a user-specified error tolerance $\varepsilon$, and a user-specified parameter scaling expressed as bounds on an parameter variation "expected hyper-rectangle" $[\theta_0-\Delta \theta^-, \theta_0+\Delta \theta^+]$, the problem is to find the model's flexibility "d" about $\theta_0$.

The solution procedure is to perform sensitivity analysis on the model at the nominal point $\theta_0$, and evaluate the corresponding $A$ and $b_{\text{char}}$ (Eqs. 6-7 and 12). Then one can solve for $d$ by a single scan through the large matrix $A$ (Eq. 11):

$$d = \min \left( \log(\varepsilon) + b_{\text{char}}^j \right) - \sum_{k=1}^{p} |A_{jk}| \cdot \Delta \theta_k$$

(9.11)

$$\Delta \theta_k = \begin{cases} \Delta \theta_k^+ & \text{if } A_{jk} > 0 \\ \Delta \theta_k^- & \text{if } A_{jk} < 0 \end{cases}$$

Once the optimal $d$ is calculated, the valid parameter range can be expressed as:

$$\theta_0 - d \cdot \Delta \theta^- < \theta < \theta_0 + d \cdot \Delta \theta^+$$

(9.12)

Linearization Error Test

The error involved in valid parameter range identification is mainly from the linear approximation to the Taylor expansion. The error can in principle be estimated by calculating the higher-order rate sensitivities, but, unfortunately, these cannot be efficiently calculated at present. In order to check the magnitude of this linearization error, we tested the vertices of the identified hyper-rectangle. Specifically, this check is to solve the ODE and confirm it satisfies Eqs. (9.2) at each vertex of the hyper-rectangle in parameter space defined by Eq. (9.12). If all of the vertices pass this test, we are reasonably confident that the linear approximation doesn’t introduce significant error. Fortunately, in all the calculations we have done so far, we haven’t encountered any
vertex check failures. An automated procedure for dealing with vertex checking failures is under development. For cases with a large number of parameters, one would not be able to check all $2^p$ vertices; in such case, we will test only those vertices predicted by the sensitivity analysis to lie close to the boundary of the valid parameter range.

It is impossible to prove the model is valid over the complete range, as the active constraints in Eq.(9.2) may not be convex or pseudo-convex. Therefore, the purpose of this vertex checking is only for testing the error associated with the linear approximation.

**Running time**

The running time of this approach consists of three main parts: time for solving the ODE/SA problem at the nominal parameter point, time for solving (9.11), and time for checking vertices by solving the ODE at each vertex. For the case of methane pyrolysis with two parameters varied, we observed that solving (9.11) takes less than 1% of the total running time, and solving the ODE/SA problem and checking vertices are the bottleneck of the whole calculation. In this two-parameter case, the CPU time for solving the ODE/SA problem and vertex checking are comparable. However, if the number of parameters is increased to $p=10$, the running time will be dominated by the cost of vertex checking, and it may no longer be practical to exhaustively check all $2^p$ vertexes.

The entire process for identifying the valid parameter range of a given model is summarized in Figure 9.3.
Reading an XMG model

Calculating Rates and Rate Sensitivities by DASPK

User-specified $\Delta \theta^-$ and $\Delta \theta^+$

Setting up $A$ and $b_{\text{char}}$

Solving problem (9.11) to obtain flexibility $d$

Checking Vertices of valid range

Vertices passed?

Outputting information

Valid parameter range:
$\theta_0 - d \cdot \Delta \theta^- < \theta < \theta_0 + d \cdot \Delta \theta^+$

Figure 9.3. Valid Parameter Range Identification process

9.2.4.2 Examples: initial concentrations of methane pyrolysis

To test this method, the author applied it into the case of methane pyrolysis with the existence of a small amount of ethane, and the varied parameters we studied were initial concentrations of the reactants. All the parameters used to generate the model are listed in Table 9.1.

<table>
<thead>
<tr>
<th>$\varepsilon$ (%)</th>
<th>$T$ (K)</th>
<th>$P$ (atm)</th>
<th>$X_{[\text{CH}_4]}$ (%)</th>
<th>$C_0$ (mol/cm$^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>1038</td>
<td>0.38</td>
<td>0.3</td>
<td>[CH$_4$]: $6.81 \times 10^{-4}$ [C$_2$H$_6$]: $1.00 \times 10^{-14}$</td>
</tr>
</tbody>
</table>

Table 9.1 Parameter settings in methane pyrolysis

In Table 9.1, $\varepsilon$ is the error tolerance to generate a kinetic model; $T$ is temperature; $P$ is pressure; $X_{[\text{CH}_4]}$ is the conversion of methane; $C_0$ are the initial concentrations of
reactants. The model generated at these parameter conditions has 44 species and 1229 reactions, with an edge of 1179 “edge” species formed by 3073 “edge” reactions. In the following section, we represent any model by the number of the species and reactions it includes; for example, we call this methane pyrolysis model “44/1229”.

<table>
<thead>
<tr>
<th>$\Delta C_0^-$ (mol/cm$^3$)</th>
<th>$\Delta C_0^+$ (mol/cm$^3$)</th>
<th>Number of time points sampled</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CH$_4$]: 2.00×10$^{-6}$</td>
<td>[CH$_4$]: 2.00×10$^{-6}$</td>
<td>10</td>
</tr>
<tr>
<td>[C$_2$H$_6$]: 2.00×10$^{-14}$</td>
<td>[C$_2$H$_6$]: 2.00×10$^{-14}$</td>
<td></td>
</tr>
</tbody>
</table>

**Table 9.2 Settings in valid parameter range analysis for methane pyrolysis model**

In Table 9.2, $\Delta C_0^-$ and $\Delta C_0^+$ are the expected parameter variation vectors. Here, $\Delta C_0^-$ and $\Delta C_0^+$ are set equal for convenience, and the valid parameter range is therefore a symmetric hyper-rectangle centered at the nominal parameter point. The continuous time domain is discretized evenly into 10 time points.

Using the valid range identification method discussed in last section, with the settings for the expected parameter variation and sample points number shown in Table 9.2, the author solved the flexibility of this model: $d = 0.837$. Combining this flexibility with the expected parameter variation vector, the valid initial concentration range for this system was obtained:

$$5.14 \times 10^{-6} \text{ mole/cm}^3 < [\text{CH}_4]_0 < 7.48 \times 10^{-6} \text{ mole/cm}^3$$

$$0 < [\text{C}_2\text{H}_6]_0 < 1.68 \times 10^{-12} \text{ mole/cm}^3$$

The vertex check confirmed the model is valid over this parameter range.

The author also did similar valid range identification for different methane pyrolysis models. Using the same parameter settings in Table 9.1, except that the user-defined tolerance was respectively set to 0.2%, 0.3%, 0.5%, 0.8%, 1.0%, and 1.2%, the author generated six different methane pyrolysis models, shown in Table 9.3.
<table>
<thead>
<tr>
<th>User-defined Tolerance $\varepsilon$</th>
<th>Model Structure</th>
<th>Flexibility $d$</th>
<th>Valid range for $C_0$ (mol/cm$^3$)</th>
</tr>
</thead>
</table>
| 0.2%                            | 44/1229 / 1179/3073 | 0.837 | $[\text{CH}_4]: 5.14 \times 10^{-6} - 7.48 \times 10^{-6}$  
$[\text{C}_2\text{H}_6]: 0 \sim 1.68 \times 10^{12}$ |
| 0.3%                            | 41/1179 / 1037/2520 | 1.196 | $[\text{CH}_4]: 4.42 \times 10^{-6} \sim 9.20 \times 10^{-6}$  
$[\text{C}_2\text{H}_6]: 0 \sim 2.40 \times 10^{12}$ |
| 0.5%                            | 35/785 / 838/1828 | 0.583 | $[\text{CH}_4]: 5.64 \times 10^{-6} \sim 7.98 \times 10^{-6}$  
$[\text{C}_2\text{H}_6]: 0 \sim 1.18 \times 10^{12}$ |
| 0.8%                            | 33/681 / 781/1654 | 0.983 | $[\text{CH}_4]: 4.84 \times 10^{-6} \sim 8.78 \times 10^{-6}$  
$[\text{C}_2\text{H}_6]: 0 \sim 1.98 \times 10^{12}$ |
| 1.0%                            | 32/625 / 736/1581 | 0.0425 | $[\text{CH}_4]: 6.73 \times 10^{-6} \sim 6.90 \times 10^{-6}$  
$[\text{C}_2\text{H}_6]: 0 \sim 9.50 \times 10^{14}$ |
| 1.2%                            | 30/580 / 688/1376 | 0.263 | $[\text{CH}_4]: 6.28 \times 10^{-6} \sim 7.34 \times 10^{-6}$  
$[\text{C}_2\text{H}_6]: 0 \sim 5.36 \times 10^{-13}$ |

Table 9.3 Valid initial concentration range identification for methane pyrolysis

Note that in Table 9.3, all the other parameters, except $\varepsilon$, used to generate models are the same as in Table 9.1. All the parameters settings for calculating the flexibility are the same as in Table 9.2.

From the results, it is noted that the flexibility of a model could be fairly small, e.g. $d = 0.0425$ of the model 32/625 in Table 3. This is because, when a model is generated, the rate-based termination rule (9.1) is only satisfied at the nominal parameter point $\theta_0$, and the flexibility of a model is not considered during the model generation process. Once a reaction model is generated, its flexibility could be any positive value. A question arises accordingly: can we combine the flexibility calculation with the model generation process to automatically generate a robust model valid over a user-defined parameter range?
9.2.5 Type 2: Robust model generation

9.2.5.1 Problem statement and solving procedure

Problem statement

The goal is to generate a model valid to an error tolerance \( \varepsilon \) not only at the nominal parameter point \( \theta_0 \) but also over a user-specified parameter range:

\[
\theta_0 - \Delta \theta^- < \theta < \theta_0 + \Delta \theta^+
\]

We call this process "robust model generation".

Algorithm description

The algorithm developed to solve this problem was based on the method for valid range identification and the model generation process. The idea of this algorithm is to calculate the flexibility of the current model to check if it is large enough to satisfy the user’s requirement. If it is not, the program goes back to XMG to generate a larger model. The main procedure of this algorithm is shown in Figure 9.4.

Specifically, this algorithm consists of two main parts: (1) flexibility calculation and (2) model enlargement. In part (1), once XMG has finished generating a model, we add our valid parameter range identification program to calculate the flexibility of the model. During the parameter range identification, the expected parameter variation vectors are set equal to the user-defined parameter variation, so the flexibility \( d = 1.0 \) is a threshold for determining if the user’s requirement on parameter variation is satisfied. If the calculated flexibility of a model is equal to or greater than 1.0, this model satisfies the user’s requirement and is output as the final model; otherwise, the model is still too small to be used in the user-defined parameter range, and it is necessary to generate a larger, more complete model.
In part (2), XMG is called again to enlarge the present model. XMG normally terminates after generating a model at $\theta_0$, so we must re-start the model generator if part (2) is needed. In practice, we simply shrink down the tolerance used in XMG, so that the termination rule for the present model might be violated and the model generation process could be activated again. Note that the tolerance used for practically enlarging an XMG model is different from the user-defined error tolerance, $\varepsilon$, which defines the user’s error tolerance. To distinguish them, we define this adjusted tolerance “practical tolerance”, $\varepsilon_p$. Initially, practical tolerance $\varepsilon_p$ is set equal to user-specified tolerance $\varepsilon$. If the practical tolerance needs shrinking, we multiply $\varepsilon_p$ by a number less than 1.0, called the “reduction factor”, $\gamma$. The reduction factor could be any fraction between 0 and 1.0. From our experience, we suggest setting $\gamma$ between 0.6–0.9. With the reduced $\varepsilon_p$, XMG will re-start and iterate to generate a larger model. This process is repeated until finally a robust model with its flexibility $d > 1.0$ is generated.

The running time of the robust analysis subroutine itself is about the running time of a few XMG iteration loops. Adding this subroutine does not increase the running time
significantly. In practice, we observed that the running time of robust model generator was about the same order of magnitude with the running time of the original XMG without the robust analysis subroutine.

9.2.5.2 Example: initial concentrations of methane pyrolysis

The author applied this algorithm to the same methane pyrolysis reaction system. The parameters of interest are still the initial concentrations of methane and ethane. All the other parameters were set as in Table 9.1. The author chose the case of user-specified tolerance $\varepsilon = 1.0\%$ because, for this case, XMG gave a model with a small flexibility, $d = 0.0425$. The reduction factor is set: $\gamma = 0.8$.

While the original XMG without flexibility calculation generated a model of 32 species and 625 reactions, the flexibility of which is only 0.0425, the updated XMG with flexibility calculation resulted in a model with 33 species and 681 reactions, which has a much better flexibility $d = 2.219$. That result shows that user can use this model in the initial concentration range

$$2.37\times 10^{-6} \text{ mole/cm}^3 < [\text{CH}_4]_0 < 1.125\times 10^{-5} \text{ mole/cm}^3$$

$$0 < [\text{C}_2\text{H}_6]_0 < 4.45\times 10^{-12} \text{ mole/cm}^3$$

without exceeding an error tolerance of 1.0%. During this robust model generation, the practical tolerance $\varepsilon_p$ used in XMG was reduced from its original value 1.0% to 0.8%, so that XMG ran one more iteration to enlarge the model from model $32/625$ to model $33/681$.

Again, we did similar analysis for another XMG model generated at user-specified tolerance of 0.5%. The results of those two cases are summarized in Table 9.4. From these examples, it is clear that the key to this algorithm is to use a suitably smaller practical tolerance than the user-specified tolerance to generate a robust model to meet the user’s requirements for the valid parameter range.
Table 9.4 Robust model generation for methane pyrolysis

In Table 9.4, all the other parameters, except ε, to generate models in XMG are the same as in Table 9.1. All the parameters settings for calculating the flexibility are the same as in Table 9.2. Reduction factor γ = 0.8.

9.2.6 Type 3: Tolerance estimation

Problem statement and solving procedure

Problem statement

The goal of tolerance estimation is to find the necessary error tolerance ε if a given model is to be used over a defined parameter range Γ. This tolerance is a measure of the model’s truncation error, and it can be evaluated by a single scan through A:

\[
ε = \exp \left( \max_{j} \left( \sum_{k=1}^{p} |A_{jk}| \Delta θ_k - b_j^{\text{char}} \right) \right) \tag{9.13}
\]

\[
Δθ_k = \begin{cases} 
Δθ_k^+ & \text{if } A_{jk} > 0 \\
Δθ_k^- & \text{if } A_{jk} < 0 
\end{cases}
\]
The whole procedure for tolerance estimation is shown in Figure 7.

**Solving procedure**

The mathematical formulation of this problem is similar to min-max optimization problem in valid range identification. The solving procedure and its running time are also similar to those in valid parameter range identification.

The whole procedure for tolerance estimation is shown in Figure 9.5.

---

**Figure 9.5 Tolerance estimation process**

9.2.6.2 Examples: initial concentrations of methane pyrolysis

The author chose the model 44/1229 in Table 9.3 as the studied model, and the user-specified parameter range is set as:

\[
4.81 \times 10^{-6} \text{ mole/cm}^3 < [\text{CH}_4]_0 < 8.81 \times 10^{-6} \text{ mole/cm}^3 \\
0 < [\text{C}_2\text{H}_6]_0 < 2.01 \times 10^{-12} \text{ mole/cm}^3,
\]
The resulting lower bound of tolerance is: $\varepsilon = 0.21\%$. This implies that if the model 44/1229 is used over this initial concentration range, the maximal formation rate of the neglected edge species is about 0.21\% of the characteristic flux $R_{\text{char}}(t)$. If the user can accept this error tolerance, model 44/1229 is useful over this range of conditions; if the user wants higher accuracy, he or she should generate a larger model using the robust model generation algorithm described above.

9.3 Relation of model, flexibility and tolerance

9.3.1 Constraints relating model, flexibility, and tolerance

In last section for three types of range analyses, we have observed that one of the solving algorithms for feasibility analysis, the direct search algorithm, included a procedure to explore all the vertex directions of the valid parameter range [Swaney and Grossmann, 1985b]. Here, we combine the vertex examination process with the original validity parameter criteria (9.4) to derive an explicit relation between a model, its flexibility, and the error tolerance. Substituting all the vertices into criteria (9.4) gives:

$$\left(\frac{s_j(t, \theta_0)}{r_j(t, \theta_0)} - \frac{\sum_i R_i(t, \theta_0) \cdot S_i(t, \theta_0)}{R_{\text{char}}^2(t, \theta_0)}\right) \cdot \Delta \theta_v \cdot d < \log(\varepsilon) + \log \left(\frac{R_{\text{char}}(t, \theta_0)}{r_j(t, \theta_0)}\right)$$

For all $j = 1, 2, \ldots, m$

For all $t = t_0, t_1, t_2, \ldots, \tau$

For all $\Delta \theta_v = \Delta \theta_v^{(1)} \cdot \Delta \theta_v^{(2)} \ldots, \Delta \theta_v^{(p)}$

Let $\alpha(j, t, \Delta \theta_v) = \left(\frac{s_j(t, \theta_0)}{r_j(t, \theta_0)} - \frac{\sum_i R_i(t, \theta_0) \cdot S_i(t, \theta_0)}{R_{\text{char}}^2(t, \theta_0)}\right) \cdot \Delta \theta_v$ and $\beta(j, t) = \log \left(\frac{R_{\text{char}}(t, \theta_0)}{r_j(t, \theta_0)}\right)$,

it can be written more compactly:
For all \( j = 1, 2, \ldots, m \)

\[
\alpha(j, t, \Delta \theta_v) \cdot d < \log(\varepsilon) + \beta(j, t)
\]

\( \text{For all } t = t_0, t_1, t_2, \ldots, \tau \)

\( \text{For all } \Delta \theta_v = \Delta \theta_{v_1}, \Delta \theta_{v_2}, \ldots, \Delta \theta_{v_p} \)

Inequality (9.15) explicitly defines a set of constraints relating the tolerance \( \varepsilon \), the flexibility \( d \), and the model structure (represented by \( \alpha \) and \( \beta \)). Each of the constraints in (9.15) corresponds to checking the parameter range validity criteria for one edge species at a specific reaction time and at a specific vertex of the hyper-rectangle valid parameter range. For any kinetic model, once the expected parameter range variation vectors have been defined and the time domain has been decided and discretized, all the \( \alpha \) and \( \beta \) in (9.15) can be then determined, so that we can draw all the constraints in (9.15) in a flexibility ~ tolerance graph. (In practice, one is only concerned about the constraints that become active over the parameter range of interest. When constructing the graph, one can easily avoid computing most of the redundant constraints.)

9.3.2 Flexibility ~ tolerance graph for a methane pyrolysis model

The flexibility ~ tolerance graph provides important information on the valid parameter range of a kinetic model. To illustrate its importance and to explain the way to make use of it, we studied the methane pyrolysis model 44/1229. In this case, all the parameters for model generation and valid range analyses are the same as in Table 9.1 and Table 9.2. Based on the rates and rate sensitivities from ODE/SA solver DASPK 3.0 [Li, S. and Petzold, 1999], we successfully built up the set of constraints (9.15) and drew the flexibility ~ tolerance graph for model 44/1229, as shown in Figure 9.6.
In this graph, we can observe several important features.

(1) In this graph, the relation of $d \sim \log(\varepsilon)$ is defined by a set of lines, each of which is corresponding to one constraint in (9.15). The total number of all the lines in flexibility ~ tolerance graph, equals to the total number of constraints in (9.15), which is the product of the number of edge species, the number of the time points sampled, and the number of the vertices of the hyper-rectangle. In this case, there are 2 varied parameters, so the vertex number is $2^2$, there are 1179 edge species, and 10 time points are sampled; therefore, the number of constraints for model 47/1229 is: $N = 1179 \times 10 \times 2^2 = 47160$. In Figure 9.6, the 12 lowest of the 47160 constraint lines are shown.
(2) Only the lowest constraint(s) in the $d \sim \log(\varepsilon)$ graph are active; other constraints are redundant. Noticing that the lowest constraint(s) might not be a single line; instead, it could be a piecewise line, consisting of different constraints in different tolerance ranges. We call this lowest piecewise line the “active constraint”. In Figure 9.6, the active constraint are line 7, when $\varepsilon = 0.1\%$–$0.7\%$, and line 1, when $\varepsilon = 0.7\%$–$10\%$.

(3) This active constraint defines the relation between the flexibility and the error tolerance for a given model, and indicates whether a model is valid in a parameter range under some tolerance. For example, in Figure 10, the active constraint shows that if one demands a $1.0\%$ error tolerance, that model can be used inside the initial concentration range defined by $d = 6.36$. If someone wants to use this model in an initial concentration range defined by $d = 2.00$, he or she has to tolerate at least $0.27\%$ error.

(4) Each constraint line in (9.15) corresponds to one edge species at a specific time point at a specific vertex of the valid parameter hyper-rectangle range. The description of the meanings of the lines in Figure 9.6 is shown in Table 9.5

<table>
<thead>
<tr>
<th>Constraint #</th>
<th>Edge species</th>
<th>Time (s)</th>
<th>Conversion</th>
<th>Vertex*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2,3-diethyl but-2yl radical</td>
<td>1053</td>
<td>0.0030</td>
<td>(+, +)</td>
</tr>
<tr>
<td>2</td>
<td>2,3-diethyl but-2yl radical</td>
<td>987</td>
<td>0.0027</td>
<td>(+, +)</td>
</tr>
<tr>
<td>3</td>
<td>2-methyl butane</td>
<td>1053</td>
<td>0.0030</td>
<td>(+, +)</td>
</tr>
<tr>
<td>4</td>
<td>2-methyl butane</td>
<td>987</td>
<td>0.0027</td>
<td>(+, +)</td>
</tr>
<tr>
<td>5</td>
<td>n-propane</td>
<td>1053</td>
<td>0.0030</td>
<td>(+, +)</td>
</tr>
<tr>
<td>6</td>
<td>n-propane</td>
<td>987</td>
<td>0.0027</td>
<td>(+, +)</td>
</tr>
<tr>
<td>7</td>
<td>Hex-1-ene-5-yl radical</td>
<td>1053</td>
<td>0.0030</td>
<td>(+, +)</td>
</tr>
<tr>
<td>8</td>
<td>Hex-1-ene-5-yl radical</td>
<td>987</td>
<td>0.0027</td>
<td>(+, +)</td>
</tr>
<tr>
<td>9</td>
<td>But-1-ene-2-yl radical</td>
<td>1053</td>
<td>0.0030</td>
<td>(+, +)</td>
</tr>
<tr>
<td>10</td>
<td>But-1-ene-2-yl radical</td>
<td>987</td>
<td>0.0027</td>
<td>(+, +)</td>
</tr>
<tr>
<td>11</td>
<td>n-propane</td>
<td>1053</td>
<td>0.0030</td>
<td>(+, +)</td>
</tr>
<tr>
<td>12</td>
<td>n-propane</td>
<td>987</td>
<td>0.0027</td>
<td>(+, +)</td>
</tr>
</tbody>
</table>

Table 9.5 Description of constraints in Figure 9.6
In Table 9.5, the vertex is represented by $(+/-, +/-)$. Plus sign means the upper bound of the parameter variation is effective, and minus sign means the lower bound of the parameter variation is effective. For example, $(+, +)$ represents the vertex at the lower bound for $[\text{CH}_4]_0$ and the upper bound for $[\text{C}_2\text{H}_6]_0$.

In Figure 9.6, the active constraint consists of line 7 and line 1. From Table 9.5, we know that line 7 corresponds to species hex-1-ene-5-yl radical at vertex $(+, +)$ and at time of $t = 1053$ s, when the conversion of 0.003 is achieved, and line 1 corresponds to species 2,3-dimethyl but-2-yl radical at vertex $(+, +)$ also at the time $t = 1053$ s. From such information, we know that the critical situation that might cause this model to be invalid is that the neglected species hex-1-ene-5-yl and 2,3-dimethyl-but-2-yl become relatively important when the methane conversion is around $0.3\%$, if the initial concentrations of methane and ethane are near the upper bound of their ranges. Therefore, we are able to understand what caused the invalidity of a model and identify the missing important species, and so update the model into a better one.

The flexibility ~ tolerance graph for a model provides much important information on how a model can be used in a valid range with a certain error tolerance, and what is critical for the model structure.

**9.3.3 Flexibility ~ tolerance ~ model graph**

If the user specifies different tolerance requirements, XMG will generate different models with different sizes, such as the six methane pyrolysis models shown in Table 9.3. We expect that a larger model should be valid for a larger parameter variation range than a smaller model, but we don’t know how much better the bigger model is. By displaying the active constraints for different models of the same reaction system on the same flexibility ~ tolerance graph, the performance of each model can be clearly understood. We call this graph “flexibility ~ tolerance ~ model graph”.

For illustration, the author compared the active constraints of the six methane pyrolysis models in Table 9.3 to form a flexibility ~ tolerance ~ model graph for the methane pyrolysis system, shown in Figure 9.7.
From this graph, we can draw several conclusions:

(1) The more detailed a model is, the higher its active constraint lies. In Figure 9.7, we can see that the active constraint of largest model 44/1229 lies above the active constraints of all the smaller models; lower than that, there are the five other active constraints for model 41/1179, model 35/785, model 33/681, model 32/625, and model 30/580 in the decreased order of the model size. Generally, a more detailed model will be suitable for a larger flexibility under the same tolerance, and a more detailed model will allow a smaller tolerance $\varepsilon$ when used in the same parameter range.

(2) Generally, the critical constraints for different models should not intersect with each other. If we use $\Omega(M)$ to represent the space bounded by active constraint of a model M, this conclusion can be also written as:

$$\Omega(M_{30/580}) \subseteq \Omega(M_{32/625}) \subseteq \Omega(M_{33/681}) \subseteq \Omega(M_{35/785}) \subseteq \Omega(M_{41/1179}) \subseteq \Omega(M_{44/1229})$$  \hspace{1cm} (9.16)
where, \( M_{30/580} \) is the model 30/580 in Table 3, and so on. In this example, \( \Omega(M) \) for each model monotonously increases when the model size increases, and we expect that this will usually be the case.

(3) There is an absolute tolerance lower bound, \( \varepsilon_L \), for each model. It is the intersection point of the active constraint and the X-axis. At this tolerance, the model’s flexibility equals zero, which means that although the model is valid at the fixed nominal parameter point \( \theta_0 \), an infinitesimal variation in the parameters will make it invalid. For a tolerance less than \( \varepsilon_L \), the model is not even valid at \( \theta_0 \). The absolute tolerance lower bounds \( \varepsilon_L \) for all six models are shown in Table 9.6.

<table>
<thead>
<tr>
<th>Model</th>
<th>Model 30/580</th>
<th>Model 32/625</th>
<th>Model 33/681</th>
<th>Model 35/785</th>
<th>Model 41/1179</th>
<th>Model 44/1229</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \varepsilon_L )</td>
<td>1.10%</td>
<td>0.99%</td>
<td>0.74%</td>
<td>0.41%</td>
<td>0.21%</td>
<td>0.16%</td>
</tr>
</tbody>
</table>

Table 6. Absolute tolerance lower bounds for methane pyrolysis models

(4) The three types of analyses discussed earlier can be easily achieved by this graph. It directly visualizes the relation between three variables, model structure, its flexibility, and the error tolerance; therefore, we can obtain the results for three types of analyses by checking this graph. For example, if the model is given, the error tolerance information and model’s flexibility are directly linked by its active constraint, and the results for range identification and error tolerance estimation can be obtained directly; if the flexibility and error tolerance are specified, the robust models are the ones whose active constraints lie above the point representing the specified flexibility and error tolerance.

(5) The distance between any of the two active constraints represents the improvement in the flexibility as model size increases. With such information, we can compare models’ performance easily. For example, in Figure 9.7, if a tolerance of 1.0% is required, we will find that model 32/625 has a very small flexibility; from model 32/625 to model 33/681 there is a large improvement in the flexibility: from \( d = 0.0425 \) of model 32/625 to \( d = 2.164 \) of model 33/681 by adding only 1 more species and 56
more reactions. With such a small overhead to gain a great increase in the flexibility, we really should choose model 33/681 instead of model 32/625. However, at the same tolerance, the improvement from model 33/681 to model 35/785 is not that impressive, because adding 2 more species and 104 more reactions increases the model’s flexibility by less than 25%. Notice that for a different tolerance $\varepsilon$, we might have different result. For example, if $\varepsilon = 4.0\%$, the active constraints of the models 32/625, 33/681, and 35/785 are almost overlapped. In this case, choosing a larger model has little improvement in flexibility, so the smallest model 32/625 among those three is enough.

In summary, the author concludes that the flexibility ~ tolerance ~ model graph is a powerful tool for the valid parameter range analyses. If a set of models corresponding to a reaction system has been generated and the corresponding flexibility ~ tolerance ~ model graph has be made, we can easily obtain useful valid parameter range information of the system. In addition, the piecewise linear active constraints for different models are easy to obtain and store. Once a model is generated, we can calculate its active constraint(s) and store it as a special feature of the model for later use.

A practical drawback of this analysis is that one flexibility ~ tolerance ~ model graph only corresponds to fixed expected parameter variation vectors and a fixed time scale; changing any of these will change the graph.

9.4 Generalization of valid range analysis algorithm

The range analysis method proposed in this work can be summarized in three major steps:

step 1: A criteria defining the completeness of a model at a single parameter point $\theta_0$:

$$ F(M, \theta_0, \varepsilon) < 0 $$

(9.17)

step 2: Taylor Expansion for criteria (18) to obtain a criteria defining the validity for a model over a parameter range around $\theta_0$:

$$ G(M, \Delta \theta, \varepsilon) < 0 $$

(9.18)
step 3: Feasibility Analysis to regulate the valid parameter range defined by (19) into a hyper-rectangle:

\[ \theta_{\text{low}} < \theta < \theta_{\text{up}} \]  \hspace{1cm} (9.19)

The general criteria (9.17) at a single parameter point are the prerequisites for the valid range analysis. If a model’s validity at even a single parameter point cannot be determined, how can we judge the model’s validity in a parameter range? So the criteria for a single point are necessary for the range analysis. Unfortunately, validity criteria for kinetic models have not been well studied, and most experts on chemical kinetics only have some fairly raw criteria in their minds. The criteria we used in this paper are the only criteria now in the literature to clearly define the validity of a general kinetic model at a specified parameter condition, and these criteria practically work well in computerized model generation. It is noteworthy that the range analysis method presented in this paper can be applied to other criteria. For example, although few validity criteria for a general kinetic model have been published, many criteria for reducing a kinetic model from some known “full” model can be found in the literature [Petzold, et al., 1997; Edwards, et al., 1998; Androulakis, 2000; Bhattacharjee et al., 2000]. Beginning with those criteria, we could apply our range analysis method to determine the valid parameter range for the reduced model.

In step 2, a Taylor Expansion is used to approximately relate the parameter range variation, error tolerance, and model structure. Taylor expansion has been widely used in sensitivity analysis related research in chemical kinetics. The present work is a new application of sensitivity analysis, this time to help define the parameter variation’s effects on model validity. There is an alternative to the Taylor expansion approach proposed in Sirdeshpande, et al. [2001], which was mainly based on sampling many points over the parameter range of interest. Their approach does not require sensitivity information, but presumably the CPU time required by the sampling approach scales exponentially with the number of parameters.
In step 3, feasibility analysis, carefully studied in Swaney and Grossmann, is a well-known procedure for characterizing a valid parameter range for a constrained system.

This three-step procedure forms the kernel of our range analysis method, based on which we further developed detailed algorithms for three different range analyses: valid parameter range identification, robust model generation, and error tolerance estimation. Among these three problems, the last two are first proposed and solved in this paper, and a new perspective is provided to understand the flexibility of a kinetic model.

Furthermore, we observed that a simpler expression of the relations of model structure, flexibility, and error tolerance could be clarified by a flexibility ~ tolerance ~ model graph. Each model’s flexibility feature is represented by a piecewise linear active constraint in such graph, which can be easily calculated and stored as a special feature of the studied model. With such a graph, we can easily obtain the results of three types of range analyses and conveniently compare different models.

As we have discussed above, the shortcomings of the range analyses methods in this paper mainly originate from the first-order approximation in the Taylor Expansion, which will result in a linear space approximating the real valid parameter space. The present method is a one-point approximation method, which is based on the rates and sensitivity information at the nominal parameter point. This could cause trouble when the first-order approximation introduces a large error. We are in the process of building an iterative approach to repeat such analyses at different parameter points to reduce the error introduced by the linear approximation.

### 9.5 Conclusions and discussions

In this chapter, the author proposed an algorithm extending the kinetic model validity criteria for a single parameter point into the criteria for validating the parameter range over which a kinetic model is valid. General methods for three important valid range analyses are presented, and some good results for the methane/ethane pyrolysis examples are achieved. Moreover, a straightforward flexibility ~ tolerance ~ model graph
is used for the first time to clarify the relation between a kinetic model's structure, its valid parameter range, and the error tolerance. The demonstration for this method was made using XMG, but the same technique could be used with models constructed using NetGen or the new RMG program described earlier in this thesis.
Chapter 10  Conclusions and Recommendations

10.1 Conclusions

This work provides new software, RMG, for automatic reaction mechanism generation. Compared to the first generation software for reaction model generation, this work significantly improves the chemistry flexibility and extendibility by designing and modeling reaction families in a new data-model driven method and by constructing and managing the thermodynamic and kinetic libraries with large number of parameters into new hierarchical structured databases.

RMG, designed by the unified modeling language, UML, and developed by the advanced programming method, object-oriented technology, facilitates software reusability and extendibility, so that maintaining, modifying, and extending RMG will be quite simple and efficient.

RMG is successfully applied to generate a reaction mechanism for n-butane low temperature oxidation, which includes a complex autoignition process. The model generated by RMG caught the fundamental phenomena of autoignition, and the predicted ignition delay time and many major products’ yields are in very good agreement with experimental data. The worst absolute error between our model prediction and experimental data is only about factor of 10, consistent with the uncertainties in some key rate parameters in the model. This is the first time that model generation software automatically generated such a complicated reaction mechanism without human interference, and provided precise predictions on ignition delay and major products yields well consistent with experimental data. The result also guides the directions of improving our RMG thermodynamics and kinetics library in the future.

In the following section, the author summarizes the achievements and contributions of RMG in different aspects.
10.1.1 Chemical structure representation

Using 2-dimensional connectivity graphs, RMG successfully represents chemical species in nature. Different from much other chemical structure modeling software, which use matrix representations of chemical graphs, RMG uses adjacency lists for chemical graph representation, since chemical structures are all sparse graphs. The resonance isomers representation problem has been successfully solved in RMG by assembling all the different hybrids of one species into a collection. In this way, any species with more than one resonance isomer will maintain all its hybrids to participate in different types of the reactions with correct rate rules.

Besides chemical species, in RMG, we also introduced a very important chemical structure, functional group, to represent a group of chemical species with similar reacting center. Introducing and implementing functional group into RMG makes it possible to represent a reaction family in a data-model driven way, so that users are able to flexibly define any kinds of reactions families of interest and to modify the present reaction family’s definition outside of the program. Furthermore, functional group definition also enables subcategorizing one big reaction family into smaller subgroups with different kinetics rates, so that we can accommodate a large number of kinetics parameters accounting for a tremendous level of chemical details.

Associated with functional group objects, the author also successfully developed a general algorithm for efficiently identifying the sub-graph relationships between a chemical species and a functional group, so that RMG has a correct and efficient way to categorize species into functional groups. This algorithm is also responsible for identifying the sub-graph relation between two functional groups, which are used later in hierarchical tree structured thermodynamics and kinetics database management to identify and test the correctness of the relationships between tree nodes at different levels. This algorithm is used everywhere in RMG to detect the reactant candidate, to search the best matched kinetics parameters for an individual reaction, and to identify the proper thermodynamic groups for thermodynamic property estimation, etc. Therefore, it is the soul of RMG to function correctly. This is the first time this algorithm has been
implemented in automatic reaction modeling software, and it provides and enables flexible, handy ways for defining chemistry details.

10.1.2 Thermodynamics property estimation

In RMG, the group additivity method for thermodynamic properties estimation proposed by Benson and coworkers [1976] has been implemented. HBI method, a better estimation method for radical thermal properties proposed by Lay and coworkers [1995], is also implemented. Therefore, RMG is able to estimate all kinds of species in a C/H/O system.

Not only implementing the group additivity method, the author also proposed to model all Benson’s thermodynamics groups and Lay’s radical groups into RMG functional groups, so that the partition of a chemical species into thermal groups can be easily and generally implemented based on our fundamental sub-graph matching algorithm. With this new method, the thermodynamic estimator was implemented in a much simpler way, and the program can be easily maintained and extended.

Furthermore, we also provided a new methodology for systematically managing a large number of thermal group values into hierarchical tree structure. J. Yu successfully built hierarchy trees for holding all the thermodynamic groups. With this hierarchical structure database, we can construct, maintain, and extend the database quickly, efficiently, and correctly. The author also implemented a quick search algorithm to find the best-matched thermal group through the thermodynamic group tree. This is the first time that hierarchy structure has been used to model and manage a thermodynamic groups library, and it is also the first time any software has implemented such hierarchy tree structured database in thermodynamics estimation. This method has proved to be flexible, extendable, and efficient compared to all other similar programs.

10.1.3 Reaction template’s definition

Because of introducing the functional group concept into RMG, we offer users a very flexible way to define reaction families. People can simply draw functional group graphs for reactant candidates and indicate the graph mutation actions to define the
reaction types that they want RMG to include, without touching any of the source code of RMG. To define a new reaction family, it only takes users several minutes to write down a very simple text file. This method, so-called data-model driven reaction family specification, is proposed for the first time in reaction modeling to flexibly handle the reaction families definition issue.

In this work, we also provided 34 reaction families to model hydrocarbon pyrolysis and oxidation systems, and this reaction family set is the richest one that has ever been used in any automatic reaction model generation software. Moreover, since RMG is not limited to having only those 34 reaction families, and people can redefine and add more reaction families if they need to, RMG will be able to be applied in all kinds of different complicated reaction systems in the future.

10.1.4 Hierarchical structured kinetics database

Handling a large number of kinetics rates is a very crucial and hard issue for much reaction modeling software. Since the reaction modeling programs before RMG do not have efficient ways to subcategorize each reaction family into subfamilies with detailed levels, they are not able to easily and quickly distinguish the difference in kinetics between reactions in the same reaction family. In RMG, because of the introduction of the functional groups concept, we can subcategorize any big reaction family into as many subgroups as we want. Furthermore, we proposed a new methodology based on a hierarchical tree structure to systematically divide a big reaction family into small subfamilies. With those thoroughly subcategorized reaction families, we could hold a very large number of kinetics rate rules for all our subfamilies, so that we are able to estimate good kinetics for every specific reaction.

Not only using functional group to represent the subfamilies, we proposed to use hierarchy tree structure to help arrange all those small subfamilies’ kinetics into a hierarchical structured database. Since the hierarchical tree structure systematically breaks down large global reaction families into local small subgroups, constructing, maintaining, and extending such a tree structured database are quite straightforward for people to understand. Following this hierarchical tree idea, we successfully constructed
the kinetics tree structured database for each forward reaction family, and we also tried our best to collect as many as possible rate rules from the literature to enrich our kinetics libraries. This method has been proved to be very efficient for accommodating and organizing many kinetics parameters. Moreover, since searching a hierarchy tree structured database is much more efficient than searching an unordered list, as implemented in much previous reaction modeling software, RMG can search through a large kinetics database within a very short time. Therefore, so far, RMG is considered as the best automatic model generation software handling large kinetics libraries systematically and efficiently.

In a word, the functional groups modeling on subfamilies makes it possible to include multiple kinetics rate rules into one single reaction family, and hierarchical tree structured kinetics database methods enable a systematic management of a large number of kinetics rate rules. Those two new methods bring RMG remarkable ability for handling complex kinetics libraries.

### 10.1.5 Reaction model generation algorithm

The original rate-based iterative model generation algorithm proposed by Susnow and coworkers [1997] has been applied in RMG. Dr. D. Matheu [2003] improved this original rate-based algorithm to include pressure dependent networks explored on the fly during reaction model generation, and his algorithm is implemented in RMG. Those two algorithms have been used in successfully modeling several important pyrolysis reaction systems. RMG extends the usage of those algorithms in modeling the n-butane low temperature oxidation process, which is a very complicated oxidation procedure, and the result shows qualitatively good agreement with the experimental data.

### 10.1.6 Valid range analysis

In this work, the author also proposed and developed a practical algorithm for analyzing the valid parameter range for a reaction kinetics model generated by a rate-based reaction generation algorithm. The algorithm was developed based on the rate-based termination rule provided in Susnow and coworker [1997]. In this algorithm, the
author analyzed first-order sensitivity coefficients at local reaction conditions to estimate the system dynamics in larger parameter ranges, and combining them with the rate-based validity rule, the author related the valid range, the error tolerance, and the reaction mechanism structure. Such relations can also be visualized to provide users a clear picture of the reaction system robustness. The applications of this method into a methane pyrolysis case have been presented in this work, too. The major results for this part of the work have been published in *Chem. Eng. Sci.* [Song et al., 2001].

### 10.1.7 Object-oriented technology and UML

As shown above, in RMG, we have many good ideas on how to handle the complicated reaction system with a large amount of data. However, having the right ideas does not guarantee that it is implemented into a good software. RMG is a large software with more than 60,000 lines. Without a good architecture design, a program with such size can soon become some “spaghetti” code that nobody is willing to use and maintain.

In order to build a good architecture for RMG to achieve those new design ideas in flexibly modeling chemical reaction system, the author applied the most advanced system modeling methodology, unified modeling language, into RMG software system analysis and architecture design. UML provides a clear, easily-understood way to visualize system objects and their relations; therefore, it is very helpful in explaining and recording system structure and in documenting software implementation details. In RMG, every package is designed, implemented, and documented in standard UML format, and those object diagrams are used in this thesis to help explain the design and implementation of RMG.

Implementation of software is another important issue that software developers should pay attention to, since any careless implementation might cause some structural problems later. People used to use procedural languages in implementing scientific application programs. Nowadays, object-oriented technology has been widely used and proved to be an efficient and advanced programming methodology, and corresponding languages, like C++, Java, and C#, have been more and more used in large software
development. In RMG, the author also applied object-oriented technology in software
design and implementation to make it well structured, easily reused and easily extended.
Java, as a popular object-oriented technology, has been used in implementing RMG.

The detailed documentation for all RMG packages, which is over 500 pages, and
the source code of RMG, which is over 60,000 lines, are too long to be attached to this
thesis, and readers can access the original UML files of RMG to learn the design and
implementation details.

Since RMG was designed and implemented in a good structure, RMG runs very
efficiently comparing to the first version of reaction modeling software. In the case of n-
butane oxidation, where RMG generates a reaction model with 68 core species and 835
core reactions, the total running time on a PC with 1.7GHz and 512MB memory is about
30 min. Usually, for the old version of XMG generating similar size of reaction models
could take hours, even when XMG was run on a much faster SGI machine.

RMG is designed and developed by UML and object-oriented method, and
therefore, it is easy to reuse and extend the object components in current RMG packages.
To maintain the good architecture and the software reusability and extendibility, the
author suggests that all software developers that will add new features to RMG follow the
UML standard design and implementation steps.

10.2 Recommendations on future work

Although RMG already possesses many advanced features, it needs
improvements in the following aspects.

10.2.1 Chemical structure representation

Present RMG models chemical species and functional groups of C/H/O system
using 2-dimensional connectivity graphs. This data structure is enough to describe the
structure connectivity information used in detecting species’ fundamental reacting
activity.
However, if people want to investigate more advanced features, such as the cis and trans configuration of a double bond, bond lengths and angles, and if a cycle structure is planar or not, etc., they need more information on the orientation of each atom in a chemical graph. To include those advanced features into RMG, we need to extend the present Graph and ChemGraph objects to include the information of atom size, 3-D atom coordinates, density of the electrons, etc.

On the other hand, if people are interested in modeling a reaction system other than a C/H/O system, (for example, much combustion process includes the formation of NO<sub>x</sub> species), including additional elements will be necessary. In order to let RMG generate such a reaction mechanism, RMG ChemGraph and FunctionalGroup class should be able to recognize nitrogen atom, its valence, and its reacting activity. Therefore, people should add this element, its corresponding information into the fundamental Atom dictionary defined in RMG. This implementation is also very straightforward for RMG. Of course, the corresponding functional groups defining the reacting activity and thermodynamic group information should also be added into our reaction family collection and thermodynamics databases.

10.2.2 Thermodynamics property estimation

The present thermodynamic properties estimation is able to, theoretically, provide thermodynamics estimation for any species in the C/H/O system. The only problem is that we don’t have good-quality data for every thermodynamic group, and we have to assign some approximate number for them, which might cause some big discrepancy between the RMG calculation and experimental data. The improvement could be made by doing quantum calculations to construct better group value results, which can be added into our thermodynamic library to replace the present approximate group values with large error bars. Such improvements have to be done for some important oxygen radicals, whose thermodynamic properties affect the oxidation process greatly, if people are using RMG to model oxidation reaction systems.

Another issue associated with thermodynamic properties estimation is how to handle cyclic structures. In RMG, cyclic structure, as a global correction to a chemical
species, is treated roughly. RMG will simply identify all the distinct cycles included in a species, and add corrections for each of them one by one. This procedure works fine with simple cyclic structures, but, for very complicated chemical species that include bipartite structure, there could be some trouble. This issue will also arise in modeling polycyclic aromatic species, in which many rings are heavily connected to each other, so that to figure out the global correction is not easy. J. Yu et al. [2003] proposed a new bond-centered group-additivity method for estimating complex aromatic structures, and the quantum calculations show the result of her algorithm works very well with many large aromatic species. Hopefully, she will add this new algorithm into RMG soon, and RMG will then perform better in estimating multi-ring species that lead to soot formation.

10.2.3 Kinetics database improvements

Since RMG’s model generation process highly depends on the quality of thermodynamic and kinetic parameters, it requires users to put in enough parameters with good precision level so that it can perform correctly. Therefore, updates and improvements of our libraries are very urgent for RMG to produce correct models and prediction of reaction systems.

In RMG, we constructed all the reaction family kinetic databases into tree structures, so that we are able to manage a very large number of reaction system parameters. This makes it possible to add, modify, and update RMG system parameters easily and efficiently. However, a large part of the current kinetics libraries are empty, since we are short of the precise kinetic rate rules to fill in our library. As a result, RMG uses a lot of approximated numbers for many reactions’ kinetics, which could cause a big deviation when we simulate the system dynamics. Even for the data already included in our library, many of them have large uncertainties. Thorough investigation and test of the qualities of our present kinetics library parameters is quite important. Therefore, we need to put more attention on collecting more kinetics data and evaluating the present kinetics data. There are two ways to generate kinetics parameters required by RMG libraries: by experimental investigation and by quantum calculation. Either way requires a lot of careful work and research time. A small group of people cannot fulfill this goal.
easily and quickly, and it needs the cooperation from many groups. Those groups should share their latest results in a quick and standardized way on how to format data, how to describe data quality and uncertainty, and how to correctly use data. If we could get a widely agreed kinetics data set to feed in RMG kinetics library, RMG would be able to perform better.

There is another important issue associated with thermodynamics consistency in RMG. In chapter 6, the author has introduced that, in order to maintain the thermodynamic consistency between the forward and backward reaction, RMG fixed one direction as the forward reaction, and the reverse reaction kinetics will be calculated from the independent forward rates and the thermodynamics equilibrium constants. This approach works well for 90% of the reaction families. However, for the reaction families whose forward and backward reactions are of the same type, for example, H abstraction and H migration, RMG currently uses the thermodynamic equilibrium constant to decide the reaction direction, i.e., the direction with Keq>1, will be the forward direction. This method clarifies a lot of confusions; however, it could cause some trouble in picking up the best kinetics. For example, for a very important reaction in n-butane combustion HO₂ + n-C₄H₁₀ -> H₂O₂ + n-C₄H₉, Keq indicates that this is the backward reaction, and to calculate its kinetics, we should use the reaction H₂O₂ + n-C₄H₉ -> HO₂ + n-C₄H₁₀ kinetics rate and Keq to estimate the kinetics rate for the forward reaction path. Ironically, in our present kinetics database, we have collected very good kinetics rate rules for HO₂ abstraction hydrogen from alkane molecule, but we don’t have a good value for the opposite direction. This omission led RMG to choose very approximate kinetics for n-C₄H₅ abstracting hydrogen from H₂O₂ and Keq to calculate the kinetics for HO₂ + n-C₄H₁₀ -> H₂O₂ + n-C₄H₉ reaction, which caused a large deviation (4 order of magnitude) for this reaction pathway. Since this is a major pathway burning n-butane, it caused a lot of trouble at the beginning of generating an n-butane oxidation model by RMG. To solve this trouble and generate the results shown in chapter 8, the author put this important reaction path into our reaction library with the correct kinetics parameters, instead of using the rate based on the estimate of the reverse direction. However, similar problems likely occur for other “symmetric” reaction families, so a systematic way to
handle this issue should be included in RMG. One possible solution is to search kinetics rate rules for both directions, and choose the direction with the kinetics of the better quality as the forward direction. However, this will require more systematic attention to the qualities and uncertainties associated with different kinetic parameters in the databases.

10.2.4 Rate-based pressure dependent model generation algorithm

In RMG, a rate-based iterative model generation algorithm with pressure dependent network exploration has been implemented. The application of those algorithms to a n-butane oxidation case has been presented, and the result shows a good qualitative agreement between the RMG prediction and experimental results.

However, there are some potential limitations of using this rate-based algorithm. This algorithm strongly depends on the qualities of kinetics libraries, since it uses the formation rates of the products or the pressure dependent networks as the standard to decide which is the next species or pressure dependent network to explore. Sometimes a little change of one reaction rate rule of a single sensitive pathway may result in a totally different reaction mechanism. Therefore, we need to develop a tool to identify the highly sensitive reaction pathways, so that we could study more on them to get better rate rules.

In rate-based iterative model generation algorithm, there is an important parameter, error tolerance, that will greatly affect the model generation process. We believe that, if we set the error tolerance small enough, all the important species and pressure dependent networks should be completely explored, though possibly in different order. However, to decide how small an error tolerance is good enough is a hard question, and we don’t have a certain answer. (For purely linear kinetics, the relation between this tolerance and the overall accuracy of the model was derived by Matheu.[2001], but no similar relation is known for nonlinear systems.) To make this algorithm widely accepted by chemistry users, we should provide a correlation between the reaction model’s quality and the error tolerance to guide users to correctly choose a good error tolerance at the beginning of model generation.
10.2.5 Valid range analysis

In this work, the author also developed a practical algorithm analyzing the valid parameter range for a reaction kinetics model based on the rate-based termination rule provided by Susnow and coworkers [Susnow et al., 1997]. This algorithm made use of the first-order sensitivity coefficient at a single parameter point to predict the system dynamic properties in a larger parameter range; therefore, it is a linear approximation of the reaction system properties at the local reaction conditions. Therefore, if users apply this algorithm over a large range of reaction conditions, they should carefully check the validity of the linear approximation. This algorithm can be also improved by using it in an iterative manner, where error introduced by the linear approximation at each step is tested and corrected, so that the linear approximation’s effects can be eliminated by the iterative approach.

The range analysis algorithm has not yet been implemented for pressure dependent reaction modeling process. Some approximation methods for numerically calculating \( \frac{dk}{dT} \) and \( \frac{dk}{dP} \) would make this sort of range analysis more feasible.

10.2.6 System analysis tool

RMG now only provides an ODE/SA solver to simulate the dynamics of a homogeneous reaction system at a constant temperature and pressure, and it also provides the first-order sensitivity analysis, based on which we could analyze the valid parameter range for a generated reaction mechanism. RMG is currently not able to do more complex simulations; however, there is space in RMG design for people to later add more complicated features, such as varied temperature and pressure reaction system simulation, other types of dynamic simulation tools, etc. Some other important analysis algorithms, such as reaction model reduction, and CFD simulation tools should be also added into RMG to improve RMG’s ability to simulate more complicated systems.

10.2.7 Graphic users interface

So far, all the functional groups used in defining reaction templates and thermodynamic and kinetic trees are defined as adjacency lists in text files. A convenient
visualization tool to draw those functional groups and hierarchy trees into pictures, so that users can read and understand those definitions more clearly, would be very helpful. Reading the original adjacency list of a species or a functional group is sometimes bothersome, and it takes much longer for people to understand the adjacency list of a graph than a visualized graph. Furthermore, good graphs are required so that we can nicely present our reaction template and thermo and kinetics trees to the whole chemistry society to request suggestions and comments. If we don’t have a handy visualization tool, all those data cannot be presented to the people outside of our group meaningfully. Therefore, we urgently need a graphical user interface, GUI, to visualize our reaction template definition and thermo and kinetics trees structures.

Another usage of GUI in RMG will be in the model generation process. Now, RMG takes a text format input file to initialize reaction generation, and it also outputs a text format output file for the final generated models. A GUI for RMG software can take care of this procedure using a standard windows graphic application interface, which can read data directly from and display results on the windows. Those GUI will bring a friendly user interface between chemists and RMG.

Developing GUI for RMG is quite straightforward, and people can use present RMG as the internal library and simply add a standard GUI using Java pre-defined GUI development packages, java.AWT and java.swing [J2SE documentation].

10.3 Final Conclusion

In this work, RMG, the second-generation reaction modeling software has been successfully developed for building large reaction mechanisms automatically. New features for building data-model driven reaction templates and for constructing hierarchical structured thermodynamics and kinetics databases are introduced for the first time by RMG to implement the chemistry flexibility and extendibility. Thermodynamics properties estimator and rate-based reaction model generator have been successfully implemented in RMG to automatically construct good quality reaction mechanisms that can correctly predict reaction system dynamics. Applications of RMG to n-butane low
temperature oxidation system have shown good qualitative agreement between the RMG prediction and experimental data. On the other hand, RMG was designed and developed using unified modeling language and object-oriented methodology, which provided a great help on developing RMG into a well-structured software with good reusability and extendibility.
References


313


