Computers and Chemical Engineering 64 (2014) 124–137

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

Language-oriented rule-based reaction network generation and analysis: Algorithms of RING

Srinivas Rangarajan[a], Ted Kaminski[b], Eric Van Wyk[b], Aditya Bhan[a,*], Prodromos Daoutidis[a,*]

[a] Department of Chemical Engineering & Materials Science, University of Minnesota, Minneapolis, MN, USA
[b] Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA

ARTICLE INFO

Article history:
Received 25 November 2013
Received in revised form 2 February 2014
Accepted 12 February 2014
Available online 20 February 2014

Keywords:
Automated reaction network generation
Isomer lumping
Reaction network analysis
Kinetic modeling
Domain-specific languages
Extensible languages

ABSTRACT

The underlying algorithms of the language interface and post-generation analysis modules in RING, a network generation and analysis tool, are discussed. The front-end is a domain-specific reaction language developed with Silver, a meta-language based on attribute grammars. The language compiler translates user inputs written as a program into internal instructions, catches syntactic and semantic errors, and performs domain-specific optimization to speed up execution. In addition to generating reaction networks, RING allows post-processing analysis options to: (a) obtain reaction pathways and overall mechanisms from initial reactants to desired products using graph traversal algorithms, (b) group together isomers to reduce the size of the network through a novel molecule hashing technique, (c) calculate thermochemical quantities through semi-empirical methods such as group additivity, and (d) formulate and solve kinetic models of the entire or lumped complex network based on a rule-based kinetics specification scheme.

1. Introduction

Chemical and biochemical systems are typically composed of a large number (several hundreds or thousands) of species interrelated through a complex network of reactions. Computational tools and techniques are therefore necessary to construct, model, or elucidate the transformations occurring in such networks (Broadbelt & Pfaendtner, 2005). To this end, automated network generators have commonly been applied for network construction and subsequent kinetic modeling (NETGEN (Broadbelt, Stark, & Klein, 1994), RDL (Prickett & Mavrovouniotis, 1997a,b), RMG (Song, 2004), RDL++(Hsu et al., 2008), Genesys (Vandewiele, Geem, Reyniers, & Marin, 2012), and COMGEN (Ratkiewicz & Truong, 2003)). Network construction, however, is also the starting point to qualitatively analyze the constituents of the network – the species and reactions – through identification of synthetic/degradation routes (Finley, Broadbelt, & Hatzimanikatis, 2009; Gonzalez-Lergier, Broadbelt, & Hatzimanikatis, 2005), metabolic flux analysis (Henry, Broadbelt, & Hatzimanikatis, 2007), mechanism identification (Fan, Bertok, & Friedler, 2002; Lin, Fan, Shafie, Bertok, & Friedler, 2009), deducing possible functionality of specific sites (Kummel, Panke, & Heinemann, 2006), etc.

We have developed Rule Input Network Generator (RING) (Rangarajan, Bhan, & Daoutidis, 2012c), a network generation and analysis tool that allows for both quantitative kinetic analysis and qualitative topological analysis of complex reaction networks. Specifically, RING exhaustively enumerates the reaction network consistent with the user specified initial reactants and reaction rules and provides options for analysis and reduction of the reaction network (see Section 2.2). RING can handle a variety of chemistries, including gas-phase free-radical and homogeneous/heterogeneous catalytic chemistries (Rangarajan, Bhan, & Daoutidis, 2010).

The central feature of RING – a reaction network generator – has already been discussed earlier (Rangarajan et al., 2010). In this article, we discuss in detail the underlying algorithms and techniques for the other modules in RING. In Section 3, the application of extensible domain-specific language (DSL) tools in developing a language user-interface for RING is discussed. Section 4.1 lays out the algorithm for identifying pathways to specific products in networks generated by RING while Section 4.2 discusses how RING identifies direct and complete mechanisms, or reaction cycles. The algorithm for lumping, or grouping together, of molecules based on constituent functional groups is presented in Section 4.3, and its extension to accommodate thermochemistry estimation using group contributions is discussed in section 5. The kinetic modeling

* Corresponding authors.
E-mail addresses: abhan@umn.edu (A. Bhan), daout001@umn.edu (P. Daoutidis).

http://dx.doi.org/10.1016/j.compchemeng.2014.02.007
0098-1354/

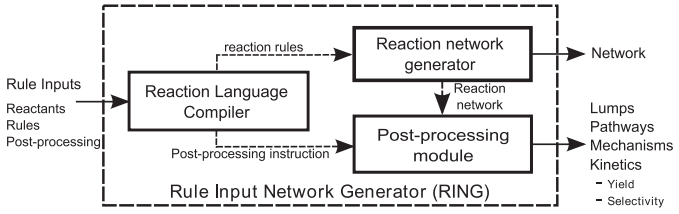


Fig. 1. Overall structure of RING.
Adapted with permission from Rangarajan et al. (2012c) Copyright© 2012 Elsevier Inc.

feature in RING is presented in Section 6. We begin our discussion with a brief description of RING and its network generation capabilities.

2. Overview of RING

Fig. 1 shows the overall structure of RING. Inputs into RING, written in a domain-specific reaction language, can be classified into three classes. First, the initial reactants of the system are written in a modified (Rangarajan et al., 2012c) SMILES (Weininger, 1988) format along with global molecular constraints, such as size and charge restrictions, that need to be satisfied at all times. Second, the reaction rules that describe the chemistry of the reaction system have to be specified either as elementary or as single-step overall reaction rules. The language compiler translates these inputs into internal instructions in the form of C++functions that call other implemented functions for generating an exhaustive reaction network consistent with the initial reactants and reaction rules. The output of the network generator module is a list of species and reactions pertaining to the network.

2.1. Network generation procedure

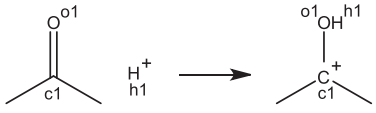
The initial reactants and reaction rules are converted internally into molecular graphs and graph transformation rules, respectively. Molecular graphs are composed of atoms as nodes and bonds as edges. Transformation rules involve "reactant patterns" of atoms and bonds in the reaction center and information that describes how properties of these atoms and bonds have to be rewritten. The reactant patterns of the rule are identified on the molecular graphs using subgraph isomorphism (Ullmann, 1976). Network construction is an iterative process wherein graph transformation rules are repeatedly applied to molecular graphs to generate new graphs. The transformation of the reactant graph to the product graph is a reaction and new molecular graphs, representing new species, are in turn subject to graph transformation rules. The procedure of graph transformation involves identifying molecular subgraphs corresponding to reactant patterns in reactant graphs through pattern matching and rewriting the properties of the nodes (atoms) and edges (bonds) of the reactant graphs. Casting network generation as a graph transformation problem makes it independent of the chemistry per se, thereby allowing for generating any chemical reaction network. The process for network generation continues until no new molecule is encountered; however, being combinatorial in nature this could lead to a very large number of reactions and species being generated. RING, therefore, provides rule constraints and rank-based termination features (Rangarajan et al., 2010). RING also extensively adopts cheminformatics algorithms to represent molecules as unique SMILES and patterns as SMARTS strings, and to identify symmetry, rings, aromaticity, etc. in a molecule. A more detailed description of the network generation process is in Rangarajan et al. (2010). A summary of RING's network generation algorithm is included in the supporting information.

2.2. Network analysis with RING

Network generation, in itself, is the first step of complex network analysis. A third category of inputs – a set of post-processing instructions – to qualitatively and quantitatively analyze the network can be specified within RING. These inputs include: (a) network queries in terms of identifying specific reactions, species, pathways, and mechanisms, (b) instructions on lumping isomers, (c) thermochemistry specification in the form of group additivity methods, and (d) kinetic parameters for each chemistry rule in a conditional rule-based approach and reactor conditions. Pathways are output as a set of reactions connecting queried product molecules from initial reactants, while mechanism queries enumerate supersets of pathways such that the net transformation has no intermediates involved in the overall reaction. Querying for specific reactions and molecules leads to a desired list of these network components, while lumping (or grouping) of isomers results in lists of species lumps and lumped reactions. Providing group additivity rules allows for calculating species and reaction enthalpy, entropy, and free energy. This could, then, be used in conjunction with the querying features to identify energetically feasible pathways and mechanisms. When kinetic and reactor parameters are provided and thermochemistry estimation rules are made available, the kinetic modeling feature outputs yields of different species, overall conversion, sensitivities, and degree of rate control (Campbell, 1994). RING is available open source (RING, 2013) under the GNU Lesser GPL v2.1.

3. Reaction language and compiler

A language interface for network generation was first proposed and introduced by Prickett and Mavrovouniotis (1997a). Hsu et al. (2008) expanded this language to include features specifically meant for heterogeneous catalysis. These are domain specific languages, or DSLs, and are custom languages developed specifically for describing reaction rules. Domain specific languages (DSLs) have several advantages (van Deursen, Klint, & Visser, 2000): (a) they allow for using high level notations as specifications that are well known to the domain expert, (b) programs are concise and "self-documenting", and (c) domain knowledge-based validation and optimizations are possible. The reaction language in RING offers these advantages as well; specifically, the syntax is composed entirely of chemistry parlance making it easier to understand and debug compared to general purpose languages. Fig. 2 shows a sample reaction rule input to RING – protonation of a carbonyl




```
1. rule KetoProtonation {
2.  neutral reactant keto {
3.   C labeled c1 { connected to 2 C
                    with single bond }
4.   O labeled o1 double bond to c1}
5.  positive reactant proton {
6.   H+ labeled h1}
7.  constraints {
8.   keto.size < 7 && keto is linear}
9.  form bond (o1, h1)
10. modify atomtype (c1, C+)
11. modify atomtype (h1, H)}
```

Fig. 2. Example reaction rule – protonation of a keto group.

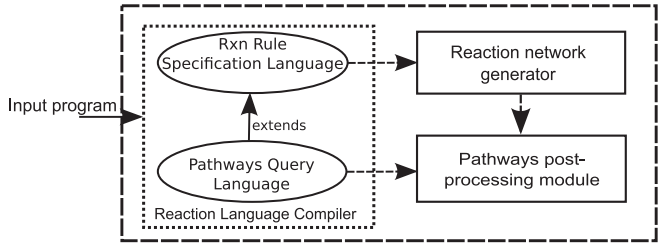


Fig. 3. The overall structure of RING, zoomed in on the compiler component, using the pathways analysis module as an example of a post-processing module.

group. The reaction rule consists of several parts: (a) declaration of the reaction center or reactant patterns (lines 2–6) that define the atoms and bonds participating in the rule, (b) specification of atom (curly brackets in line 3) and molecular constraints (lines 7 and 8) describing restrictions on the specific atoms in the reaction center or the entire reactant respectively, and (c) a list of transformations (lines 9–11) describing changes in the electronic configuration of the atoms or the order (single, double, etc.) of bonds

High-level notations used in the language (Fig. 2) such as "rule", "neutral", "reactant", "single bond", "positive", etc. are derived from common chemistry terminology. Further, the structure of a reaction rule specification scheme – description of reaction center, stipulation of constraints, and description of transformations – closely mimics a chemist's description of the reaction rule. These two factors, thereby, allow for a one-to-one correspondence between how the user would perceive the reaction rule and write it down in the reaction language.

Fig. 3 zooms in on the compiler in the overall structure of RING. The core of the reaction language of RING – reaction rule specification language – is an *extensible language* that focuses solely on describing the reaction rules of interest. These specifications are then transformed by the compiler into C++code that makes use of the reaction network generator library. RING is equipped with a number of extensions, specifically those for specifying post-processing features such as pathways and mechanisms queries. Each new feature or module in RING can contribute a *language extension* to the compiler, usually with additional syntax for describing any specific additional inputs required. The compiler translates these instructions into appropriate C++code that can then be compiled with RING's core C++implementation.

The core of the reaction language of RING bears some similarity to RDL++(Hsu et al., 2008), although the superficial syntax is quite different, as RING is not based on *S-expressions*. There are, however, some notable differences from that system. First, inputs into RING are compiled to C++, whereas RDL interprets them. This allows the RING compiler to perform many "optimizations" on the rules given by the user. This can potentially be a significant advantage in the specification of molecule constraints because these are directly translated to C++Boolean functions that are optimized to fail fast (see Section 3.1). Second, while both RING and RDL++ perform name-binding and basic type checking (for example, where a molecule is expected, ensuring that a name refers to a molecule and not a bond), RING further ensures that the transformations described are appropriate at compile time, checking for basic chemistry mistakes like valency violations. For example, in the sample rule in Fig. 2, if either of the two `modify atomtype` statements are missing, the reaction rule would not conserve charge, and the compiler would generate an appropriate error. Further, if `double bond to c1` was erroneously written as `double bond to c2` or `aromatic bond to c1`, the compiler would generate errors stating that `c2` is not a previously defined label and that aromatic bonds can only exist between two aromatic atoms.

The extended Backus–Naur form (EBNF) of the grammar of the language is given in the supporting information. RING's distribution includes a complete manual with all the syntax. Interested readers can access the documentation available online (RING, 2013).

3.1. Compiler optimizations

Three broad categories of optimizations are performed by the compiler to enhance the speed of execution. We discuss these categories first and then provide illustrative statistics to demonstrate the efficacy of these optimizations.

3.1.1. Constraint categorization

The network generation algorithm needs to sift through candidate molecules for each of the reactants in a bimolecular reaction rule to identify potential co-reactants. Each possible pair of molecules must be considered for each rule. If the pair satisfies the constraints, a check for the presence of the relevant reaction patterns in the reactants can be done and subsequently new reactions generated. The constraints in a rule can be categorized as depending on one or the other reactant individually, and those "combined" constraints that unavoidably depend on both reactants. Ordering the constraints such that individual constraints are checked first can potentially speed up the network generation process. For example, if a molecule does not satisfy the constraints pertaining to the first reactant in the rule, then there is no need to check for a potential molecule pair. The RING compiler automatically classifies the constraints into these categories and emits them as separate constraint checking functions for the network generator. These functions are used in the specific order – individual constraints of the first reactant, individual constraints of the second reactant if the first set of constraints are satisfied, and combined constraints if individual reactant constraints are all satisfied. The order of specification of constraints by the user is, thus, immaterial.

3.1.2. Constraint ordering

The order in which either individual or combined constraints are checked may also affect performance. For example, it is faster to check the number of heavy atoms or the charge of a molecule than to check if the molecule has a large functional group, such as acid anhydride. The compiler estimates heuristically the cost of checking each constraint, and orders them to attempt to verify those constraints that are quick and easy to check before those that are slower. For example, checking for molecule size or charge precedes any checks for molecular fragments.

3.1.3. Pattern re-ordering

Checking constraints that involve identification of specific functional groups in a molecule and detecting reactant patterns require pattern matching of a fragment in a molecule. In such cases, the particular arrangement of atoms in the reactant patterns or functional groups can speed up the matching process. The compiler re-orders these patterns before presenting it to the network generator to try to fail to match as early as possible. Unlike constraints re-ordering, where there is a cost associated with checking each constraint, here the cost for each atom is roughly the same. Instead, the atoms and bonds are ordered roughly by likelihood that they occur at all in molecules. Most organic molecules derived from biomass/petroleum sources have more carbon atoms than oxygen atoms. Consider a six carbon ring and a pattern with two carbons and an oxygen, such as the fragment "C–C–O". There are 12 different ways the two carbons could match this molecule before failing due to the lack of an oxygen, but matching the oxygen first would fail immediately. Placing rarer atoms first can thus make the matching fail early and thus speed up the pattern matching process. RING applies several empirical heuristics for the likelihood of occurrence,

Table 1
Benchmarking statistics: run-time ratios for successive compiler optimizations.

Benchmarks	Reactions	Species	Run-time ratio	
			None→Constr. Cat[a]	Constr. Cat→All
Base catalysis	12,771	4609	1.02 ± 0.08	1.00 ± 0.01
Fructose-to-HMF	1223	546	1.09 ± 0.02	0.99 ± 0.02
Glucose pyrolysis	14,375	3131	1.00 ± 0.01	1.00 ± 0.01
HMF→Levulinic acid	39,844	14,875	58.30 ± 0.59	1.00 ± 0.01
Propane aromatization	2031	594	1.41 ± 0.03	1.21 ± 0.00

[a] Constraint categorization optimization.

such as: (a) nitrogen, sulfur, and phosphorous atoms are rarer than oxygen atoms, which in turn are rare compared to carbon atoms, (b) charged atoms occur less frequently than neutral atoms in a network, and (c) stronger bonds (double and triple) are rarer than single bonds.

Table 1 lists the run-time ratios upon enabling: (a) only the constraint categorization optimization compared to having no optimizations (None→Constr. Cat), and (b) all the optimizations with respect to having constraint categorization alone (Constr. Cat→All) for five systems in a benchmarking study. The systems included in the study are: (a) the synthesis network for forming longer chain alcohols from smaller (C_1-C_2) oxygenates using base catalyzed carbon–carbon bond formation and metal catalyzed (de)hydrogenation chemistries, (b) acid catalyzed conversion of fructose to 5-hydroxymethylfurfural (HMF), (c) pyrolysis of glucose by neutral electrocyclic reaction steps, (d) conversion of HMF to levulinic acid in acidic medium, and (e) acid catalyzed aromatization of propane. The first system (base catalysis) consists of single-step non-elementary overall reaction rules, while the other systems are modeled in terms of elementary steps. The reaction rules are given in the supporting information. It can be noted that while optimizations do not lead to statistically significant speed-up in the first three cases, there is considerable savings in the run times for the last two systems – HMF-to-levulinic acid and propane aromatization. Specifically, the network generation time for the HMF-to-levulinic acid system speeds up by a factor of 60 upon enabling constraint categorization alone. This is attributable to the nature of constraints imposed in some of the reaction rules of this system wherein one of the reactants is restricted to being an oxygenate. Checking for this constraint early prevents RING from performing several unnecessary steps before eventually rejecting a molecule because it does not satisfy that constraint. This system, however, shows no further noticeable improvement upon subsequently enabling the other two optimizations. On the other hand, the propane aromatization system shows significant speed-up due to constraints categorization alone as well as all upon enabling all the optimizations. The additional improvement in the latter case is possibly due to the constraint ordering optimization because this system contains a greater number of constraints relative to the other systems. Pattern re-ordering, although not explicitly resulting in any statistically significant improvement in these five systems, can lead to about 15% speed-up (and potentially more) in identifying individual patterns in some cases (see data in supporting information). Thus, the statistics presented in Table 1 suggest that systems with: (a) reaction rules having very restrictive constraints, and (b) large reaction networks (several tens of thousands of reactions) can benefit significantly from compiler optimizations.

3.2. Language extension

The RING compiler and language is capable of supporting independently developed post-processing modules due to its design and implementation in a domain-specific language called Silver (Van Wyk, Bodin, Gao, & Krishnan, 2010) that uses a parser generator called Copper (Van Wyk & Schwerdfeger, 2007). These two tools are designed to support implementing extensible languages. A language extension can add both new syntax and analysis of the existing language. For example, an extension can add new error checks for the sensibility of the reaction rules, or an entirely new syntax intended for a new post-processing option. The difficulty of accomplishing language extension lies in two areas: (a) a full range of extensions have to be possible without dramatically complicating the design of the compiler and (b) different extensions should not conflict so that a working compiler cannot be generated.

Silver is a functional language based on attribute grammars, with a strong composition model for attribute grammars and is used to define the semantics (for error checking), optimization, and translation of RING programs. Thus, the main components of the RING compiler are written as an attribute grammar in Silver, and these make no reference to any extensions (post-processing options). Instead, Silver is simply able to take the extension grammars it is provided with and automatically compose them with the host language grammar, producing a working compiler with all the requested pieces combined. Host languages and extensions for Java (Van Wyk, Krishnan, Schwerdfeger, & Bodin, 2007), C, Promela (Mali & Van Wyk, 2011), and Modelica have been written in Silver, and in fact the Silver compiler is also written in Silver. These capabilities allow for post-processing modules to fully integrate with the RING compiler.

Copper is a parser and context-aware scanner generator that is used to define the concrete (or surface) syntax of RING. Context-aware scanning solves a number of problems in composing language extensions, one being the problem with different extensions introducing the same keyword into the language (Van Wyk & Schwerdfeger, 2007). Context-aware scanners return only tokens that would not cause a syntax error and thus if two extensions introduce the same keywords but they appear in different contexts, the scanner will always return the appropriate one. This makes it somewhat easier to write the context-free grammars that define language syntax and to ensure that they exists in the LALR(1) subclass of context-free grammars (Aho, Sethi, & Ullman, 1986) that are supported by Copper.

The formalisms of attribute grammars, used by Silver, and context-free grammars, used by Copper, naturally compose and thus it is rather straightforward to take a host language specification and collection of language extension specifications to create the specification for a customized extended language. However, these compositions are not always well-defined, in the case of attribute grammars, or in the LALR(1) class, in the case of context free grammars. Both Silver and Copper have *modular* analyses that can be used by the language extension designer to check if their extension is of the form that it will later compose with other extensions that also pass this analysis. These analyses effectively *verify* an extension as one that is composable. A programmer that selects only verified extensions is assured that the composition of the host language and his or her selection of extensions will be well-defined.

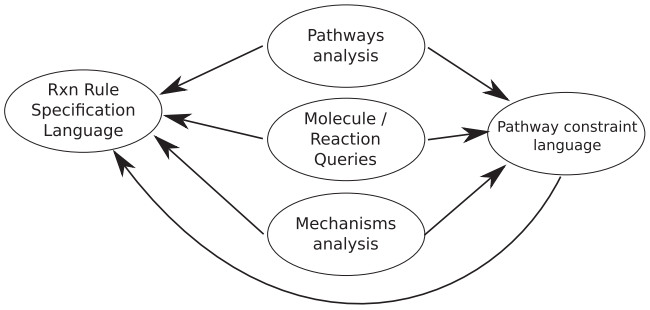


Fig. 4. The host language, and a representative subset of extensions to it, implemented in Silver. The arrows represent dependencies between grammars.

```
Find pathways to mol{
 mol is "CC(=O)C"
 } constraints {
  length < 6
  contains <=2 rule "HydrideShift"
  eliminate similar pathways
} store in "HMFPathways.txt"
```

Fig. 5. Sample post-processing instruction written in the language: a query to obtain pathways to acetone.

For further details on these analysis we refer the reader to the relevant papers (Kaminski & Van Wyk, 2012; Schwerdfeger & Van Wyk, 2009).

The RING compiler comes with a number of post-processing extensions already, the organization of a representative part of which can be seen in Fig. 4. The language extension, "pathway constraint language", adds the syntax for expression pathway constraints to the host language (reaction rule specification language). Each post-processing analysis option further has a corresponding language extension that depends on the host language and the pathway constraint language extension, and adds a specific type of analysis expressible in the extended language. For example, "pathways analysis" language allows for expressing pathway queries in addition to specifying the reaction rules of a system. Each of these extensions is an independent (only those dependencies shown) grammar, and the full compiler is built by having Silver compose them all together.

4. Post-processing

Instructions given subsequent to reaction rule description for identifying network information such as isomer lumps, specific reactions and molecules, and pathways and mechanisms, constitute post-processing options in RING. In this section, we discuss in detail the algorithms of these options.

4.1. Pathway identification

The pathway identification algorithm in RING finds all pathways between the initial reactants and specified products of the generated network. Fig. 5 shows a sample query for pathways to acetone of length less than six steps and not having more than two instances of the rule "HydrideShift" (assuming such a rule was specified earlier). The algorithm makes use of two sets of information identified and stored during network generation. First, RING keeps track of the rank of each species in the network – the minimum number of steps required for that molecule to form from any of the initial reactants. Second, the "closer" parent of each product molecule in a reaction is identified during network generation in a three-step process. The reactant with a larger rank (that is, the one further from the initial reactants) is the de facto closer parent; if all reactants have the same rank, however, the one that contributes most number of atoms to the product becomes its closer parent. In case of a tie, the first reactant is assumed to be the closer parent. The ranks and parent information are used in a reverse depth-first search starting from the product and traversing backwards along the network to reach the initial reactants. The details of the algorithm are given in Algorithm 1.

Algorithm 1. FindPathways (Molecule M_0, integer MaxPathLength, PathwayConstraints P_C)

PathwayStack ← {} Stack of reactions constituting a pathway
AllPathways ← {} List of pathways
MoleculeStack ← M_0 Stack of molecules traversed
while ! *MoleculeStack*.empty() **do**
  if *MoleculeStack*.top() is an initial reactant **then**
    if *MoleculeStack* has no repeating entry **then**
      AllPathways.push (*PathwayStack*)
      PathwayStack.pop()
      MoleculeStack.pop()
  else
    find new Reaction, R, that forms the Molecule *MoleculeStack*.top()
    if found & & *PathwayStack*.size() <MaxPathLength **then**
      find closer parent Molecule, P, of *MoleculeStack*.top() in R
      if P ∉ *MoleculeStack* & & *PathwayStack*.size() +Rank(P) ≤
  MaxPathLength **then**
        PathwayStack.push(R)
        MoleculeStack.push(P)
    else
      PathwayStack.pop()
      MoleculeStack.pop()
return AllPathways satisfying constraints P_C

The inputs of the algorithm are the target molecule, the maximum path length desired and additional pathway constraints (Fig. 5), while the output is a list of pathways, each of which, in turn, is a set of reactions. Since the rank of each molecule in the network is known through network generation, the shortest pathway to M_0 is already known. Therefore, for the algorithm to proceed, M_0 must satisfy the condition: Rank(M_0) ≤ MaxPathLength. It is better to traverse backwards from the product to the initial reactant because the parent molecule of each product of a reaction is known. During the process of traversal, a stack of intermediate molecules, *MoleculeStack*, is maintained. The first, and hence the bottom, element of the stack is M_0. The algorithm also maintains a stack of reactions that constitutes the pathway. At each step of the traversal, a reaction that produces the molecule at the top of *MoleculeStack* is found and the parent, 'P', of this molecule is chosen as the next intermediate molecule if P's rank permits that a pathway can be formed in MaxPathLength steps or less. The reaction is then added into the *PathwayStack*. Furthermore, at each step, the next molecule is chosen such that it is not already present in *MoleculeStack*. This ensures that cycles are avoided and only acyclic paths are returned by the algorithm. If, ultimately, a parent is reached within MaxPathLength steps such that it is one of the initial reactants, a pathway is deemed found, and the stack of reactions in *PathwayStack* is added to a list of pathways, *AllPathways*. The algorithm then backtracks to the previous molecule in *MoleculeStack* and proceeds as above. The algorithm terminates when *MoleculeStack* ultimately turns empty. The pathways, thus found, do not consider the co-reactants and products of a reaction and the relevant intermediate molecules of a pathway are determined by the parent–daughter relationship of each reaction. As shown in Fig. 5, the user can specify constraints to describe the target molecule M_0 and subsequently provide pathway constraints. All pathways obtained are subsequently checked to ensure they satisfy other pathway constraints. These pathway constraints include: (a) the number of occurrences of specific rules, (b) bounds on the occurrence of specific molecules as reactants or

products in the entire pathway or in a reaction belonging to a particular reaction rule in the pathway, and (c) upper bounds on the activation barrier (if available or can be calculated for individual reactions).

Sometimes, several pathways may be found between an initial reactant and a specified product that have the same length, and the same number of reactions of each rule type. Such pathways differ only in the order of reaction within the pathway with intermediates of one pathway being isomers of that of another. Such pathways can be grouped together as a single pathway in RING by using the `eliminate similar pathways` command, as shown in Fig. 5.

4.1.1. Comparison with other pathway finding algorithms

Pathways identification has extensive applications in biological network analysis and biological network reconstruction as it allows for understanding various complex biochemical processes such as metabolism. These pathway finding algorithms identify k-shortest paths (Eppenstein, 1998) between source and target compounds or reactions of a directed weighted/unweighted network with or without tracking the destination of each of the atoms (Croes, Couche, Wodak, & van Helden, 2006; Heath, Bennett, & Kavraki, 2010; Jeong, Tombor, Albert, Oltvai, & Barabasi, 2000). These algorithms start with a given set of reactions typically constructed from databases. To assess the performance of our algorithm, we compare our algorithm with these state-of-the-art generic pathway finding algorithms. The comparisons are only qualitative because these algorithms cater to any assembled network of reactions while our algorithm is specific to networks generated from an initial set of reactants by successive application of reaction rules; our algorithms are therefore optimized for the specific case of analyzing automatically generated reaction networks.

The algorithm in RING differs from these algorithms in several ways. First, because the network was constructed from initial reactants and reaction rules, it is safe to assume that each species in the network can be traced back to the initial reactant in at least as many steps as the rank of the species. This assumption, however, does not hold in the algorithms discussed above. Second, in RING, backtracking along any path from the products will ultimately lead to the initial reactants; this again does not hold in generic pathway identification algorithms. Indeed, in those algorithms, forward or reverse search of the network from the reactants or products respectively will have similar performance. Third, our pathway algorithm can, in effect, track atoms along the pathway because the information of the parent reactants for each reaction is available. For the generic pathway identification algorithms, tracking the destination of the atoms of a reactant in the network requires identification of atom mapping between the reactants and products.

4.2. Mechanism enumeration

In addition to pathways, RING can identify *direct* and *complete* mechanisms. Direct mechanisms represent reaction cycles containing a set of reactions such that the overall reaction has no reactive intermediates. Complete mechanisms represent, on the other hand, a set of direct mechanisms (or reaction cycles) that describe the complete transformation from the initial reactants to any products. Complete mechanisms are supersets of reaction pathways because they contain the additional information of reactions leading from/to co-reactants/co-products. We also note that both direct and complete mechanisms refer only to a set of reactions and their stoichiometry and not to their kinetics or thermochemistry, although these can be calculated if relevant information is available. Further, at this stage, we do not even consider possible rate-determining steps.

Algorithm 2 describes the procedure for finding direct mechanisms. The procedure is similar to that of finding pathways – a reverse depth-first search is employed with stacks for reactions and molecules that get populated along the traversal. At each step of the traversal, a reaction 'R' is chosen, like in pathways. However, 'R' is chosen on the basis of the stoichiometric coefficient of the current intermediate – top molecule of *MoleculeStack* – in the overall reaction, O_R, of all the reactions in *ReactionStack*. Therefore, this reaction could form/consume the intermediate as appropriate. Further, the algorithm ensures that adding the new reaction will not lead to closed cycles that are net-zero in overall stoichiometry. The intermediate chosen for the next step of the traversal is a reactive intermediate in O_R. A direct mechanism is found when no reactive intermediates are explicitly involved in the overall reaction. As the overall reaction is checked each time a reaction is added, this procedure ensures that the mechanism is composed of a minimal set of reactions and, hence, is direct. Further, the stoichiometric coefficient of a reactive intermediate in *ReactionStack* may not just be 1 (or −1) but could be higher or lower. Similarly, the stoichiometric coefficient of the intermediate in 'R' need not be 1 (or −1). In such cases, appropriate stoichiometric numbers, ν_1 and ν_2 in the algorithm, need to be found so that the overall reaction O_R, of 'R' and reactions in *ReactionStack* taken together, does not explicitly involve the reactive intermediate at all. The algorithm terminates when *MoleculeStack* is empty.

Algorithm 2. FindDirectMechanisms (Molecule M_0, integer MaxLength, MechanismConstraints M_C)

ReactionStack ← {} Stack of reactions constituting a direct mechanism
AllDirectMechs ← {} List of direct mechanisms
MoleculeStack ← M_0 Stack of molecules traversed
while ! *MoleculeStack*.empty() **do**
  if *ReactionStack* is a direct mechanism **then**
    AllDirectMechs.push (*ReactionStack*)
    ReactionStack.pop()
    MoleculeStack.pop()
  else
    find new reaction, R, forming/consuming *MoleculeStack*.top()
    if found & & # of unique reactions in *ReactionStack* +1 ≤ MaxLength **then**
      if adding R into the *ReactionStack* does not lead to cycles **then**
        determine the stoichiometric factors, ν_1 and ν_2, of *ReactionStack* and R respectively
        get the overall reaction O_R of $\nu_1 \times$ *ReactionStack* and $\nu_2 \times$ R
        if $O_R \neq \varphi$ **then**
          find reactive intermediate $I \in O_R$
          if found **then**
            ReactionStack.push(R)
            MoleculeStack.push(M_1)
  else
    MechanismStack.pop()
    MoleculeStack.pop()
return AllDirectMechs satisfying M_C

Algorithm 3 lays out the procedure adopted in RING for finding complete mechanisms. This algorithm also involves a reverse depth-first search strategy like in the identification of pathways and direct mechanisms, but at each step of the traversal, direct mechanisms are added instead of individual reactions. Note that direct mechanisms are also referred to as reaction cycles ("Rxn-Cycles", specifically) in the algorithm. The direct mechanisms are calculated on-the-fly when a new intermediate is encountered and stored until the end. This way, direct mechanisms of only the relevant intermediates need to be identified, and further, only once. At each step, the next intermediate is chosen from amongst all the non-initial reactants of the overall reaction O_R. An overall reaction is found when the reactants of O_R are all initial reactants of the given system. In this sense, mechanisms and pathways parallel stoichiometric and path-finding approaches in systems biology (Planes & Beasley, 2008).

Algorithm 3. FindCompleteMechs(Molecule M_0, integer MaxLength, integer MaxRxnCycles, OverallConstraints M_C)

```
MechanismStack ← {} Stack of direct mechanisms constituting a complete
  mechanism
AllMechanisms ← {} List of complete mechanisms
MoleculeStack ← M0 Stack of molecules traversed
while! MoleculeStack.empty() do
  if MechanismStack is a complete mechanism then
    AllMechanisms.push (MechanismStack)
    MechanismStack.pop()
    MoleculeStack.pop()
  else
    find direct mechanism, Dm, forming MoleculeStack.top(), and not
  considered before
    if found & & MechanismStack.size() +Dm.size() ≤ MaxLength then
      determine the stoichiometric factors, ν1 and ν2, of MechanismStack
  and Dm respectively
      get the overall reaction OR of MechanismStack and Dm
      find reactant M1 ∈ OR and ≠ any initial reactant
      if found & & MechanismStack.NumberOfRxnCycles() ≤
  MaxRxnCycles then
        MechanismStack.push(Dm)
        MoleculeStack.push(M1)
    else
      MechanismStack.pop()
      MoleculeStack.pop()
return AllMechs satisfying MC
```

4.2.1. Comparison with other mechanism identification algorithms

Algorithms for the construction of reaction mechanisms have been proposed, notably by Happel and Sellers (1982), Otarod and Happel (1992), Mavrovouniotis and Stephanopolous (1992), and Mavrovouniotis (1992). These algorithms construct *all* possible direct mechanisms from a given set of reactions. The underlying idea is to exhaustively consider all distinct combinations of reactions so that the set of reactions chosen have no net consumption/formation of reactive intermediates and is minimal in size, thus forming a direct mechanism. This is done, for example, in the case of the algorithm by Mavrovouniotis and Stephanopolous (1992), by finding all pairs of reactions for a reactive intermediate – one forming the reactive intermediate and another consuming it – and adding them together to get a new set of reactions without that intermediate. Subsequently, the reactions involving that reactive intermediate are replaced by this new set of reactions so that the intermediate is completely eliminated from the network. This procedure is done iteratively to obtain all direct mechanisms eventually.

Our algorithm differs from these in several ways and, again, we provide only qualitative and descriptive comparison because the scope of the algorithms mentioned above are different from the ones proposed here. First, in RING, mechanisms – direct or overall – are sought for specified target molecules. That is, not all possible mechanisms are sought, only specific mechanisms that satisfy user-specified constraints are identified. This does not require the successive elimination of all intermediates, such as in the algorithm by Mavrovouniotis and Stephanopolous (1992); only those intermediates involved in the synthesis of the target molecules need to be eliminated. Second, algorithms mentioned above find only direct mechanisms; they, however, do not find the overall mechanism from initial reactants to final products. Third, our algorithm identifies empty cycles and eliminates them. The other algorithms mentioned above do not report a cycle detection and elimination method.

Alternative mechanism identification methods involving constrained optimization have also been proposed (Planes & Beasley, 2009). These methods formulate an integer or mixed-integer linear programming problem to select reactions that taken together have no net consumption/production of reactive intermediates. Such algorithms offer the advantage of identifying a linearly independent set of overall mechanisms, for example, as in elementary modes and extreme pathways (Klamt & Stelling, 2003), or identifying alternative solutions (Lee, Phalakornkule, Domach, & Grossmann, 2000). These methods can also be used to identify cycles by setting that both reactants and products be net-zero. Marvin, Rangarajan, and Daoutidis (2013) describe a method for identifying mechanisms that combines network generation using RING with constrained optimization algorithms. RING outputs the network in formats that can be exported into optimization software (specifically GAMS). The advantage of such a method is that in addition to identifying all mechanisms, they can be sorted or ranked according to any user-defined objective.

4.3. Lumping

The size of a complex reaction network can be reduced by lumping, or grouping, isomers. The reduced network can be more amenable to further analysis such as kinetic modeling. In RING, the process for identifying the lumps consists of three steps: (a) collation of molecules with the same number of different types of functional groups into one lump, or functional lumping, (b) assignment of a representative molecule to each lump based on user-input structural criteria for cyclic and acyclic species, and (c) further lumping of paraffins, olefins, naphthenes, hydrocarbon aromatics (PONA), or molecules satisfying user-defined properties based on molecular formula. Each of these steps is considered in detail below.

4.3.1. Molecule collation

The first step in the process, if lumping is sought by the user, is to find and group all molecules that have the same number of each functional group. For example, 2-pentanol and 3-pentanol are lumped together because they both have two carbon atoms belonging to a methyl group (**C**H_3), two methylene carbon atoms (**C**H_2), and one carbon and oxygen of the **C**H**O**H group. Fig. 6(a) shows collation of two groups of functionally equivalent molecules – secondary pentanols and xylenes. On the other hand, 1-pentanol is a separate lump because it has three methylene carbon atoms, one methyl carbon, and one carbon and oxygen atom of the **C**H_2**O**H group. It should be noted that functional groups have a specific combination of atoms and bonding. Thus, two molecules have exactly the same number of each functional group if there exists a mapping between each atom of one molecule and a unique atom in the other molecule. The mapping, in this case, is defined as possible when the two atoms are identical themselves and have identical nearest atoms and bonds. Thus, functional equivalence between two molecules can be established by keeping track of what kinds of atoms are present in the two. To do this, RING adopts a simplified hashing scheme.

Hashing is a technique by which data objects can be referenced by a fixed-length object such as a bit-string. Within the context of molecular databases, Wipke, Krishnan, and Ouchi (1978) and Ihlenfeldt and Gasteiger (1994) used hashing predominantly for molecule indexing in databases and quick retrieval of molecular information. Wipke et al. (1978) proposed and tested different hash functions generated by combining different parts of a canonical molecule name (SEMA) (Wipke & Dyott, 1974) through boolean operations such as XOR, to obtain different-sized (8, 9, or 10) bit-string hash values. These bit-strings, thereby, implicitly contain the structural and stereochemical information of the compound. Ihlenfeldt and Gasteiger (1994), on the other hand, use the complete molecular topology instead of only using the name for hashing. An atom hashing seed is first generated for each atom as the product of prime numbers corresponding to certain seed parameters. Subsequently, the atom hash is generated by combining the seed of each atom with its neighbors followed by

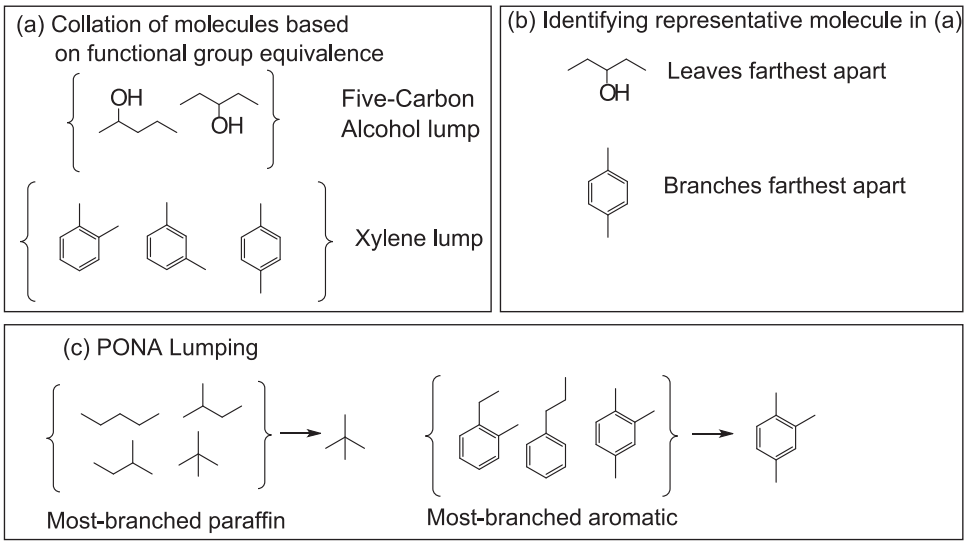


Fig. 6. Three-step lumping process: (a) functional group-based equivalence is used to collate molecules, (b) a representative of each collated lump is identified, and (c) paraffins, olefins, naphthenes, and aromatics (PONA) lumps are further lumped based on molecular formula.

equalization of bit distribution. The molecule hash is then obtained as a combination (logical XORs) of the atom hashes of the constituents. This procedure, thus, directly takes into account every atom and its neighboring environment and bonding in the molecule to generate a hash value. In RING, the hashing scheme exploits the properties of prime numbers and adapts ideas from Ihlenfeldt and Gasteiger (1994) and Wipke et al. (1978), as discussed below. However, the use of molecule hashing in RING is different from that described above. In traditional hashing, hash functions are designed so as to minimize the chances of collision of hash values of different molecules. In contrast, the key requirements for lumping are that: (a) all molecules that are functionally equivalent – i.e., they have the same number of each type of functional group – *must* necessarily have the same hash value, and (b) collision between nonequivalent molecules should be *completely* avoided.

Algorithm 4 describes the algorithm for generating the hash value of molecules. The first step is the evaluation of atom hash seeds for each atom as the product of primes corresponding to various parameters such as the number of nearest non-hydrogen neighbors, element type of the atom (C, N, O, etc.), element type of the neighboring atoms, aromaticity, and bond orders of each bond connected to it (see supporting information). The final value of the seed is the product of each of these factors. For example, the hash seed of the tertiary carbon in 2-butanol is calculated as shown in Fig. 7. The carbon atom has three neighboring atoms, hence the factor 2^3, while the prime corresponding to carbon, prime(C), is cubed because the atom under consideration is carbon and it has two neighboring carbon atoms. Subsequently, each seed is assigned a prime number on-the-fly by RING during reaction network generation. This prime number constitutes the atom hash of that atom. It can be noted that two atoms of the same element having identical

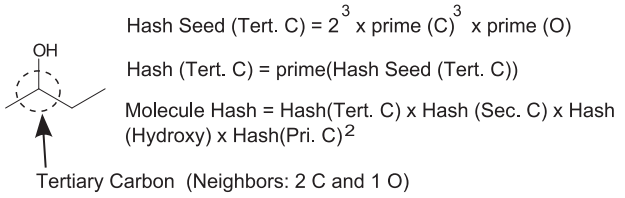


Fig. 7. An example of molecule hash value calculation. The steps are: (1) calculate hash seed of atom, (2) calculate hash of the atom using the hash seed, and (3) calculate molecule hash from the constituent atom hashes.

nearest neighbors have the same atom hash value. Thus, an atom hash value corresponds to a particular class/kind of atom, such as the tertiary carbon (Tert. C) shown in Fig. 7. The molecule hash value is evaluated as the product of the atom hash values of each of the non-hydrogen atoms in the molecule (hydrogen atom hashes are, however, considered for purely hydrogenic species such as H+, H-, H· etc.). More information on the individual functions used in Algorithm 4 is given in supporting information.

Algorithm 4. (Integer, Integer) **HashValue** (Mol)

```
Hash ←1, ElectronicHash ←1
for each non-Hydrogen atom a_i of Mol do
  AtomHash ←1
    n = # nearest neighbors of a_i
    AtomHashSeed = 2^n × ElementPrimeValue(a_i)
    if atom is aromatic then
      AtomHashSeed = AtomHashSeed ×3
      ElectronicHash = ElectronicHash × AtomElectronicHash(a_i)
      for each neighbor, N, of a_i do
        AtomHashSeed = AtomHashSeed × ElementPrimeValue(N)^bondorder
    AtomHash = AtomPrimeValue(AtomHashSeed)
Hash = Hash × AtomHash
return (Hash, ElectronicHash)
```

An additional hash value accounting for the electronic configuration of the atoms of the molecule is also evaluated. For each atom not in its elemental ground-state electronic configuration, the product of its atom hash and a prime number corresponding to the electronic nature of the atom (magnitude of charge and presence/absence of unpaired electrons) is evaluated as an electron hash of the atom. The electronic hash of the molecule is the product of the electron hash values of the non-ground-state atoms. The molecule electronic hash of a neutral and stable molecule, such as 2-butanol, is 1 by definition. The hash value of the molecule is, hence, the integer pair of the molecule hash value and molecule electronic hash.

The atom hash seed, being a product of primes, takes a unique value that can be obtained only by the specific combination of the identity of the atom, its neighbors, and the orders of the included bonds. This implies that the atom hash values, which are assigned prime numbers based on their hash values, are unique. The atom hash implicitly takes into account the associated functional group information the atom is a part of; therefore, the product of atom hashes leads to a hash value that is unique to the set of functional

groups in the molecule. Thus, two molecules having the same set of functional groups will have the same hash values.

The hashing technique for functional equivalence using an integer pair, calculated as products of prime numbers, is atypical when compared to the bit-strings-based hashing functions discussed in Ihlenfeldt and Gasteiger (1994). However, the utility of hash values in RING is similar to that of the bit-string technique as they both provide distinct identification for quick retrieval of molecular information. The hash value is implemented as a pair of unsigned integers in RING; there is, therefore, a finite possibility that hash values become large enough to cause a numerical overflow. The hash values can, in such a case, wrap to result in a different number, and thereby potentially lead to collisions. However, this is resolved by comparing the sizes of the two colliding molecules which will necessarily be different.

4.3.2. Identifying a representative molecule

The representative molecule of functionally equivalent lumps is determined on the basis of user-defined criteria. Specifically: (a) the representative molecule of acyclic species lumps can have leaves (the end atoms of the molecule) closest to (or farthest apart from) each other, and (b) the representative molecule of lumps of cyclic species can have branches farthest apart from (or closest to) each other. Fig. 6(b) shows that 3-pentanol and p-xylene are chosen as the representative molecule based on leaves and, hence, branches being farthest apart. Alternatively, the molecule with branch ends (or leaves) closest to each other could be set as the representative. In such a case, for example, o-xylene will be the lump representative of xylenes.

4.3.3. Additional lumping

The lumps identified through functional lumping can be collapsed further to form fewer groups. At this stage, the lumping is according to the number of heavy and hydrogen atoms in the molecule. These molecule constituents should satisfy certain molecular characteristics set by the user. RING implements the additional lumping by sifting through each lump, determining each time if the lump representative matches the characteristics specified for additional lumping and then accordingly groups together the lumps to get the new set of lumps. There are four pre-specified classes for lumping hydrocarbons – paraffins, olefins, naphthenes, hydrocarbon aromatics, and their reactive intermediates. The user has a choice to represent this lump by the constituent having the most/least number of branches. Fig. 6(c) shows an example where paraffins and aromatics are both lumped to most branched molecules. Alternatively, either or both of them could have been lumped to the least branched of the constituents.

In addition to these pre-specified classes, there can be additional user-defined classes. The user can specify molecular characteristics involving size, shape, and/or presence/absence of a specified frequency of functional groups that describes the desired class of molecules and then further specify the representative nature (most/least branched). For example, surface alkoxide intermediates in solid Brønsted acid catalysis can be lumped using this method. Further, n- and iso-alcohols lumped separately by the functional lumping scheme can be further lumped together using this strategy.

4.3.4. Comparison with other lumping methods

Methods to lump a reaction network fall into two broad categories: (a) mathematical lumping based on kinetics, and (b) chemical functionality-based lumping. Mathematical lumping methods (Coxson & Bischoff, 1987; Kuo & Wei, 1969; Li & Rabitz, 1989) identify a lumping matrix "M" that groups one or more species so that the resultant model results are as close to the original. The constituents of such groups may or may not have physical meaning (Ho, 2008). Chemical functionality-based lumping methods, on the other hand, group together molecules that have similar set of functional groups. These lumps have similar chemical and physical properties in addition to being related through several reactions in the network. RING's lumping scheme falls under this second category and we compare our algorithm with other similar methods.

The vector-based representation of structure-oriented lumping (SOL) (Quann & Jaffe, 1992) offers a natural framework for lumping structural isomers that have the same set and number of different functional groups but have a different order or position of these groups in the molecule. However, molecules can only be represented as lumps and it is not always possible to get the structure of the individual molecules that constitute the lump from the vector. Further, the vector that stores the structural representation is fixed and pre-assigned. For each new chemistry, therefore, the internal representation vector must be expanded to appropriately account for new functional groups. RING offers the feature of SOL – functional groups-based lumping – through the functional lumping feature but is more generic as it is chemistry-independent and identifies and tracks functional groups dynamically. The software RDL++(Hsu et al., 2008) performs lumping of hydrocarbon isomers on the basis of molecular formula as a post-processing step. This method is not applicable for lumping oxygenates because even linear molecules with different functional groups can have the same molecular formula; for example, dimethyl ether and ethanol are both of the form C_2H_6O, even though they have different functional groups. The lumping technique in RING can distinguish these two molecules; however, provisions also exist to allow lumping of molecules based on the molecular formula for user-specified categories of molecules (such as paraffins, naphthenics, surface alkoxide intermediates, etc.).

4.3.5. Lumped network

Once molecules are grouped together to their respective lumps, the reactions of the network can be lumped as well, on the basis of the lumps of the reactants and the products. A lumped reaction, then, is one wherein the reactants and products are represented by their respective lumps. A lumped network, consequently, is the network of lumped reactions. Several reactions, of a given reaction rule, can collapse to the same lumped reaction. The size of the lumped network, therefore, is significantly smaller compared to that of the parent network. This "reduced" network, in principle, contains distinct reactions of the different groups of molecules of the network.

4.4. Molecule and reaction queries

RING also allows querying for molecules and reactions. Such queries are implicitly part of the pathways and mechanism identification – for example, a query for a molecule is the first step (Fig. 5) – but can also be sought independently. Molecule queries are input in the form of seeking all molecules that satisfy specified molecular constraints. Reaction queries, on the other hand, are input as reaction constraints by specifying the rule, reactant, product, or any combination of these.

5. Thermochemistry estimation

RING allows for estimating thermochemistry of species in the network through the group contribution methodology (Benson, 1976). Specifically, RING provides two options for specifying the group additivity information. First, groups and their contributions to enthalpy, entropy, and specific heat capacity at different temperatures be specified. Second, corrections can be specified by defining fragments and their correction contributions. These corrections can

```
group additivity {
fragment {
  C labeled c1
H labeled h1 single bond to c1
H labeled h2 single bond to c1
H labeled h3 single bond to c1
C labeled c2 single bond to c1
} enthalpy -42.9 kJ/mol entropy 127.12 J/mol/K cp (298=> 25.31, 400 => 32.07,
500 => 38.4, 600=> 44.06, 800=>53.56, 1000=>60.63, 1500=>72.47)
}

group corrections {
gasPhaseSpecies fragment {
  nonringatom C labeled c1 {connected to 3 C with single bond}
nonringatom C labeled c2 single bond to c1 {connected to 2 C with single bond}
} enthalpy 2.9 kJ/mol entropy -0.7 J/mol/K cp (298=> -0.9, 400 => -1.08, 500 => -1.10,
600=> -1.01, 800=>-0.76, 1000=>-0.56, 1500=>-0.35)
}
```

Fig. 8. Sample group additivity and corrections specification for thermochemistry estimation in RING.

account for non-nearest neighbor effects such as gauche or 1,5 interactions.

The user specifies the groups and their contributions as shown in Fig. 8. The first atom in the group additivity fragment is the central atom while the others are neighboring atoms. This is in accordance with Benson's definition of groups. No such differentiation is necessary for group corrections. The term "gasPhaseSpecies" is a characteristic defined by the user to describe molecules that are in the gas phase (and not surface bound). Including this characteristic will result in the fragment correction should being applied only if the molecule is gaseous. RING compiles the additivity inputs into multiple sets of group additivity specifications classified according to the first (or central) atomtype (C, O, C+, C, etc.). For each set of group fragments, a hash value pair, say (h1,h2), is calculated for the fragments similar to that in lumping. The first value of the hash pair, h1, takes into consideration the central atom and immediate neighbor information, while the second value, h2, takes into consideration the electronic configuration of the central and neighboring atoms. While most of Benson's groups only consider nearest neighbors, some groups have additional information about atoms twice-removed from the central atom; specifically, in cases of groups where neighboring carbon atoms are involved in C–C or C–O double bonds, the specific atoms are written as 'Cd' or 'CO' and are differentiated from the regular carbon atom. Note that a group such as C(C)(H)(H)(H) as defined in Fig. 8 is contained in C(Cd)(H)(H)(H). That is, if an atom matches the latter fragment, it will match the first as well. Therefore, any group additivity method needs to check the second fragment before it checks for the first fragment. Similarly, a group such as C(Cd)(Cd)C(H) needs to be checked prior to C(Cd)(C)(C)(H) which in turn needs to be checked prior to checking for C(C)(C)(C)(H). To distinguish neighboring C atoms from 'Cd' or 'CO', the hash value h2 that RING calculates is appropriately modified to reflect the number of double bonds of the neighboring atoms.

When a thermochemistry value (enthalpy, entropy, or specific heat capacity) for a molecule has to be predicted, RING loops over all the atoms in the molecule to determine their contributions and adds them up. Specifically, for each atom, RING determines its atomtype, calculates the hash pair, and checks if the appropriate set of group fragments has a member with the same hash pair value. Note that the hash pair value of the atom takes into account any double bonds of the neighboring atoms. If a group with the atom's hash pair value is not found, then the hash value h2 is recalculated assuming one less double bond to check if the new hash pair has some matching group. This will be repeated successively, until h2 does not take into account any double bonds, at which point if no matching groups are available, RING will throw an error. This procedure ensures that a group with more information (in terms of having 'Cd' or 'CO' over just C) is checked prior to checking for groups with less information (and hence a more generic group). When a matching group is identified, RING checks for matches of the group in the entire molecule. For each match, RING identifies the corresponding central atom (the atom that matches the central atom of the group), finds if the atom's hash pair equals the original value of (h1,h2), accounts for the contribution of those atoms, and removes the atoms from further consideration.

Once RING calculates the contributions of each additive group, corrections are calculated and added by going over each user-defined correction fragment and checking for matches in the molecule. For each distinct match, an appropriate correction is added. The final value is then the molecule thermochemical property value. If specific heat capacity values (cp) are available at different temperatures as shown in Fig. 8, enthalpy and entropy at any temperature can be calculated. For this purpose, cp is calculated using linear interpolation; for example, cp at 450 K is calculated as an average of cp values at 400 and 500 K.

This group contribution scheme has been used to calculate the thermochemical properties of gaseous phase stable and radical species, surface alkoxides arising in Brønsted heterogeneous catalysis, and surface intermediates in metal catalysis. The twin features of additivity and corrections enables defining non-standard Benson-like methods, such as for calculating thermochemistry of surface intermediates on metals (Salciccioli, Edie, & Vlachos, 2012). For example, Salciccioli et al. (2012) distinguish CH_2 groups depending on whether the molecule is gaseous or surface intermediate. This can be handled by first defining the group in the additivity scheme as that for gaseous species and then adding a correction for the group applicable only for surface species (by an appropriately defined characteristic by the user) that is equal to the difference between the surface value and the gas phase value. This method has also been used to calculate thermochemistry of surface intermediates on one metal with that on another reference metal (Rangarajan, Brydon, Bhan, & Daoutidis, 2014) using linear scaling corrections proposed by Nørskov and coworkers (Abild-Pedersen et al., 2007) and to calculate octanol–water partition coefficients (Rangarajan, Bhan, & Daoutidis, 2012a) using atom contribution methods (Wildman & Crippen, 1999), thereby indicating its generality.

5.1. Symmetry corrections

Benson's method also includes a configurational entropy correction term corresponding to the total symmetry number of the molecule. The total symmetry number is calculated in RING based on molecular graph automorphism orbits. The number of automorphic graphs are calculated using the algorithm proposed by Bohanec and Perdih (1993). Several corrections, however, are included because 3D molecular symmetry transformations are not directly obtained from 2D topological symmetry. Specifically, we include: (a) corrections to capture the symmetry number corresponding to the sp^3 C with two sets of topologically equivalent neighbors (as in group $C(B1)_2(B2)_2$ in Muller, Scacchi, and Cme (1991)), (b) reduction in symmetry in spiro compounds as noted by Walters and Yalkowsky (1996), and (c) reduction in internal symmetry number in the case of extended double bond sequence (based on the tabulation of Song (2004)). It should be noted that these corrections are currently included only for species that are interpreted by RING to not be surface intermediates. It is common to calculate translational and rotational entropic contributions of heterogeneous surface intermediates from frustrated vibrational frequencies (Gokhale, Kandoi, Greeley, Mavrikakis, & Dumesic, 2004). Benson's method also requires the calculation of number of optical isomers, for which, RING adopts the methodology to calculate the number of enantiomers and meso compounds using the algorithm proposed by Razinger, Balasubramanian, Perdih, and Munk (1993).

6. Kinetic modeling

Kinetic modeling allows for obtaining quantitative insights, as opposed to qualitative topological network analysis results, by providing information on concentration, yield, and selectivity of each species at different stages (or at different times) in the reactor, sensitivity of outputs to kinetic parameters, and rate determining steps. RING calls the open source software IDAS (Hindmarsh et al., 2005) to solve a differential-algebraic system representing the kinetic model of the complex system assuming each reaction is elementary and follows mass action kinetics. The user has to specify rules to calculate kinetic parameters. Fig. 9 shows how rules to calculate the kinetics of a particular reaction rule are written in RING in context of an example rule, viz., hydrogen abstraction. Conditional "if" statements can be written to assign different kinetic parameters depending on the nature of reactants and/or products. For example, in Fig. 9, the rule specifies that if "r1", which represents a reactant in the rule "Habstraction" as defined by the user while specifying reaction rules, is a methyl radical ([C.]), then assign kinetics on the basis of the nature of the product. The word "primaryRadical" is a molecule characteristic defined by the user to represent primary radicals. The rule, therefore, goes on to specify that if the product is a primary radical, use a particular value (in line 3) for pre exponential factor, activation barrier, and n (temperature exponent); otherwise use another value (given in line 4). If "r1" is not methyl, the values in line 5 are used. This scheme for kinetics specification allows for adding more rules to refine the estimation of kinetics. In addition, activation barriers can also be specified using linear free energy relationships such as the Brønsted-Evans-Polanyi relationships which estimate the barriers as a linear function of the enthalpy of the reaction (Brønsted, 1928). Thermodynamic consistency can be forced by the user by specifying that the specific kinetics be calculated from the kinetics of the reverse step. For example, lines 6 and 7 in Fig. 9 define the kinetics of C–C scission step to be the reverse of that of C–C formation step.

```
1. kinetics Habstraction {
2.  if (r1 is "[C.]") {
3.    if product mol any (mol is primaryRadical)
          {A   5.436e-1 cc/mmol/h Ea  29903.72 J/mol  n  3.65}
4.  A   2.718 cc/mmol/h Ea   5481 cal/mol n   3.46 }
5. A   2.5e10*3600/1000 cc/mmol/h Ea   10400*4.18 J/mol n   0.0 }

6. kinetics CCScission {
7. use reverse of CCformation}
```

Fig. 9. Sample rule inputs into RING for the estimation of kinetic parameters.

The user specifies reactor parameters – pressure, temperature, volume, and inlet flow rates of reactants – and RING solves for an isothermal steady state plug flow reactor PFR. Outputs are flow rate, F_i, of different species in the network at different stages of the reactor, sensitivity of species flow rates to kinetic rate constants (dF_i/dk_j), and degree of rate control or DORC (Campbell, 1994). DORC is defined as $(d\ln r_i/d\ln k_j)_{k_j \neq k_l, K_j}$ wherein r_i and k_j are the rate of production/consumption of species i and the rate constant of reaction j respectively; this value is calculated assuming all other rate constants k_l are fixed except the reverse reaction whose kinetics is determined by K_j, the equilibrium constant, which is also fixed. An example of kinetic modeling of nonane pyrolysis with RING is discussed in the supporting information.

The kinetic modeling module in RING is comparable to other network generators. RING offers the capability of modeling both homogeneous and heterogeneous chemistries. To formulate and solve kinetic models relevant for heterogeneous catalysis, RING invokes the quasi-steady state assumption (QSSA) for surface intermediates and solves the resultant differential-algebraic system of equations that takes into account a site balance for conserving the total number of surface sites. Consistent initial conditions are required to solve such systems, and assuming that initially all surface sites are free leads to inconsistencies because QSSA of other surface species will not hold. RING uses the consistent initial conditions calculation feature of IDAS (Hindmarsh et al., 2005) to calculate the initial concentrations on the surface species. However, one limitation arises in solving systems such as metal catalysis – the equations corresponding to QSSA are nonlinear with the possibility of multiple solutions. A more robust method to obtain the correct initial conditions is therefore currently being pursued. Tools such as RMG (Song, 2004; Van Geem et al., 2006) offers apriori estimation of kinetic parameters (Sumathi & Green, 2002) taking into account pressure dependence for gas phase homogeneous chemistries (Matheu, Green, & Grenda, 2003). Semi-empirical estimation of Arrhenius parameters has been proposed especially for free radical chemistry. This includes: (a) group additivity schemes that can be used to estimate kinetic parameters based on the atoms in the reaction centers and their neighborhood for each reaction rule (Green, 2007; Sabbe et al., 2007; Sabbe, Reyniers, Van Speybroeck, Waroquier, & Marin, 2008; Sabbe, Reyniers, Waroquier, & Marin, 2010; Saeys, Reyniers, Van Speybroeck, Waroquier, & Marin, 2006; West, Allen, & Green, 2011) and (b) pre-defined elaborate rule-based kinetics assignment scheme (Carstensen & Dean, 2009). Both these methods are derived from high-level quantum chemical calculations. These features reduce the dependence on the quality of user-inputs for kinetic modeling. However, such schemes have so far been extensively documented only for gas phase free radical chemistry rules and are not universally used or available across other chemistries. RING provides the option of user specification of kinetics which can accommodate both schemes.

NETGEN (Broadbelt et al., 1994) and RMG (Song, 2004; Van Geem et al., 2006) also employ rate-based construction of reaction networks. This method combines network generation and kinetic modeling whereby species and corresponding reactions are added only when their rates are larger than a threshold value; once a new

reaction and species are added, the integration of the model is restarted. This method has been demonstrated to significantly reduce the size of the reaction mechanism. RING does not currently have rate-based construction. However, we have demonstrated using RING that the size of the reaction of complex systems such as alkane aromatization on solid Brønsted acid catalysts can be pruned down significantly (up to two or three orders of magnitude) by (a) imposing constraints in reaction rules based on inputs from computational chemistry studies, and (b) lumping molecules that are not experimentally distinguished (Rangarajan, Bhan, & Daoutidis, 2012b).

7. Discussion

Several salient and distinctive characteristics of RING can be noted. RING has been demonstrated to be versatile for network generation and querying a broad spectrum of chemistries, such as gas phase free radical, liquid phase acid/base catalyzed, and heterogeneous solid acid/base/metal catalyzed chemistries (Rangarajan et al., 2010). Several options are provided to the users such as (a) constraints on reaction rules, (b) query features, (c) lumping strategies, (d) group additivity-based thermochemistry calculations, and (e) kinetic modeling. These options, further, have a common underlying theme – they are all rule-based. For example, the chemistry is specified through reaction rules, pathway queries in the form of rules to identify molecules and particular type of pathways, and kinetic parameters are specified in the form of rules involving conditional statements. This rule-based feature enables translating expert knowledge into specific instructions, thereby lending the tool flexibility. The domain-specific language used as input to RING provides a high-level, declarative language for describing chemistries to the system. This allows the inputs to be described as conceived rather than translating them into abstractions in general purpose programming languages, and further offers features such as error-checking and optimizations.

DSLs, however, also have several general disadvantages (van Deursen et al., 2000). In particular, one of the biggest is balancing between domain-specific and general purpose language features – offer too few general-purpose language features and the DSL's applicability is very constrained and limited, but offer too many and the DSL turns into just another general purpose language. The language extension model offers a solution to these problems. Instead of needing to design the core reaction rule specification language to handle any possible future development – something that would necessitate many general purpose language features – instead, the core of the language can be kept declarative and simple, intended only to describe the chemistry. Adapting the language to serve new purposes and provide new analyses can instead be accomplished through language extension. While RING already comes with several extensions – molecule, reaction, pathways and mechanisms queries, and kinetic modeling – new modules can be added. Each of these new features can be developed independently as separate extensions that end-users can compose with the RING compiler automatically using Silver, resulting in a language and compiler tailored to their needs.

RING can be used to model and analyze a variety of biomass and hydrocarbon processing systems including homogeneous and heterogeneous chemistries. If only general chemistry of the system is known, topological network analysis can be performed using RING to identify possible pathways and products prior to experimentation, or to identify plausible pathways to experimentally observed products that is consistent other experimental observations. For example, we showed that glycerol dehydration to acrolein on Brønsted acid catalysts necessarily involves 3-hydroxypropanal, consistent with experimental observations that it was a primary product which was observed at low conversions only (Rangarajan et al., 2012b). Further, RING could also be used for mechanism hypothesis and for proposing experiments to discriminate multiple candidate mechanisms. For example, for acetone conversion on Brønsted acid catalysts, we identified among several possible mechanisms one plausible route that was able to match the experimental inference of overall stoichiometric reaction through which two molecules of acetone got converted into one molecule each of acetic acid and isobutene. Based on this pathway, we could propose experiments such as isotope labeling studies that could confirm our predictions (Rangarajan et al., 2012b). In addition to an understanding of the chemistry, if it is possible to determine the energetics (activation barrier and thermochemistry) for each reaction step, then plausible energetically feasible mechanisms can be identified using RING. For example, we have used RING to identify plausible mechanisms for glycerol decomposition and hydrogenolysis to form syn gas or 1,2 propane diol respectively, by using RING's pathway identification features along with semi-empirical estimation of thermochemistry (using group additivity) and activation barriers (using linear free energy relationships) given in the literature (Rangarajan et al., 2014). As a parallel concept within the context of chemical synthesis, RING was used to identify synthetically feasible fatty alcohols from biomass derived oxygenates using heterogeneous catalysis that have potential application in developing nonionic surfactants (Rangarajan et al., 2012a). Networks generated by RING have also been embedded into mixed-integer linear programming problems to simultaneously identify desirable compounds and their optimal synthesis routes in terms of economic, energetic, and reaction rate objectives (Marvin et al., 2013) Finally, if kinetic parameters can be estimated for each reaction step apriori, RING could be used to formulate and solve kinetic models so that quantitative results such as yields, selectivity, rate determining steps, and dominant reactions can be identified.

8. Conclusion

The algorithm and implementation details of RING, a rule-based reaction network generation and analysis tool, are discussed. RING takes in as input the initial reactants and reaction rules, written as instructions in an English-like reaction language, and generates as output the reaction network in the form of reactions and species lists. In addition, post-processing modules allow: (a) lumping functionally equivalent structural isomers to reduce size of the network, (b) querying the network for specific molecules, reactions, pathways, and mechanisms, (c) estimating thermochemical properties of species and reactions in the network, and (d) kinetic modeling. Algorithms and methods from computer science, graph theory, and cheminformatics have been adapted and implemented to (a) develop a domain specific reaction language compiler that acts as a user-interface for RING, (b) represent molecules externally as strings based on SMILES, (c) represent molecules internally as molecular graphs and reactions and graph transformation, (d) match patterns of fragments in molecular graphs, (e) identify pathways and mechanisms as a modified depth-first traversal, and (f) identify lumps and groups in a group additivity scheme using a modified hashing technique. The reaction language provides chemistry-specific syntax, catches semantic inconsistencies in the form of erroneous chemistry rules, and performs problem-specific optimizations to speed up the processing of reaction rules. RING has been applied to construct and analyze a wide variety of chemistries such as homogeneous free radical and acid/base chemistry, and heterogeneous acid, base, or metal catalyzed transformations. RING is available open source (RING, 2013).

Acknowledgements

Financial support from the National Science Foundation (CBET # 1307089, IIS grant # 0905581) and from the University of Minnesota Digital Technology Center is gratefully acknowledged. The authors also acknowledge partial support from the Initiative for Renewable Energy (Large Grant: RL-0004-09), Digital Technology Center, and the doctoral dissertation fellowship at the University of Minnesota.

Appendix A. Supplementary Data

Supplementary data associated with this article can be found, in the online version, at http://dx.doi.org/10.1016/j.compchemeng.2014.02.007.

References

Abild-Pedersen, F., Greeley, J., Studt, F., Rossmeisl, J., Munter, T. R., Moses, P. G., Skúlason, E., Bligaard, T., & Nørskov, J. K. (2007). Scaling properties of adsorption energies for hydrogen-containing molecules on transition-metal surfaces. *Physical Review Letters*, *99*, 016105.

Aho, A., Sethi, R., & Ullman, J. (1986). *Compilers – Principles, techniques, and tools*. Reading, MA: Addison-Wesley.

Benson, S. (1976). *Thermochemical kinetics*. John Wiley & Sons.

Bohanec, S., & Perdih, M. (1993). Symmetry of chemical structures: A novel method of graph automorphism group determination. *Journal of Chemical Information and Computer Sciences*, *33*, 719–726. http://dx.doi.org/10.1021/ci00015a010

Broadbelt, L. J., & Pfaendtner, J. (2005). Lexicography of kinetic modeling of complex reaction networks. *AIChE Journal*, *51*, 2112–2121.

Broadbelt, L. J., Stark, S. M., & Klein, M. T. (1994). Computer-generated pyrolysis modeling – On the fly generation of species, reactions and rates. *Industrial & Engineering Chemistry Research*, *33*, 790–799.

Brønsted, J. N. (1928). Acid and basic catalysis. *Chemical Reviews*, *5*, 231–338. http://dx.doi.org/10.1021/cr60019a001

Campbell, C. (1994). Future directions and industrial perspectives micro- and macro-kinetics: Their relationship in heterogeneous catalysis. *Topics in Catalysis*, *1*, 353–366.

Carstensen, H. H., & Dean, A. M. (2009). Rate constant rules for the automated generation of gas-phase reaction mechanisms. *The Journal of Physical Chemistry A*, *113*, 367–380. http://dx.doi.org/10.1021/jp804939v

Coxson, P. G., & Bischoff, K. B. (1987). Lumping strategy. 1. Introductory techniques and applications of cluster analysis. *Industrial & Engineering Chemistry Research*, *26*, 1239–1248, refdoi10.1021/ie00066a031.

Croes, D., Couche, F., Wodak, S. J., & van Helden, J. (2006). Inferring meaningful pathways in weighted metabolic networks. *Journal of Molecular Biology*, *356*, 222–236.

van Deursen, A., Klint, P., & Visser, J. (2000). Domain-specific languages: An annotated bibliography. *ACM SIGPLAN Notices*, *35*, 26–36.

Eppenstein, D. (1998). Finding the k shortest paths. *SIAM Journal of Computing*, *28*, 652–673.

Fan, L., Bertok, B., & Friedler, F. (2002). A graph-theoretic method to identify candidate mechanisms for deriving the rate law of a catalytic reaction. *Computers and Chemistry*, *26*, 265–292.

Finley, S. D., Broadbelt, L. J., & Hatzimanikatis, V. (2009). Computational framework for predictive biodegradation. *Biotechnology and Bioengineering*, *104*, 1086–1097.

Gokhale, A. A., Kandoi, S., Greeley, J. P., Mavrikakis, M., & Dumesic, J. A. (2004). Molecular-level descriptions of surface chemistry in kinetic models using density functional theory. *Chemical Engineering Science*, *59*, 4679–4691.

Gonzalez-Lergier, J., Broadbelt, L. J., & Hatzimanikatis, V. (2005). Theoretical considerations and computational analysis of the complexity in polyketide synthesis pathways. *Journal of the American Chemical Society*, *127*, 9930–9938.

Green, W. H. (2007). Predictive kinetics: A new approach for the 21st century. In G. B. Marin (Ed.), *Chemical engineering kinetics. Vol. 32: Advances in chemical engineering* (pp. 1–313). Academic Press.

Happel, J., & Sellers, P. H. (1982). *Multiple reaction mechanisms in catalysis*. Industrial & Engineering Chemistry Research.

Heath, A. P., Bennett, G. N., & Kavraki, L. E. (2010). Finding metabolic pathways using atom tracking. *Bioinformatics*, *26*, 1548–1555.

Henry, C. S., Broadbelt, L. J., & Hatzimanikatis, V. (2007). Thermodynamics-based metabolic flux analysis. *Biophysical Journal*, *92*, 1792–1805.

Hindmarsh, A. C., Brown, P. N., Grant, K. E., Lee, S. L., Serban, R., Shumaker, D. E., & Woodward, C. S. (2005). SUNDIALS: Suite of nonlinear and differential/algebraic equation solvers. *ACM Transactions on Mathematical Software*, *31*, 363–396.

Ho, T. C. (2008). Kinetic modeling of large-scale reaction systems. *Catalysis Reviews*, *50*, 287–378, refdoi10.1080/01614940802019425.

Hsu, S., Krishnamurthy, B., Rao, P., Zhao, C. H., Jagannathan, S., & Venkatasubramanian, V. (2008). A domain-specific compiler theory based framework for automated reaction network generation. *Computers & Chemical Engineering*, *32*, 2455–2470.

Ihlenfeldt, W. D., & Gasteiger, J. (1994). Hash codes for the identification and classification of molecular structure elements. *Journal of Computational Chemistry*, *15*, 793–813.

Jeong, H., Tombor, B., Albert, R., Oltvai, Z. N., & Barabasi, A. L. (2000). The large-scale organization of metabolic networks. *Nature*, *407*, 651–654.

Kaminski, T., & Van Wyk, E. (2012). Modular well-definedness analysis for attribute grammars. In *Proceedings of the 5th International Conference on Software Language Engineering (SLE 2012)* (pp. 352–371). Springer Verlag.

Klamt, S., & Stelling, J. (2003). Two approaches for metabolic pathway analysis? *Trends in Biotechnology*, *21*, 64–69.

Kummel, A., Panke, S., & Heinemann, M. (2006). Putative regulatory sites unraveled by network-embedded thermodynamic analysis of metabolome data. *Molecular Systems Biology*, *2*, 0034.

Kuo, J. C. W., & Wei, J. (1969). Lumping analysis in monomolecular reaction systems. Analysis of approximately lumpable system. *Industrial & Engineering Chemistry Fundamentals*, *8*, 124–133. http://dx.doi.org/10.1021/i160029a020

Lee, S., Phalakornkule, C., Domach, M. M., & Grossmann, I. E. (2000). Recursive {MILP} model for finding all the alternate optima in {LP} models for metabolic networks. *Computers & Chemical Engineering*, *24*, 711–716.

Li, G., & Rabitz, H. (1989). A general analysis of exact lumping in chemical kinetics. *Chemical Engineering Science*, *44*, 1413–1430.

Lin, Y. C., Fan, L. T., Shafie, S., Bertok, B., & Friedler, F. (2009). Generation of light hydrocarbons through Fischer–Tropsch synthesis: Identification of potentially dominant catalytic pathways via the graph-theoretic method and energetic analysis. *Computers and Chemical Engineering*, *33*, 1182–1186.

Mali, Y., & Van Wyk, E. (2011). Building extensible specifications and implementations of Promela with AbleP. In *Proceedings of 18th the international SPIN workshop on model checking of software (SPIN 2011), 6823* (pp. 108–125).

Marvin, W. A., Rangarajan, S., & Daoutidis, P. (2013). Automated generation and optimal selection of biofuel-gasoline blends and their synthesis routes. *Energy & Fuels*, *27*, 3585–3594. http://dx.doi.org/10.1021/ef4003318

Matheu, D. M., Green, W. H., & Grenda, J. M. (2003). Capturing pressure-dependence in automated mechanism generation: Reactions through cycloalkyl intermediates. *International Journal of Chemical Kinetics*, *35*, 95–119.

Mavrovouniotis, M. L. (1992). Synthesis of reaction mechanisms consisting of reversible and irreversible steps. 2. Formalization and analysis of the synthesis algorithm. *Industrial & Engineering Chemistry Research*, *31*, 1637–1653.

Mavrovouniotis, M. L., & Stephanopolous, G. (1992). Synthesis of reaction mechanisms consisting of reversible and irreversible steps. 1. A synthesis approach in the context of simple examples. *Industrial & Engineering Chemistry Research*, *31*, 1625–1637.

Muller, C., Scacchi, G., & Cme, G. (1991). A topological method for determining the external symmetry number of molecules. *Computers & Chemistry*, *15*, 17–27.

Otarod, M., & Happel, J. (1992). Studies on the structure of chemical mechanisms. *Chemical Engineering Science*, *47*, 587–592.

Planes, F., & Beasley, J. (2009). Path finding approaches and metabolic pathways. *Discrete Applied Mathematics*, *157*, 2244–2256 (Networks in computational biology).

Planes, F. J., & Beasley, J. E. (2008). A critical examination of stoichiometric and path-finding approaches to metabolic pathways. *Briefings in Bioinformatics*, *9*, 422–436. http://bib.oxfordjournals.org/content/9/5/422.full.pdf+html

Prickett, S. E., & Mavrovouniotis, M. L. (1997a). Construction of complex reaction systems.1. Reaction description language. *Computers & Chemical Engineering*, *21*, 1219–1235.

Prickett, S. E., & Mavrovouniotis, M. L. (1997b). Construction of complex reaction systems. 2. Molecule manipulation and reaction application algorithms. *Computers & Chemical Engineering*, *21*, 1237–1254.

Quann, R. J., & Jaffe, S. B. (1992). Structure oriented lumping – Describing the chemistry of complex hydrocarbon mixtures. *Industrial & Engineering Chemistry Research*, *31*, 2483–2497.

Rangarajan, S., Bhan, A., & Daoutidis, P. (2010). Rule-based generation of thermochemical routes to biomass conversion. *Industrial & Engineering Chemistry Research*, *49*, 10459–10470.

Rangarajan, S., Bhan, A., & Daoutidis, P. (2012a). Identification and analysis of synthesis routes in complex catalytic reaction networks for biomass upgrading. *Applied Catalysis B: Environmental*, *145*, 149–160.

Rangarajan, S., Bhan, A., & Daoutidis, P. (2012b). Language-oriented rule-based reaction network generation and analysis: Applications of RING. *Computers & Chemical Engineering*, *46*, 141–152.

Rangarajan, S., Bhan, A., & Daoutidis, P. (2012c). Language-oriented rule-based reaction network generation and analysis: Description of RING. *Computers & Chemical Engineering*, *45*, 114–123.

Rangarajan, S., Brydon, R. R. O., Bhan, A., & Daoutidis, P. (2014). Automated identification of energetically feasible mechanisms of complex reaction networks in heterogeneous catalysis: Application To glycerol conversion on transition metals. *Green Chemistry*, *16*, 813–823.

Ratkiewicz, A., & Truong, T. N. (2003). Application of chemical graph theory for automated mechanism generation. *Journal of Chemical Information and Modeling*, *43*, 36–44.

Razinger, M., Balasubramanian, K., Perdih, M., & Munk, M. E. (1993). Stereoisomer generation in computer-enhanced structure elucidation. *Journal of Chemical Information and Computer Sciences*, *33*, 812–825. http://dx.doi.org/10.1021/ci00016a003. PMID: 8113334,

RING, 2013. http://research.cems.umn.edu/bhan/software.php

Sabbe, M. K., Reyniers, M. F., Van Speybroeck, V., Waroquier, M., & Marin, G. B. (2008). Carbon-centered radical addition and -scission reactions: Modeling of activation energies and pre-exponential factors. *ChemPhysChem*, *9*, 124–140.

Sabbe, M. K., Reyniers, M. F., Waroquier, M., & Marin, G. B. (2010). Hydrogen radical additions to unsaturated hydrocarbons and the reverse-scission reactions: Modeling of activation energies and pre-exponential factors. *ChemPhysChem*, *11*, 195–210.

Sabbe, M. K., Vandeputte, A. G., Reyniers, M. F., Van Speybroeck, V., Waroquier, M., & Marin, G. B. (2007). Ab initio thermochemistry and kinetics for carbon-centered radical addition and -scission reactions. *The Journal of Physical Chemistry A*, *111*, 8416–8428. http://dx.doi.org/10.1021/jp072897t. PMID: 17676722,

Saeys, M., Reyniers, M. F., Van Speybroeck, V., Waroquier, M., & Marin, G. B. (2006). Ab initio group contribution method for activation energies of hydrogen abstraction reactions. *ChemPhysChem*, *7*, 188–199.

Salciccioli, M., Edie, S. M., & Vlachos, D. G. (2012). Adsorption of acid, ester, and ether functional groups on Pt: Fast prediction of thermochemical properties of adsorbed oxygenates via DFT-based group additivity methods. *The Journal of Physical Chemistry C*, *116*, 1873–1886. http://dx.doi.org/10.1021/jp2091413

Schwerdfeger, A., & Van Wyk, E. (2009). Verifiable composition of deterministic grammars. In *Proceedings of ACM SIGPLAN conference on programming language design and implementation (PLDI)* ACM, (pp. 199–210).

Song, J. (2004). *Massachusetts Institute of Technology*. (PhD Dissertation).

Sumathi, R., & Green, W. H., Jr. (2002). A priori rate constants for kinetic modeling. *Theoretical Chemistry Accounts*, *108*, 187–213.

Ullmann, J. R. (1976). An algorithm for subgraph isomorphism. *Journal of the ACM*, *23*, 31–42.

Van Geem, K., Reyniers, M. F., Marin, G., Song, J., Green, W., & Matheu, D. M. (2006). Automatic reaction network generation using RMG for steam cracking of n-hexane. *AIChE Journal*, *52*, 718–730.

Van Wyk, E., Bodin, D., Gao, J., & Krishnan, L. (2010). Silver: An extensible attribute grammar system. *Science of Computer Programming*, *75*, 39–54.

Van Wyk, E., Krishnan, L., Schwerdfeger, A., & Bodin, D. (2007). Attribute grammar-based language extensions for Java. In *European Conference on Object Oriented Progress (ECOOP)*, *4609* (pp. 575–599).

Van Wyk, E., & Schwerdfeger, A. (2007). Context-aware scanning for parsing extensible languages. In *International conference on generative programming and component engineering (GPCE)*.

Vandewiele, N. M., Geem, K. M. V., Reyniers, M. F., & Marin, G. B. (2012). Genesys: Kinetic model construction using chemo-informatics. 22nd International Symposium on Chemical Reaction Engineering (ISCRE 22). *Chemical Engineering Journal*, *207–208*, 526–538.

Walters, W. P., & Yalkowsky, S. H. (1996). Eschera computer program for the determination of external rotational symmetry numbers from molecular topology. *Journal of Chemical Information and Computer Sciences*, *36*, 1015–1017. http://dx.doi.org/10.1021/ci950278o

Weininger, D. (1988). SMILES, a chemical language and information system. 1. Introduction to Methodology and Encoding Rules. *Journal of Chemical Information and Computer Sciences*, *28*, 31–36.

West, R. H., Allen, J. W., & Green, W. H. (2011). Automatic reaction mechanism generation with group additive kinetics. In *AIChE annual meeting* Minneapolis, MN.

Wildman, S. A., & Crippen, G. M. (1999). Prediction of physicochemical parameters by atomic contributions. *Journal of Chemical Information and Computer Sciences*, *39*, 868–873.

Wipke, W. T., & Dyott, T. M. (1974). Steriochemically unique naming algorithm. *Journal of the American Chemical Society*, *96*, 4834–4842.

Wipke, W. T., Krishnan, S., & Ouchi, G. I. (1978). Hash functions for rapid storage and retrieval of chemical structures. *Journal of Chemical Information and Computer Science*, *18*, 32–37.