Reaction Language Documentation

1. Introduction

The reaction language interface of RING is a domain specific language for high level specification of reaction rules for reaction network generation, and post processing rules for network reduction and analysis. The language is English-like, in the sense that the syntax uses common English terminologies used in the Chemistry and ChemE community. This report is an introduction to the syntax and the grammar of the language. Section 2 discusses the structure and the syntax of the program, while a compilation of typical reaction rules is provided in Section 3 for direct reference.

2. Syntax and grammar of the language

The structure of a program written in RING is as given in the order below

1. Reactant specification
2. Declarations (groups, composite atoms, and characteristics)
3. Global constraints
4. Reaction rules
5. Lumping Strategy
6. Pathway specification
7. Mechanism specification
8. Molecule query
9. Reaction query

The syntax for each one of them is given below. Note that each of these components can be empty. For example, if the compiler doesn't see any lumping strategy specification between reaction rules and pathway specification, it assumes that lumping is not sought.

Note: All syntaxes are given in Arial 12 point font in blue. Any term given within "<<" and ">>" refer to variable names, typically alphanumeric strings, while anything given within "<" and ">" refer to classes/ groups of keywords, commonly referred to as nonterminals. Optional commands are given in Arial 12 point font in orange.

2.1 Reactant specification

The syntax is

```
input reactant "<<SMILES string>>"
```

Note that the SMILES string has to be within double quotes.

SMILES can typically be obtained for a molecule from molecule editors such as ChemDraw.
Note: Molecular Hydrogen can be represented as [HH] or [H][H]. All composite atoms must be within curly brackets. For example, a composite atom like platinum would be "{Pt}". If Hydrogen has to be explicitly mentioned for a non-hydrogenic molecule (that is, molecules other than molecular hydrogen, H+, H-, H:, etc), it is suggested that the atom to which H is attached to is enclosed within a square brackets. So, for example, [{Pt}H] means that platinum is attached to a single hydrogen. In most cases, explicit H is not required and the number of attached hydrogens can be inferred from valency.

2.2 Global constraints

The syntax is

```
global constraints on <<mol id>> {
        <constraints specification>
}
```

Note that mol id is some identifier for molecules. Any alphanumeric string is allowed, but the word "molecule" is recommended in this particular case. The constraints specification commands will be discussed in Section 2.4.

2.3 Declarations

Declarations are of three types: declarations of groups, composite atoms, and characteristics.
A group is a collection of atoms that are all connected, thereby possessing a specific functionality. Groups are user defined upfront and used throughout reaction rules as discussed in Section 2.4. Characteristics are user defined properties that the user can ascribe to molecules as discussed in Section 2.4. Composite atoms are user defined atoms and can be used to: (a) define new elements, such as metals, and (b) define a reaction site such as zeolites.

2.3.1 Groups

Groups are defined as

```
group <<group name>> (<<comma separated label list>>){
        <Assignments>}
```

The Assignments are as declared in Section 2.4. Note that the labels used in Assignments are to be the same as the ones given in the label list.

2.3.2 Characteristics

Characteristics are defined in the language as follows

```
define characteristic <<characteristic name>> on <<mol id >> {
        <constraints specification>
    }
```

The mol id is used as an identifier like in global constraints.

2.3.3 Composite atoms

Composite atoms are declared as follows

```
    define composite atom <<comma separated list>>
```

Note that composite atoms are alphanumeric strings starting with an alphabet.
Note: Only the errors with charge conservation of composite atoms will be caught. The onus is on the user to catch other valency related errors.

2.4 Reaction rules

The syntax for reaction rules is

```
    rule <<rule name>> {
        Prefix reactant <<reactant name>>{
            <Assignments>
        }
        Prefix reactant <<reactant name >> {
            <Assignments>
        }

        constraints {
            <constraints specification>
        }
        Transformation statements
        product constraints on <<mol id>>{

            <constraints specification>
        }
        <Additional Information>
    }
```

Rule name is a single word alphanumeric string. One or two reactant definitions are allowed. Prefixes can be used to describe the reactants, and are discussed in Section 2.4.3. The lines in orange are, therefore, optional. Molecular constraints are given within the “constraints {…}” block. Transformations follow constraints specification, and are followed by product constraints. The mol id variable name for product constraints can be any alphanumeric string, but use of the word “molecule” is recommended.

2.4.1 Assignments

The atoms and their connectivity are described unambiguously in the Assignments block. The atom constraints are also given within the block.

The syntax is

<atomtype> labeled *<<atom label >>* {*<Atomconstraints>* } for describing the first atom. Subsequently, the syntax is

<atomtype> labeled *<<atom label>>* *<bond type>* bond to *<<atom label>>* {*<Atomconstraints>* } for describing all other atoms.

The label name is an alphanumeric string. Each atom of the reaction center, including all reactants, should be assigned a unique label. The atomtype is a recognized electronic state of different elements. The allowed atomtypes are as given in Table 1.

Table 1: Accepted atomtypes

Element	Neutral	Positive	Negative	Radical	Extra lone pair	Positive radical	Aromatic	Onium
Carbon	C	C+	C-	C.	C: (carbene)	-	c	C*
Hydrogen	H	H+	H-	H.	-	-	-	-
Oxygen	O	O+	O-	O.	-	O+.	o	-
Nitrogen	N	N+	N-	N.	-	N+.	n	-
Sulphur	S	S+	S-	S.	-	S+.	s	-
Phosphorus	P	P+	P-	P.	-	P+.	p	-
Composite e.g. Pt	Pt	Pt+	Pt-	Pt.	Pt:	Pt+.	-	-

Note that Hydrogen and composite atoms CANNOT be aromatic.

In addition to specifying an element's atomtype, RING also allows wildcard specifications. Table 2 lists these possibilities. Note that each type has two possibilities, a term and a symbol for specification. For example,

any atom labeled a1 and $ labeled a1 can be used interchangeably to refer to any neutral atom and

any atom + labeled a2 and $+ labeled a2 can be used interchangeably to refer to any positively charged atom.

Table 2: Wildcard element types

Element type	Term	Symbol
Any atom (any of C,H, N, O, S, P)	any atom	$
Heteroatom (N, O, P, S)	heteroatom	&
Heavy atom (all apart from H)	heavy atom	X

In addition to the atomtypes specification given in Table 1, certain prefixes can be added to specify characteristic of the atom. These are: aromatic (the atom is aromatic), nonaroamtic (the atom is not aromatic), ringatom (atom is a part of a ring), nonringatom (atom is not a part of a ring), allylic (atom is in conjugation with a C=C), and nonallylic (atom is not in conjugation with a C=C). For example,

c labeled c1 and aromatic C labeled c1 are equivalent.

Accepted bond type include: single, double, triple, ring (the bond belongs to a ring – can be of any order), nonring (not in a ring – can be of any order), aromatic (can be only between two aromatic atoms), any (can be any bond), strong (can be double, triple, or aromatic), partial (for non bonded interactions, Hydrogen bond, etc).

Atom constraints, limitations imposed on the environment of an atom, can be assigned along with each assignment statement. The atom constraints are specified at the end of each atom assignment's line. Multiple constraints, given in comma separated form, are treated as logical AND operations. The syntax is

! connected to <ConstraintNumber> <atomtype> with <bond type> bond, *<Atom constraints>* .

The bonding specification is optional, with single bond being default. It specifies how the assigned atom is connected to the neighboring atom constrained to be of the given atomtype. The nonterminal "ConstraintNumber" specifies the number of occurrences of neighboring atoms of the specified atomtype. It is optional too, the default being ">=1". The possible options are: (a) n, (b) "> n", (c) "= n", (d) "< n", (e) ">= n", and (f) "<= n", where 'n' is a nonnegative number. The options (a) and (c) are equivalent. Atom constraints can also specify connectivity to groups. In such a case, the syntax is

! connected to <ConstraintNumber> *<<group name>>* with <bond type> bond, *<Atom constraints>* .

The first atom of the group is assumed to be the atom neighboring the assigned atom in this case.

An example of assignment block is as given below:

O labeled o1
C labeled c1 double bond to o1 {connected to 2 C}

This assignment block, in essence, defines a keto group, because the carbon, which is doubly bonded to the oxygen, is connected to two other neutral carbon atoms.

A ring constraint can also be specified as atom constraint. The syntax is

! in ring of size *<ConstraintNumber>*

This constraint is used to specify that the atom is (or is not) in a ring of a given size.

The nature of the assignment statements prevents specifying cyclic reaction centers. The following syntax allows it.

ringbond <<atom label a1>> <bond type> bond to <atom label a2>

Note that the two atom labels 'a1' and 'a2' have both been specified already.

2.4.2 Constraints specification

Molecular constraints can be on size, shape, charge, fragments and characteristics. The following are the possible options

1. Size: The syntax is

    << *reactant name*>>.size

Note that the size can be compared. The comparisons allowed are =, <, and >. The comparison values are integers or integer values. For example, the following syntax checks if the size of a reactant 'a' is less than 10.

    a.size <10

The following syntax checks if reactant 'a' is less than reactant 'b'.

    a.size <b.size

2. Charge: The syntax is

   <<*reactant name* >>.charge

   The operators and comparison values are similar to that of size

3. Shape: The syntax for specifying reactant is cyclic

   <<*reactant name*>> is cyclic

4. Characteristic: The syntax is

   <<*reactant name* >> is aromatic to specify aromaticity as a constraint
   <<*reactant name*>> is oxygenate to specify the reactant is an oxygenate
   <<*reactant name* >> is heteroaromatic to specify the reactant is aromatic, but with a non Carbon atom in the ring
   <<*reactant name* >> is bridged to specify the reactant has a bridged atom
   <<*reactant name*>> is <*declared characteristic*> to specify that the reactant possesses the declared characteristic.
   << *reactant name*>> is <<*SMILES string* >> to specify that the molecule is exactly the one represented by the SMILES string (note the string is given in quotes)
   <<*reactant name*>>.formula is <<*Molecular formula*>> to specify that the molecule's formula is that given (molecular formula of a neutral molecule is given in the order C,H,N,O,P,S, with the number of atoms given after the corresponding element symbol)

5. Fragment specification: The first step is fragment definition

   The syntax for fragment definition is

   ```
   fragment <<fragment name>> {
           <Assignments>
   }
   ```

   Fragment specification syntax is
   <<reactant name>> contains <<fragment name>> for specifying that the reactant has the particular fragment
   <<reactant name>> contains < ConstraintNumber > of <<fragment name>> for specifying that the reactant has the particular fragment the given number of times

Note that fragment definition here is local and is used only within this rule

6. Group specification: This is analogous to fragment specification, the only difference being that unlike the local fragment definition, groups are declared globally at the beginning.
   a contains group <<group name>> implies a contains the specified group
   a contains *<some number>* of <<group name>> implies a contains the specified group the given number of times.

Note that all these constraints are evaluated as Boolean value. Therefore, one can add a "!" in front to negate the requirement. For example, ! mol.size <10 is equal to mol.size >=10 (the size of the molecule is greater than or equal to 10).

2.4.3 Prefixes

Declaration of reactants can have prefixes describing the reactant, instead of certain types of constraints. Allowed prefixes are given in Table 3.

Table 3: Prefixes and their meanings

Prefix	Meaning
positive	Charge =1
negative	Charge =-1
neutral	Charge =0
aromatic	The reactant is aromatic
olefinic	The reactant is olefinic (has a C=C)
paraffinic	The reactant is a paraffin (has only C-C single bonds)
cyclic	Reactant is cyclic
linear	Reactant is linear

For example,

```
neutral reactant R1 {
        C labeled c1
        O labeled o1 double bond to c1
}
```

declares a neutral reactant 'R1' that has a C=O fragment. The Carbon is labeled c1 and Oxygen is labeled o1. In addition to the prescribed prefixes, one can also use a custom characteristic, discussed in Section 2.3.2, as a prefix. Note that multiple prefixes can also be used as long as they do not conflict one another. For example, the prefixes cyclic neutral specifies that the reactant has a ring and is a neutral molecule, but the prefixes neutral positive will throw an error. **Note also that the prefixes should not conflict with constraints.**

2.4.4 Boolean operations

Constraints can be combined in any logical manner using '!' for NOT, '||'for OR, and '&&' for AND. For example,

a.size < 10 || a is cyclic implies that either the size of the reactant is less than 10 or is cyclic. As a second example,
! a is cyclic implies that the reactant is linear.
Parenthesis can be used to specify more complex constraints. For example,
(a.size <10 && a.charge >0) ||(a.size<8 && a.charge =0) implies that either size is less than 10 and charge positive, OR size is less than 8 and charge neutral.

2.4.5 Combined constraints

Constraints in bimolecular reaction can involve both reactants. For example,
a.size + b.size <10 limits the total size of the two reactants to ten, while
a.charge = 0 || b.size >5 requires that either 'a' be neutral or 'b' be atleast 6 atoms in size. Note that if intramolecular reactions are sought through a bimolecular specification, then these combined constraints are **ignored**.

2.4.6 Transformation statements

There are two types of transformation statements – statements that specify atom connectivity changes and statements that describe atomtype changes.

Syntax and meaning for atom connectivity changes is given in Table 4.

Table 4: Syntax for atom connectivity changes

Syntax	**Meaning**
form bond (<<atom label a1>>, <<atom label a2>>)	Form single bond between two atoms, labeled a1 and a2.
form < bond type > bond (<<atom label a1>>, <<atom label a2>>)	Form bond of the given type between two atoms, labeled a1 and a2.
break bond (<<atom label a1>>, <<atom label a2>>)	Break bond between two atoms, labeled a1, and a2.
Break < bond type> bond (<<atom label a1>>, <<atom label a2>>)	Break bond of the given type between two atoms, labeled a1, and a2.
modify bond (<<atom label a1>>, <<atom label a2>>, < bond type>)	Modify bond between two atoms, labeled a and a2 to a specified bond type.
decrease bond order (<<atom label a1>>, <<atom label a2>>)	Decrease the bond order of the bond between atoms labeled a1 and a2. This bond should be a double bond or stronger
increase bond order (<<atom label a1>>, <<atom label a2>>)	Increase bond order of the bond between atoms labeled a1 and a2.

The syntax for describing atomtype changes is
modify atomtype (<<*atom label a1*>>, <*atomtype*>)
This statement modifies the atomtype of the atom labeled a1 to the specified atomtype given within the parenthesis. Note that the specified atomtype cannot be aromatic in nature because aromaticity is decided based on the entire molecule.

2.4.7 Additional Information

There are two types of additional information that can be provided: Setting rule cost and specifying if intramolecular reactions are allowed for a bimolecular reaction.

A cost can be assigned to each rule, as a heuristic measure of the penalty in choosing the reaction of a given reaction rule in a pathway. This parameter is, thus, used only for extracting pathways. Rule cost is set in the additional information block. The syntax is

    rule cost is <number>

The nonterminal "number" can only be a nonnegative integer.
A maximum rank constraint can be assigned to each rule to restrict molecules with a rank greater than prescribed value from participating in a reaction of that rule. This can be given by

    maximum rule rank is <number>

By default, intramolecular reactions are forbidden. But, the user can toggle it on by

    allow intramolecular reaction

If only intramolecular reactions are required, then it can be set by

    only intramolecular reaction

2.4.8 Duplicate declaration

Sometimes, two identical sites can participate in an elementary step. For example, dissociative adsorption of Hydrogen on metal sites requires two adjacent metal atoms. In such cases, the user can specify one of them and declare the other as duplicate. The syntax for duplication is

    reactant <<reactant name A>> duplicates <<reactant name B>> (<label mapping>)

The statement defines a reactant A that is a duplicate of B and the nonterminal "label mapping" provides, in a comma separated form, the one-to-one correspondence between the atom labels of reactant A and that of B. The syntax is

    <<atom label b1>> => <<atom label a1>>, <<atom label b2>> => <<atom label a2,….

Here, there is a one-to-one correspondence set by stating that label a1 maps to the atom labeled b1. Note that b1 is already defined, a1 is the new label being assigned for the reactant A.

2.4.9 Reactant definition using groups

Reactants can also be specified using groups that have been specified.

    reactant <<reactant name A>> group <<group name>> (<label mapping>)

The label mapping provides the one-to-one correspondence between the atom labels for the reactant with the atom labels of the group.

2.5 Lumping Strategy

At the end of all reaction rules, one can provide a lumping strategy. The lumping is done to collect and put together all functionally equivalent isomeric species. Functional equivalence implies that the constituents of the lump all have the same number of each functional group in them. Lumping strategy is optional, the default option being that no lumping is done. The syntax for lumping is

```
Lump all isomers {
        <Lumping options>
}
```

Lumping options include three kinds of specifications

1. Specification of representative molecule for acyclic lumps. The syntax is
   ```
   represent acyclic with farthest apart
   represent acyclic with closest apart
   ```
   to specify that acyclic lumps must be represented by the molecule that has branches the farthest/ least apart from each other
2. Specification of representative molecule for cyclic lumps. The syntax is
   ```
   represent cyclic with farthest apart
   represent cyclic with closest apart
   ```
   to specify that cyclic lumps must be represented by the molecule that has branches the farthest/ least apart from each other
3. Specification of additional lumping for paraffins, olefins, naphthenics, and aromatics based on molecular formula. The syntax is
   ```
   Lump <hydrocarbon> to <most/least> branched
   ```
   The nonterminal "hydrocarbon" includes: paraffin, olefin, naphthenics, and aromatics. Each of these lumps can be represented by most/ least branched molecule in the lump. Note that this lumping will be done subsequent to the previous two specifications, if given.

2.6 Pathway Specification

Reaction pathways can be extracted from a reaction network on the basis of user-defined specification. The user can specify the molecules for which pathways from initial reactants are required, and constraints on the nature of the extracted pathway. The syntax is

```
find pathways to <<molecule id>> {
        <Constraints specification>
}
constraints {
        <Pathway constraints>
} store in <<file name>>
```

The pathways to a molecule satisfying the constraints of the Constraints specification block will be obtained such that the constraints imposed in the Pathway constraints block are satisfied. Furthermore, the pathways obtained are all stored in a file with the given file name (provide file name with extension, preferably .txt). The Constraints specification block is the same as Section 2.4.2, with an additional option of specifying the exact molecule. The syntax for this is

    <<molecule id>> is "*<<SMILES string>>*"

which specifies what the molecule exactly is. The SMILES string is given within double quotes like in Section 2.1. The following constraints can be imposed on the pathways:

1. Maximum length of a pathway
   maximum length *<number>*
   maximum length shortest + <number> to specify a magnitude relative to the shortest possible path, as determined by RING. Give "shortest + 0" for the shortest path
2. Maximum cost of a pathway
   Maximum cost *<number>*
3. Minimum length and cost of pathways
   minimum length *<number>*, cost *<number>*
4. Distinct pathways (two pathways are distinct if the number of reactions of each reaction rule of one is different from that of the other)
   eliminate similar pathways
5. Constraints on rules and molecules occurring in a pathway
   a. Specify how many reactions of a given reaction rule occur in the pathway
      contains *<ConstraintNumber>* rule *<<rule name>>*
   b. Specify how many reactions of a given reaction rule with a molecule of a given type (including SMILES specification)
      contains *<ConstraintNumber>* rule *<<rule name>>* with <reaction member> *<<mol id>>* {
      <Constraints specification>}
      "reaction member" says if constraint is on the reactant (reactant) or the product (product). By default, constraint is checked on the reactant
   c. Specify how many reactions of a given bimolecular reaction rule with molecules of a given type
      contains *<ConstraintNumber>* rule *<<rule name>>* with <reaction member> *<<mol id id 1>>*, *<<mol id id 2>>* {*<Constraints specification>*}
      "reaction member" says if constraint is on the reactants (reactants) or the products (products). By default, constraint is checked on the reactants. The Constraints specification block here includes combined constraints as described in Section 2.4.5. These constraints, when applied on products, checks for every pair of products.
   d. Specifying that a rule occurs only with a particular type of molecule (including SMILES specification)
      rule only occurs with *<<mol id>>* {
      <Constraints specification>}
   e. Specifying how many times a molecule can occur in a pathway
      contains *<ConstraintNumber>* *<<mol id >>* {

```
                    <Constraints specification> }
```

f. Specifying the nature of special reactions such as self reaction (A + A → products) or intramolecular reactions (A → products)

```
        rule only occurs as self reaction
        rule only occurs as intramolecular reaction
```

All logical combinations (AND, OR, and NOT, along with the use of parentheses, as discussed in Section 2.4.4) of the constraints 5a-e can also be used.

2.7 Mechanism Specification

Reaction mechanisms can be queried from the reaction network. The syntax for direct mechanisms is

```
find direct mechanisms to <<molecule id>> {
        <Constraints specification>
}
constraints {
        <cycles  constraints>
} store in <<file name>>
```

The syntax for complete mechanisms is

```
find complete mechanisms to <<molecule id>> {
        <Constraints specification>
}
overall constraints {
        <overall mech constraints>
}
cycles constraints {
        <cycles constraints >
}
store in <<file name>>
```

Constraints on individual cycles – direct mechanisms – are:

1. Maximum length

```
maximum length <number>
```

2. Maximum cost

    Maximum cost <number>

3. Minimum length, cost

    Minimum length <number>, cost <number>

4. Distinct mechanisms

    Eliminiate similar mechanisms

5. Constraints on rules and molecules

    a. All constraints in 5a-e in Section 2.6 is allowed here

    b. The stoichiometric coefficient in the overall reaction

```
contains <ConstraintNumber> <reaction member> <<molecule id>>
{
```

```
                <constraints specification >
        } in overall stoichiometry
```

A negative value in constraint number will refer to reactants, a positive value will refer to products. Therefore, >=1 would mean that the stoichiometric coefficient in the products, for the specified molecule, is 1 or more.

Constraints allowed on complete mechanisms include:
1. 1-3 of allowed constraints for individual cycles
2. Minimum number of cycles

    Minimum cycles <number>
3. Maximum number of cycles

    Maximum cycles <number>
4. Constraint 5b of allowed constraints for individual cycles (constraint on stoichiometric coefficient)

2.8 Molecule query

Species in the network can be queried to get all those in the network that satisfy a given set of molecular constraints. The syntax is

```
find all <<molecule id>> {
        <Constraints specification>
} store in <<file name>>
```

2.9 Reaction query

Reactions in the network can be queried to get all those in the network that satisfy a given set of reaction constraints. The syntax is

```
find all reactions {
        <Reaction constraints >
} store in <<file name>>
```

The following constraints can be specified:

1. Forced specification of only ONE particular rule

    `rule is only <<rule id>>`
2. Specification of the reactants or products of the reaction

    `reaction with <<ConstraintNumber>> <Reaction Member>`
    `<<molecule id>>{ <Constraints specification>}`
3. Specification of reaction rule (different from 1 is that this could be used in an OR statement)

    `reaction rule is <<rule id>>`
4. Specification that the reaction one is looking for is intramolecular only!

    `reaction is intramolecular`