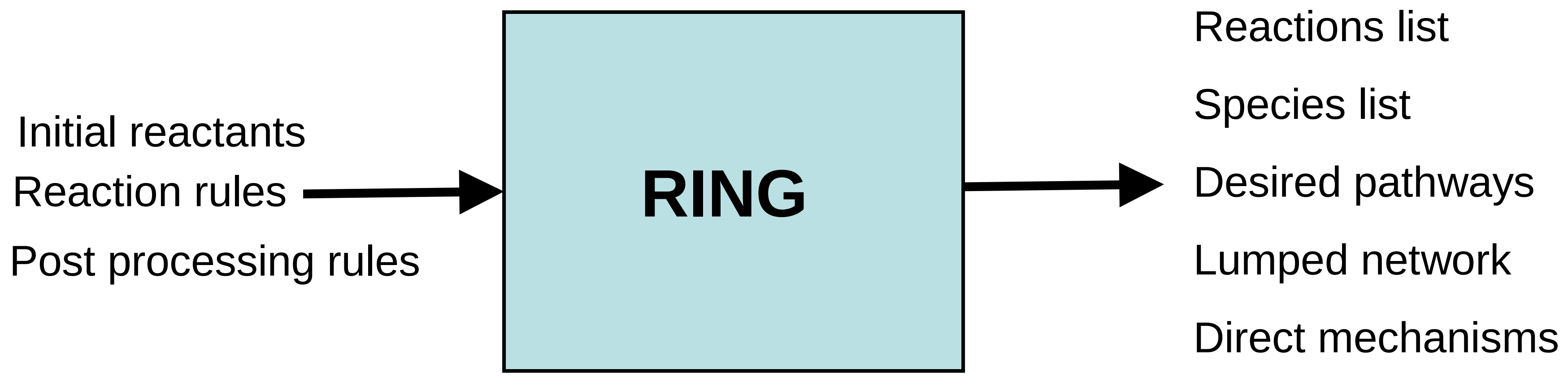
RING (Rule Input Network Generator)

- RING is an automated reaction network generator.
- Major inputs: Reactants, reaction rules
- Major outputs: Reactions, all possible species
- Secondary input: Post processing instructions
- Other outputs: Reaction pathways, lumped network



Language for RING

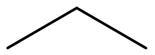
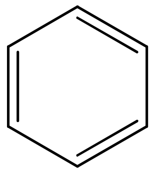
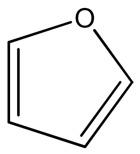
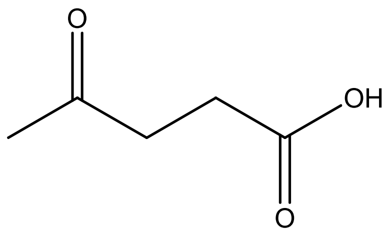
- RING has an English-like front end
- It is a domain specific, high level, specification language for reaction network generation and analysis
- The syntax is based on common Chemistry and Chemical Engineering parlance
- The semantics (meaning) is based on fundamental chemistry principles
- Advantages of using the language as a front-end
  - **High level abstraction – can describe the problem at the level of chemistry**
  - **Catch chemistry specific errors**
  - **Easy programming compared to general-purpose languages**

Reaction Language structure

- Reactants specification
  - Stating the reactants as SMILES strings
- Group/ Composite atom/ Characteristic declaration
  - Defining reusable declarations (more later)
- Global constraints
  - Specifying constraints that apply to all molecules at all times
- Reaction rules
  - Elementary/ non elementary reaction steps
- Lumping strategy
  - Describe how lumping has to be done
- Post processing
  - Describe what pathways/ mechanisms are required

Initial reactants

input reactant “*SMILES string*”

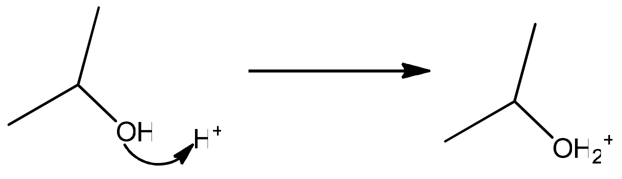
“[H+]”	- Proton	H+
“CCC”	- Propane	
“c1ccccc1”	- Benzene	
“o1cccc1”	- Furan	
“CC(=O)CCC(=O)O”	- Levulinic acid	

Global constraints

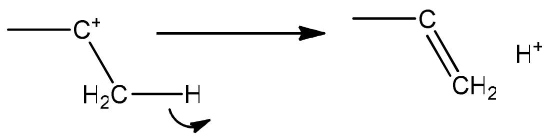
```
global constraints on  Molecule
{
    //declaration of a fragment named 'a'
    fragment a
    {
        C+ labeled 1
        C labeled 2 double bond to 1
    }
    ! Molecule contains a
    Molecule.size < 6
    fragment b
    {
        C labeled c1
        C labeled c2 double bond to c1
        X labeled x1 double bond to c2
    }
    ! Molecule contains b
}
```

Sample reaction rules

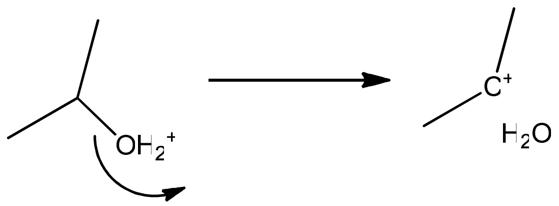
- Sample elementary steps
  - Protonation of alcoholic group
  - Dehydration of oxonium ion
  - Deprotonation of carbenium ion to form a C-C double bond
- Inputs: Fructose, proton



Protonation of alcohol

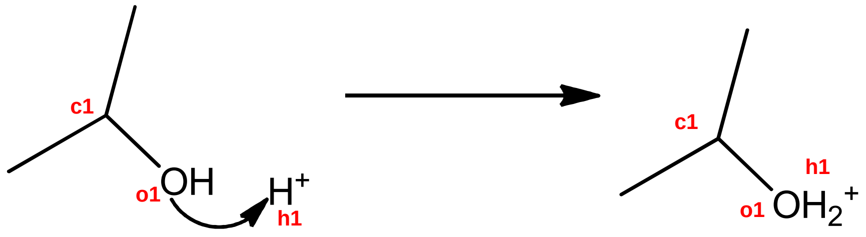


Deprotonation to form olefin



Dehydration

Reaction rules (Protonation of alcohol)



```
rule AlcProt {
neutral reactant r1 {
    C labeled c1 {connected to 2 C}
    O labeled o1 single bond to c1
}
positive reactant r2{
    H+ labeled h1
}
constraints {
    r1 is cyclic}
form bond (o1, h1)
modify atomtype (o1, O+)
modify atomtype (h1, H)
}
```

Rule name “AlcProt”

First reactant “r1” is neutral and has a reactant pattern C-O such that C is labeled c1, O is labeled o1 and bonded to c1 by a single bond; c1 is also connected to 2 neutral carbon atoms, constituting the atom environment constraint

Defining second reactant, which basically is proton, labeled h1

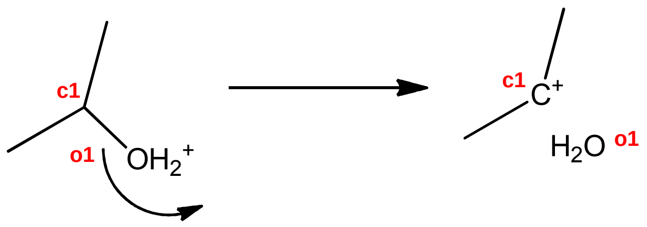
Molecular constraints require, apart from that r1 is neutral and r2 is positive declared as prefixes in while reactant declaration, that r1 is cyclic. Since Fructose that we feed as input is cyclic, it will participate in this reaction

Form bond between o1 and h1 as shown by the arrow in the figure

Oxygen loses its lone pair and acquires a positive charge, while Hydrogen acquires the electron through bonding and loses the charge

More options (prefixes, ring bond, bond prefixes, fragment and pdct constraints)

Reaction rule (dehydration of oxonium)



```
rule dehydration{
reactant r1 {
    C labeled c1
    O+ labeled o1 single bond to c1
}
constraints {
    r1.charge = 1
    r1.size <12
}
break bond (c1, o1)
modify atomtype (c1, C+)
modify atomtype (o1,O)
product constraints on molecule{
    ! molecule is
    primaryCarbeniumIon}
}
```

Rule “dehydration”

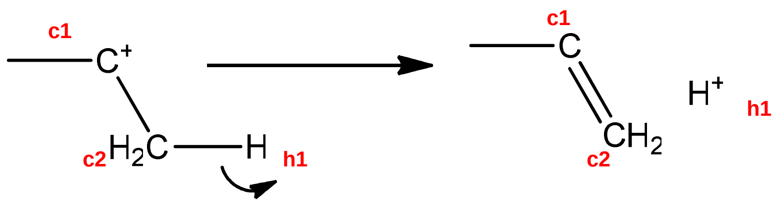
Reactant r1 contains a C-O+ single bond

Molecule constraints include conditions on size (<12) and charge (=+1)

Break bond between the Carbon and Oxygen, as shown by the arrow. Carbon picks up positive charge, Oxygen loses its charge

None of the products can be a primary carbenium ion (note that product constraints apply on all product species!)

Reaction rule (Deprotonation to form C=C)



```
rule Deprot{
reactant r1 {
    C+ labeled c1 {connected to 2 C}
    C labeled c2 single bond to c1
    H labeled h1 single bond to c2
}
constraints{
    r1.charge =1 && r1.size <12
}
break bond (c2,h1)
increase bond order (c1, c2)
modify atomtype (c1,C)
modify atomtype (h1, H+)
}
```

Defining rule “Deprot”

Declaring a reactant r1 with a reactant pattern “C+CH”. The Carbon, ‘c1’ is connected to another Carbon atom, ‘c2’, and the Hydrogen to ‘c2’

Constraints include: Charge or r1 is +1, size <12 atoms

Transformations include, breaking the C, H bond, formation of the double bond to form C=C, the Carbon losing its positive charge, while the Hydrogen leaves as a proton

Lumping Strategy

```
lump all isomers {
    represent acyclic with farthest apart
    represent cyclic with farthest apart
    lump paraffins to most branched
    lump olefins to most branched
    lump naphthenics to most branched
    lump aromatics to most branched
}
```

Allow lumping

Acyclic species are to be lumped so that branches are farthest apart; cyclic to be lumped so that branches are farthest apart.

In addition, paraffins, olefins, naphthenics, and aromatics are to be lumped together based on molecular formula, and represented by the most branched

Pathways specification

```
find pathways to mol{
     mol.size <10 && mol is aromatic
}
constraints {
     maximum length 15
     eliminate similar pathways
} store in “aromaticPathways.txt”
```

Find pathways to all aromatic molecules of size <10

Constraints on the pathways extracted require that length of the pathways is less than 15 and the pathways are distinct (meaning no two pathways have the same number of reactions of each rule). Pathways are stored in the file “aromaticPathways.txt”

```
find pathways to mol{
     mol is “CC(=O)CCC(=O)O”
     } constraints {
     maximum length 12
     contains <=2 rule Hshift
     contains 1 rule Deprot with mol {
             mol.size <=11}
     } store in “LevuPaths.txt”
```

Find pathways to HMF in particular, specified by the SMILES string of HMF

Constraints on pathways require that the length is less than 12, the pathway contains less than 3 instances of the rule “Hshift” (rule not discussed here but is one of the 7 rules input to RING for Fructose to HMF). The rule “Deprot” should be present exactly once and should occur in a reaction with a molecule of size <11 as the reactant. Store pathways in “HMFPathways.txt”

Mechanisms (direct)

```
find direct mechanisms to mol{
    mol.size <10 && mol is aromatic
}
constraints {
    maximum length 15
    eliminate similar mechanisms
} store in “aromaticDirmechs.txt”
```

```
find direct mechanisms to mol{
    mol is “CC(=O)CCC(=O)O”
    } constraints {
    maximum length 12
    contains <=2 rule Hshift
    contains 1 rule Deprot with mol {
            mol.size <=11}
    } store in “LevuPaths.txt”
```

Mechanisms (complete)

```
find complete mechanisms to mol{
     mol.size <10 && mol is aromatic
}
overall constraints {
     maximum length 15
     maximum cycles 5
     contains 1 mol {mol is “CC(=O)O”}
     }cycles constraints {
            maximum length 12
            contains <=2 rule Hshift
            contains 1 rule Deprot with mol {
                          mol.size <=11}
     } store in “AromaticCompleteMechs.txt”
```

Query molecules

```
find all mol{
        mol.size <10 && ! mol is aromatic
        fragment f{
                C labeled c1
                O labeled o1 double bond to c1}
        ! mol contains >=1 of f
} store in “nonAromaticMols.txt”


find all reactions {
        rule is Prot
        reaction with mol{ mol is “C=CC”}
} store in “aromaticMols.txt”
```

Declarations (Groups and composites)

- Groups – collection of interconnected atoms defined upfront for reuse throughout the program

```
group keto(carbon, oxygen)
{
        C labeled carbon {connected to 2 C}
        O labeled oxygen double bond to carbon
}
```

- Composite atoms – User defined atom for convenience. Can be used to define elements other than C, N, S, O, P, H, such as Pt. Can also be used to define groups of atoms with a collective property – zeolite (Zeo), for example. There is no restriction on valency

```
define composite atom Pt, Zeo
```

  - Note that the composite atoms have to be input using SMILES strings – they can be individual atoms, or a part of a molecule. The atom as such is always enclosed in curly braces: So possible reactant input include

```
Input reactant "{Pt}"

Input reactant "[{Zeo}H]"
```

Declarations(Characteristics)

```
define characteristic primaryCarbeniumCarbonyl on mol
{
        fragment f{
                C+ labeled carbon {connected to 1 C}
                O labeled oxygen double bond to carbon}
        mol.charge =1 && mol contains f
}
define group CdoubleC{
        C labeled c1
        C labeled c2 double bond to c2}

define characteristic bigCyclicOlefinicMolec on molec
{
        molec.size >5 && molec contains group CdoubleC && mol is cyclic
}
```