RING: General introduction and installation instructions

RING has three components – a language compiler, a reaction network generator, and post processing module. In this document, how these components are implemented is described in detail. Subsequently, instructions on installing RING are given to help get started.

1. Components of RING

1.1. Language compiler

The language compiler has been built using SILVER and COPPER (http://melt.cs.umn.edu/index.html). The function of this compiler is to take in a program written in the reaction language for a given reaction system and translate it into a C++ file that calls one or more classes and functions of RING. At this point, the release does not include the source code of the compiler, i.e. files written using SILVER to build the compiler, and only a java executable of the compiler is included. Feel free to contact us if you are interested in the source code, we will be happy to share it with you. To run the executable, you'd require java runtime environment.

1.2. Reaction network generator

The network generator, as the name suggest, constructs the reaction network based on the rules and initial reactants given in the program. The classes and functions for the network generator have been implemented in C++. Therefore, a C++ compiler is required. We have, so far, tested our code in gcc and Microsoft Visual C++ Express Edition 2008. We recommend that you use one of these two. The C++ file that the language compiler generates is compiled with the C++ implementations of RING to get an a.out (linux) or an exe file (windows). A linux script and a windows batch file has been included that automates the translation, compilation, and execution.

1.3. Post-processing module

The generated network can be analyzed in terms of finding pathways, mechanisms, querying for reactions and species in the network, and lumping isomers. These options are also written in C++.

2. Installation instructions

RING is distributed in a zip file containing the following folders:

1. bin – contains a jar file of the compiler "react.bin.jar" that translates an input program.
2. cpp – contains C++ implementations of all functions in RING.
3. examples – contains a set of examples written for different kinds of systems.
4. doc – contains some more specific examples with explanations and a documentation of the syntax of the language.

5. Projects – contains a linux script and a windows batch file for running RING. This space is also where the outputs of RING.
6. The parent folder contains the license text. RING is available under GNU Lesser GPL 2.1

Things required before getting started:

1. Ensure that Java JRE (http://www.java.com/) is installed
2. Download and install gcc (g++ for C++) or Microspft Visual C++ express 2008 (http://www.microsoft.com/visualstudio/en-us/products/2008-editions/express, choose to download just the Visual C++ 2008 Express edition with SP1 option) depending on the operating system
3. A good text editor, for example, notepad++ (http://notepad-plus-plus.org/)

Installation Procedure

Unzip the contents of the zip file and save the parent folder in an appropriate place. There are two files in the "Projects" folder: a linux script "ring.sh" and a windows batch file "ring.bat".

If you are a windows user, you need to:

a. Go to the "Projects" folder and open the batch file "ring.bat".
b. You will see a call line: call "c:\Program Files (x86)\...\vcvarsall.bat"
c. Upon installing Visual C++ on your system locate the batch file "vcvarsall.bat" in the "Microsoft Visual Studio" folder which would, by default, be in the c:\Program Files folder. If you use Windows Vista or 7, the path already in there would work just fine; XP users would not require "(x86)" after Program Files. In case, you did not choose the default option of installing Visual C++ under Program Files, you will have to give the appropriate path to the batch file "vcvarsall.bat".

3. Running an input program

Input programs are written as text files with the extension ".txt". To compile a program, run the script (or the batch file) with the program file name as input. For example, in windows:

a. Go to the "Projects" folder.
b. Type, ring ..\examples\Fructose_to_HMF.txt to compile the example program file for Fructose dehydration to HMF in the examples folder.
c. This creates a program.exe executable that can be run by just typing "program"(in Windows; on Linux, the corresponding a.out will be generated)

4. Programming in the reaction language

"RINGTutorial.pdf" in \doc folder has a presentation on how to program in the reaction language. We discuss error checking in detail here. A document "RLDocumentation.pdf" has the details of the syntax of the language.

Syntax errors: These are errors that violate the grammar of the language, for example, misspelling key words, wrong order of key words, etc. These errors, at present, are hard to understand. For example, to specify a reactant that is positively charged, the correct syntax is

```
positive reactant r1{….
```

If, instead, we write

reactant positive r1{….

a syntax error will be thrown.

Error at line 37, column 9, in file TestFile.txt
(parser state: <*some number*>*;* real character index: <*some number*>*):*
Expected a token of the following types:
[react_Ident_t]
Input currently matches:
[react_Positive_kwd]

This error basically says that, there is a syntax error in the input program file "TestFile.txt", line 37, and column 9. A different token was expected than the one currently matched. It may be clear sometimes, like in this case, where it may be evident that the error is associated with the keyword "positive" being in an incorrect position; however, in general, our best advice is to go to the particular line and check the syntax of the line carefully (also check one line above and below) and compare it with the syntax given in the language documentation.

Semantic errors: These are errors that violate chemistry principles (like charge misbalance), referring to undefined labels and variables, and so on. Semantic errors point to the line number and describe the error clearly compared to syntactic errors.

For example, while specifying the transformations, if we do not provide a correct and complete set of bond and atomtype transformations so that there are gross chemistry inconsistencies like charge between reactants and products not being conserved, there is an error that is thrown.

TextFile.txt: 35 Error: Reaction rule Dehydration does not conserve charge!

Here the error clearly specifies that in line 35, where the rule "Dehydration" is first declared, there is a charge conservation error! Following lines will point to specific atoms having that discrepancy.