University of Puerto Rico - Mayagüez Campus
Programming Languages - ICOM 4036
Prof. Wilson Rivera - Spring Semester 2017

Chem Sim

Luis G. Rivera Santiago
Andrés R. Hernández Delgado
Rogelio Díaz Vera

Introduction

As engineering students, we are aware of the problems that our fellow engineers face on their day to day. One of these problems is finding the mass balance in a system. A mass balance is an application of conservation of mass, widely used in engineering, to analyze chemical systems. Mass balances are widely used for the design of chemical reactors, chemical production, and modelling of physical systems. The people that often are in need of calculating mass balances are engineers, especially chemical engineers, and environmental analysts. This situation of having to calculate an unknown mass flow is a simple task that can turn very tedious, very quickly, as the systems being studied grow. Our solution to this problem is ChemSim.

An easy to use tool that can solve mass balance problems would be a huge advancement for all of the professionals and students that rely on these calculations every day to do their work. A tool like this would let the professionals focus on what they really need to focus, the design of the system, instead of the menial math that can be readily and automatically calculated. It will also let the students concentrate on what they need to, learning how to design so they can become the next professionals on their respective fields. ChemSim will be the instrument that they can use to do this. In a way, ChemSim will be advancing the fields mentioned above by promoting a more efficient use of time of the minds that can be working on something else other than monotonous calculations.

ChemSim is aimed at Chemical Engineers and other professionals with no previous programming experience, or with limited time. We want to develop a simple, easy to use programming language for elementary mass balances. The use of the programming language should be simple and fast enough so that it is simpler to use rather than making the math by hand. It will also be easy to learn so that we can fulfill our primary goal, freeing people's minds to do the more difficult tasks. ChemSim will be free and open source, therefore letting students, usually a population without many resources, use it as an educational tool. By making it open source, we are also providing code savvy professionals with a tool that they can easily modify to their own uses.

In summary, ChemSim is a simple and easy to use tool that will allow engineers to calculate mass balances, thus freeing them to work on the more complicated problems. All while being an educational tool for future professionals.

Language Tutorial

Download ChemSim:

1. Make sure to *download* and *install* Python 3 in the computer.
2. *Go to* the ChemSim website **https://andresher.github.io/chemsim/**
3. *Click* “View on GitHub” to access the download page.
4. *Click* “Clone or Download” and then “Download as Zip”.
5. *Unzip* the file.

Run ChemSim from the Terminal:

1. *Open* the terminal.
2. *Change* directory to where the project folder from ChemSim is located.
3. *Run* the command: **`Python ChemSim.py`**
4. *Start* programming a system with ChemSim.

Run ChemSim from the Terminal (with a CSIM file):

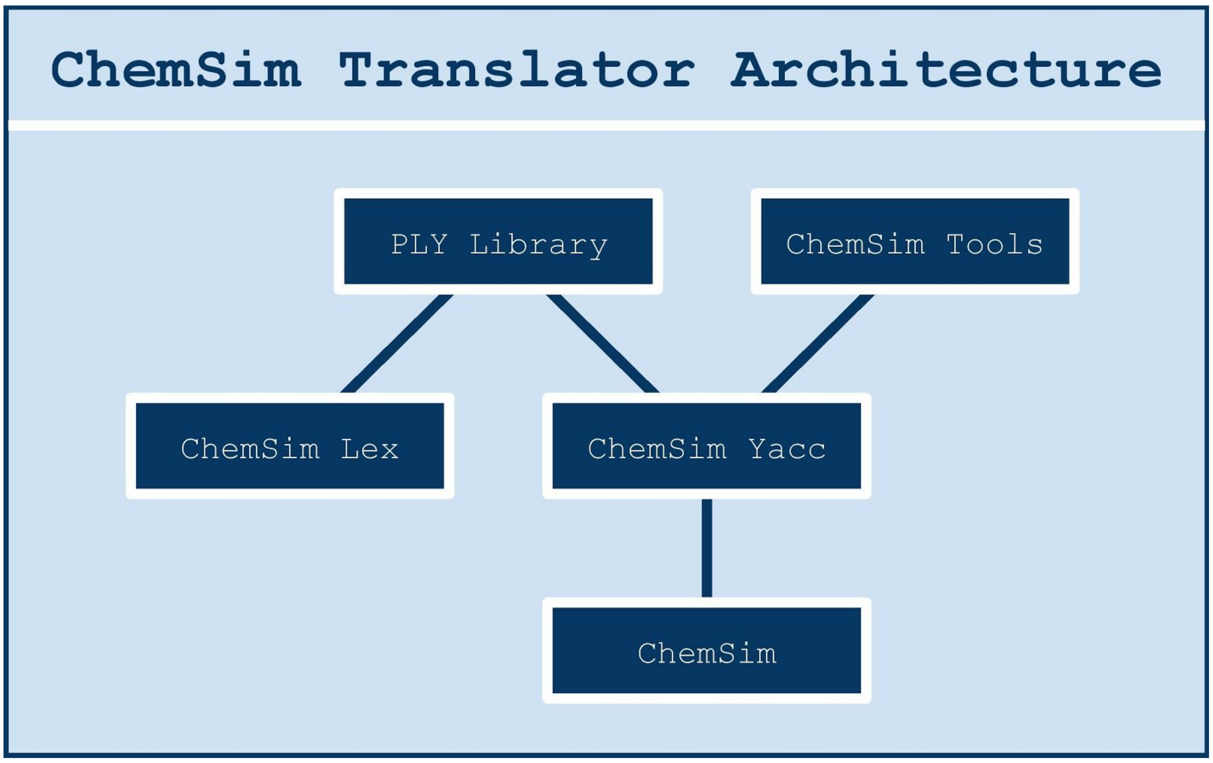
1. *Open* the terminal.
2. *Change* directory to where the project folder from ChemSim is located.
3. *Save* your CSIM file in the same folder as ChemSim.
4. *Run* the command: **`Python ChemSim.py myFileName.csim`**
5. The program will run and your results will be saved.

Language Reference Manual

ChemSim commands:

1. **`create`** - Command used to create a flux, a machine or a system.
2. **`flux`** - Command used to define a Flux. A Flux is defined by a name, a flux speed, and the list of compounds in the flux.
3. **`machine`** - Command used to define a Machine. A Machine is defined by a name, and a list of input and output fluxes that are connected to the machine.
4. **`system`** - Command used to define a System. A System is defined by a name and the machines that are part of the system.
5. **`input`**- Command used to define the fluxes that are part of the input of a machine.
6. **`output`** - Command used to define the fluxes that are part of the output of a machine.
7. **`run`** - Command to run the program and solve the system by calculating the missing flux speed of the system.
8. **`save`** - Command to save the results from the program in a CSV file.

Language Development



Module Interaction

In ChemSim.py is where the main execution starts. Here, the programs keeps running while receiving inputs from the user. These inputs are the commands and parameters required for solving the systems and are provided line by line or with a .csim script file. The execution stops when the user enters the quit command or it is read in the script.

The inputs received are passed as arguments to the parser ChemSimYacc.py, which imports Yacc from the PLY library. The Yacc parser functions are responsible of defining the grammar of the inputs. The Yacc parser verifies that the inputs match the specifications of the tokens defined in the ChemSimLex.py. These specifications consist of the token name followed by the regular expression required in the inputs. This modules also makes the processing of the given commands with the help of the ChemSimTools.py module, where all ChemSim datatypes classes are implemented. After all the inputs are processed, if the user wrote the save command, the program saves the implemented system in a .csv file for better visualization of the outputs.

Software Development Environment

- **PyCharm 2016.3.2 by JetBrains** – Integrated Development Environment (IDE) used to develop applications using the Python programming language. The development team chose PyCharm as the IDE because it has all the tools we need from code analysis to integrated debugger and unit tester. In addition to that, PyCharm is fully cross platform and this is really helpful since some members of our team will be using Windows while others will be using Mac to develop.

- **GitHub** – Web-based version control repository with a graphical interface. It is easy to use, and you don't have to set up a Git server. It also provides several collaboration features, such as a wiki page and basic task management tools that will help us for the project.

- **PLY (Python Lex-Yacc)** – Parsing tool written purely in Python. It is an implementation of Lex and Yacc parsing tools. It consist of two separate tools or modules; lex.py and yacc.py. The first one provides the lexical analysis part while the second one deals with creating a parser.

Test Methodology

In order to satisfy all the requirements for ChemSim, the developers established testing as a priority in the software development process. Therefore, each one of the main modules had their own testing module to make sure they all worked as expected. In addition, all the functions were tested individually once they were implemented. Here are some of the testing procedures and actions used in the development of ChemSim:

- ChemSimLex Module - A testing module (ChemSimLexTester) was implemented to be able to test different tokens in a command-line environment. The module helps identifying and labeling tokens from sample code inputs.

- ChemSimTools Module - A testing module (ChemSimToolsTester) was implemented to test that all the functions used to create and solve a system were working as they should.

- Test Function - A test function, that is also a reserved word in the lexer, used to test if the whole program is working properly from the ChemSim main module. It will print the results that are saved in the CSV file showing if the system was successfully created and solved.

Testing Sample Program

```
    create flux StartFlux 23 H2O 70 NaCl 30

    create flux Flux1 13 H2O 70 NaCl 30

    create flux Flux2 ? H2O 70 NaCl 30

    create machine Machine1 input StartFlux output Flux1 Flux2

    create system System1 Machine1

    run System1

    save System1
```

Conclusion

While working with ChemSim, we were able to understand better how a programming language works. It is important to have the syntax and lexical analyzers well defined in order to have a well structured language, but it takes a lot of thinking to know how would be the most efficient and convenient way to implement them and to implement the command processing functions. And doing that, is not an easy task. Initially we planned to make ChemSim able to solve a complete mass balance system, but due to our limited time and the difficulty of the matter, we made it able to solve only simple systems. And even though they were simple problems, it took a lot of time and work to make ChemSim capable to solve them. At the end of the project, we realized how much work is needed in order to have a complete programming language, how useful and important it is in our lives.