University of Puerto Rico at Mayaguez

Department of Electrical and Computer Engineering

ICOM 4036 Programming Languages

LogicsAI: *Final Report*

Carlos Donato Rivera

Gilberto Jiménez Orench

Angel Pérez Fernández

Luis Sala Ortíz

Prof. Wilson Rivera Gallegos

ICOM4036-036

Introduction

Nowadays, programmers have implemented computer applications made to conduct a dialogue between the user and itself. These applications are known as chatterbots, and uses artificial intelligence algorithms to have conversations with humans. A common example world-wide known would be the web application “Cleverbot”, created by Rollo Carpenter. This chatterbot has the ability to learn from human input. However, the world of artificial intelligence is quite vast, and still developing, which means this language will only serve for a small fraction of the idea of artificial dialogue between computer applications and humans.

Nonetheless, for most human conversations to be positively achieved, there has to be a logical interaction between two speakers. When using declarative sentences, a truth value is implied behind each of them. For example, if we got the following two statements, “Feathered animals are birds”, and “Crows have feathers.”, then the following sentence “Crows are birds.” makes sense and it is considered to be true.. Therefore, we must include Logic as a philosophy to apply to the chatterbot application because by logic is how humans reach common understanding.

The “LogicsAI” language helps to interpret a dialogue between human and computer application based on the logic value of the input. In other words, the user can perform logic operations such as NOT, AND, OR, IF, and many more operations. The user can make a dialogue with the computer program based in the truthiness of the inputs the user has given. Also, given the circumstances when the answer does not have enough information to make a conclusion, some artificial intelligence algorithms can also provide a mechanism of self-learning for the computer program, finally making able to interact some kind of conversation between user and chatterbot program.

Language Tutorial & Reference Manual

The main purpose of LogicsAI is to detect several syntax components from user input and interpret it in order to correctly obtain an output. Once up and running the user can input simple statements, with the following considerations:

- For subjects, the user should precede them with a # symbol.
  - Ex. dog -> #dog
  - Ex. The sky is blue -> The #sky is blue
- For objects, the user should precede them is a @ symbol.
  - Ex. blue -> @blue
  - Ex. The sky is blue -> The sky is @blue.
- For negation of objects, the user should precede them with the symbols !@.
  - Ex. red -> !@red
  - Ex. The rose is not red -> The rose is not !@red.

The input statements are written as explained in the console running Logics AI.

- Ex. The sky is blue -> The #sky is @blue.

After entering such input, the run command should be executed by entering the word "run" in the console. The console will show the statement just entered in other to allow the user to confirm it visually.

After entering statements as the previously discussed, the user would be able to ask through the console and receive an answer.

- Ex. Is the #sky @blue?
- Answer: Yes

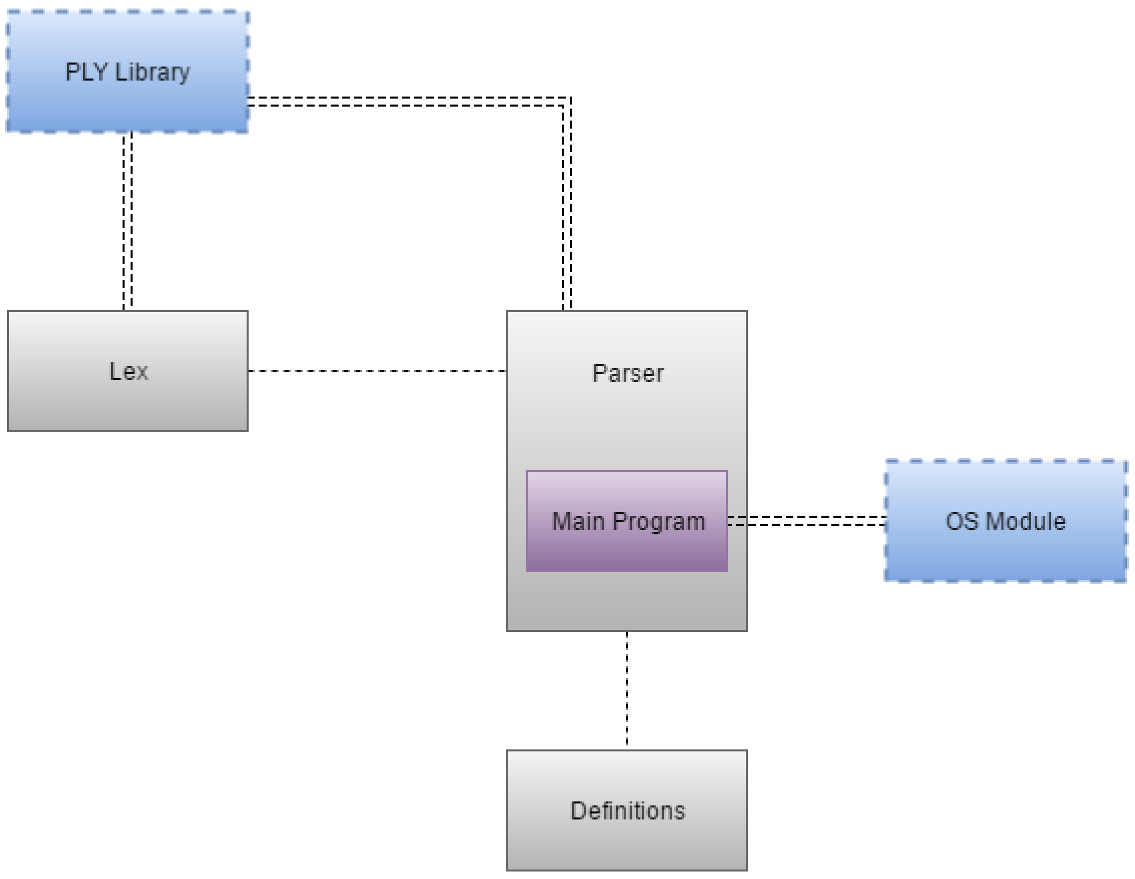
There are several ways to write the input that would be recognized by LogicsAI:

- THE SUBJECT IS OBJECT
- THE SUBJECT IS NOT COMPOBJ

- HE SUBJECT AND SUBJECT ARE OBJECT
- THE SUBJECT AND SUBJECT ARE NOT COMPOBJ
- THE SUBJECT IS OBJECT AND NOT COMPOBJ
- IF THE SUBJECT IS OBJECT THEN SUBJECT IS OBJECT
- IS THE SUBJECT OBJECT
- WHAT IS OBJECT
- WHAT IS NOT COMPOBJ

Development

Translator Architecture



Interfaces between Modules Description

In the LogicsAI translator architecture, the Main Program module is the most important as it is within the Parser module and it shares the objects used as part of the code generation process while interpreting our programming language, as well as other important objects. In the Main Program module, the LogicsAI lines of code are received as input from the user and are then passed as an argument to the Yacc parser imported to the Parser module from the PLY library. This Yacc parser uses the Python functions defined in the same module. These functions define the grammar characteristics of the LogicsAI programming language specified in Backus Naur Form (BNF) format. Furthermore, within these functions we can find the corresponding algorithms required to interpret the language, which are executed as part of the code generation process. One of those functions, executed as BNF expressions, are recognized in the LogicsAI's line of code, which the Yacc parser is currently evaluating.

In order for the Yacc parser to be able to match those Parser functions against the LogicsAI code, it must consider the token specifications which are defined and provided by the Lex module. These token specifications consists of the token name, and the regular expression necessary to match those tokens in any LogicsAI code provided as input.

Finally, in the Main Program , all the Python code necessary to run the AI simulation is extracted and saved on a Python file named by default 'output.py'. At this point, as a final step, the Python program described in the output file is executed by the use of the OS module which comes as part of the Python language.

Software Development Environment Description

These programs were used as part of the LogicsAI translator development process:

Anaconda - A python compiler, which was very useful when installing the necessary packages to make the program possible.

- Anaconda was developed considering Python 2.8

Trello- An online visual platform which is great for developing ideas and working with a team to create the best software designs based on the shared ideas.

Test Methodology Description

We tested mostly the Parser and Lexer modules before testing the system in as a whole. The following actions and descriptions were taken into consideration:

● Lexer - We make sure that the tokens were identified and labeled correctly after

inserting some sample code inputs.

● Parser - We managed the BNF statements to make the Yacc parser to properly execute the corresponding Python functions.

● Program itself - We make sure each line of code was properly stored.

Conclusion

While doing the project and dealing with the challenges that arose from working with Artificial Intelligence systems, we realized how difficult it was. We started with an ideal perspective of what LogicsAI should have be, but along the way we had to modified it according with the mechanisms, tools and time that we had available. First, the working tools that were envisioned for the language were completely different to the ones used. The library PLY was ideal for the lexer and parser, but not for everything we needed. Even when other tools were found to be potential aids in the process, we couldn't use them because they didn't entirely helped in our goals. As a result, LogicsAI is just able to manage simple syntax statements. Regardless the fact that LogicsAI didn't turned out what we wanted from the beginning, it is a good start point to reach the original goal. Working on this project proved to be a great way to learn the real challenges of creating a programming language. No matter the results, we gained valuable knowledge that without doubts will help us in many other matters.