WWU Department of Computer Science

Grail

A Python Smalltalk Translator

Adam Taylor

Walla Walla University
SEVENTH-DAY ADVENTIST HIGHER EDUCATION

ABSTRACT

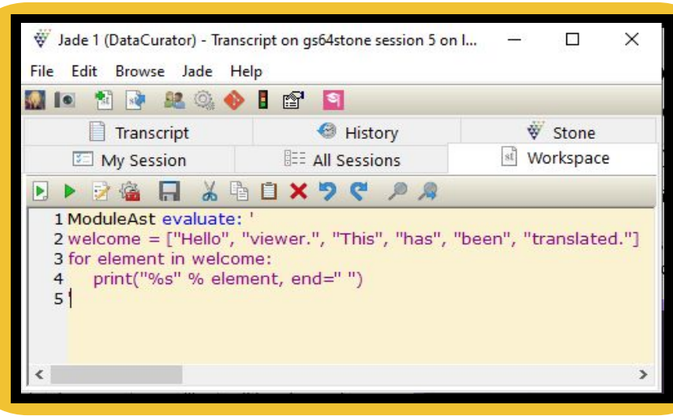
Grail is a project that seeks to parse a Python AST into executable Gemstone Smalltalk code. This aims to provide Smalltalk's key relational database integration and scalability to one of the most popular and data-oriented languages, Python. Over the past year, progress has been made on a scoped variable system, many built-in functions, and string injection. The previous members of this project created all the skeleton classes, a system that reads the Python AST, a sizable portion of basic types, a working function creator, loops, and if-else statements. In completing this project phase, many concepts were used, including parent-child object relationships, Euler's identity of complex powers, and double-dispatch object methods. Due to my contribution, the Grail project can run algorithms limited by variable usage, three-quarters of Python's built-in functions, and input and output can be formatted according to Python type standards.

INTRODUCTION

Python is a familiar and useful language that is loved by many developers. It extends its reach into the machine learning and data science sectors and provides a massive contribution to the computer science field. That said, it has its limitations. Grail, this project, is a Python to Gemstone Smalltalk translator. The goal of this process is to allow developers to work with the adaptable and familiar Python syntax but still be able to access the power and scalability that Smalltalk offers. This poster will specifically discuss the most challenging contributions I made to this project. This will include a discussion on implementing a scoped variable system and the unique features required by a Python-style variable system. It will also include a deep dive into the complexities that Python math functions face because of their compatibility with imaginary numbers. Finally, it will describe the difficulty of creating a versatile string injection programming with multiple width, precision, and tagging options.

VARIABLES

Smalltalk comes with two types of variables that are not associated with object instances. One can create local variables by surrounding a word in pipes, or create a global variable by adding it to one of the built-in system dictionaries. This means that there is no scoping in Smalltalk, so a variable is either global or local and cannot be accessed in different functions. To solve this problem, an inverted linked list was used where each node in the list accounts for a scope. Each node contains a symbol table, used for variable accessing. This implementation allows later scopes the ability to see variables in previous scopes but not the other way around.



BUILT-IN FUNCTIONS

Not all of the functionality of Python has an equivalent version in Smalltalk. Because of this, some of the built-in functions of Python had to be reverse engineered and turned into a new function. One example of this is the power function. Smalltalk does not have the ability to raise complex numbers to powers or raise a real number to an imaginary power. Because of this, the power function needed to be created from scratch. This required the object oriented programming technique of double dispatch to minimize the code used and to respect the hierarchy of numbers and the different formulas required to perform exponentiation. Three quarters of these built-in functions were implemented during this phase of the project.

CONCLUSION

This project has been highly informative. It has given insight into unique and valuable features in Gemstone Smalltalk and Python. They are vastly distinct types of languages, making some challenges exceedingly difficult to implement. They also have enough similarities that problems one might think would be a challenge only took a few minutes to create. While there were other tasks completed in this project, these were the most time-consuming features I implemented.

SUMMARY

Bellow is a list of some of the accomplishments of this project. This also serves as a recap for the rest of the poster.

- A variable system was implemented.
- Global and local key works created.
- Made the parsers for iterables functional.
- Three quarters of the Python builtin functions were implemented, including the power function.
- Functionality was added to built in types, such as the string modulo, or string formatting, method.
- Pre-existing math methods were translated to double dispatch methods.
- Python code can now be written in string in the Smalltalk environment, translated, and then run.

REFERENCES

Smalltalk manual:
https://www.gnu.org/software/smalltalk/manual-base/html_node
Python AST and built-in function descriptions:
https://docs.python.org/3/library/ast.html
https://docs.python.org/3/library/functions.html

June 2023, WWU Computer Science Department, sponsored by James Foster