HamSkill: Run Haskell Anywhere with Jython

CS252 Project Proposal

Zayd Hammoudeh
(zayd.hammoudeh@sjsu.edu)

1 Running in the Java Virtual Machine

C is one of the most commonly used languages when the goal is maximum performance. However, C/C++'s "write once, compile anywhere" paradigm limits its portability. In contrast, the near ubiquity of the Java Virtual Machine (JVM) allows it to be "write once, run anywhere."

On many occasions, developers have leveraged the Java environment to allow other languages to take advantage of the JVM's "run anywhere" capability. Examples include: JRuby for the Ruby programming language [1], Scala [5], Renjin for the R programming language [4], and Jython for the Python programming language [2].

Currently, there is no full implementation of Haskell in the JVM. One Haskell dialect that is runnable in Java is Frege [6].

In this project, I will implement, *HamSkill*, a stripped down version of Haskell in the Java Virtual Machine.

2 Implementation Proposal

This section outlines the overall implementation plan for this project. It is divided into subsections based on the overall themes and ideas.

2.1 Programming Language

When selecting the implementation language, my criteria were: runnable in Java, maximum flexibility, and I wanted to have some exposure to the language to reduce the number of variables. The language that best fit this criteria was Jython. My preliminary experiments indicate that with some degree of manipulation the overall plan is achievable.

```
# Create a temporary memory space
temp_heap = {'x': 10}

# Define the recursive function as a string.
my_func = """\
            def string_fibonacci(n):
                if n==0 or n==1:
                    return n
                return my_func(n-1) + my_func(n-2)

            result = my_func(x)"""

# Run the function above
exec(my_func, temp_heap)

# Print the result
print temp_heap['result']
```

Figure 1: Jython Function Creation on the Fly Using `exec`

For example, figure 2.1 shows example code I write in Jython that is executed purely from a string. This paradigm will allow me to take code either from a file or from command line and execute it in real time. What is more, this code also supports passing arguments as I am doing with argument "`x`".

Some of the guidance for implementing this style of code came from [3].

2.2 Supported Types

HamSkill will only support a select subset of Haskell's available types. The list of planned types are: `Bool`, `Integer` (i.e. bounded), `Char`, and `List` (currently only finite lists are planned, but that may change depending on the speed and complexity of the implementation).

While implementing floating point numbers would not add substantial complexity at a basic level, ensuring that the floating point behavior of *HamSkill* and Haskell are identical is beyond the scope of this project.

2.3 Memory Management

Given the use model of Python's `exec` command as shown in figure 2.1, the local variables will need to be kept in a Python Dictionary. I expect that I will need to build some amount of structure around that to ensure objects are immutable. However, that remains to be seen.

2.4 GHCII-like Behavior

One of the reasons I am planning to use `exec` and not simply translate code to a file is to allow for a GHCI-like terminal where a user can enter supported Haskell code, and it will be executed in real-time by *HamSkill*.

2.5 Higher Order Support

By representing functions as strings, I expect I will be able to pass them as parameters and perform actions on them. The specifics of this part of the implementation has not been entirely thought through. There may need to be some unforeseen tradeoffs.

2.6 Currying

Given that the heap object I pass into the `exec` function is under my control, I expect to be able to perform currying of functions.

3 Tentative Schedule

If you are satisfied with this plan, I will draft a schedule. I did not want to put too much thought into this until I got your buyoff on the plan as a whole.

Bibliography

[1] Home jruby.org. http://jruby.org/. (Accessed on 02/24/2016).

[2] Platform specific notes. http://www.jython.org/archive/21/platform.html. (Accessed on 02/25/2016).

[3] python - using exec() with recursive functions - stack overflow. http://stackoverflow.com/questions/871887/using-exec-with-recursive-functions. (Accessed on 02/25/2016).

[4] Renjin.org — about. http://www.renjin.org/about.html. (Accessed on 02/25/2016).

[5] The scala programming language. http://www.scala-lang.org/. (Accessed on 02/25/2016).

[6] "Ingo Wechsung". Frege/frege: Frege is a haskell for the jvm. it brings purely functional programing to the java platform. https://github.com/Frege/frege. (Accessed on 02/25/2016).