A brief status report for ruby_curry codebase

This document summarizes the working and components of the entire project using 2 methods :

1. Code workflow explanation
2. Directory-wise file descriptions

Installation/Usage Instructions

Following are the instructions to setup ruby_curry on a local machine :

1. Clone from the Github repository
   git clone https://github.com/karthiksenthil/ruby_curry
   cd ruby_curry

   OR

   Download and extract from the zip file of codebase
   wget https://github.com/karthiksenthil/ruby_curry/archive/master.zip
   unzip master.zip
   cd ruby_curry-master

Inside the ruby_curry codebase you can explore the different files. Description of each directory and files is given below in “Directory-wise file descriptions”.

In order to use the compiler for a sample Curry program(in specific JSON format), follow these instructions :

1. Save the JSON file for the program in the “*user”* directory inside ruby_curry codebase
   Example : user/append.json
2. Move into the user directory
   cd user
3. Compile the JSON into ruby object code
   ../bin/cmd.sh <program_name>
   Example: ../bin/cmd.sh append (**Note** : File extension JSON should not be provided)
4. That will create a Ruby file in the same user directory with name *“program_name_objectCode.rb”* . This can be executed using a ruby interpreter :
   ruby program_name_objectCode.rb
   Example: ruby append_objectCode.rb

Code Workflow

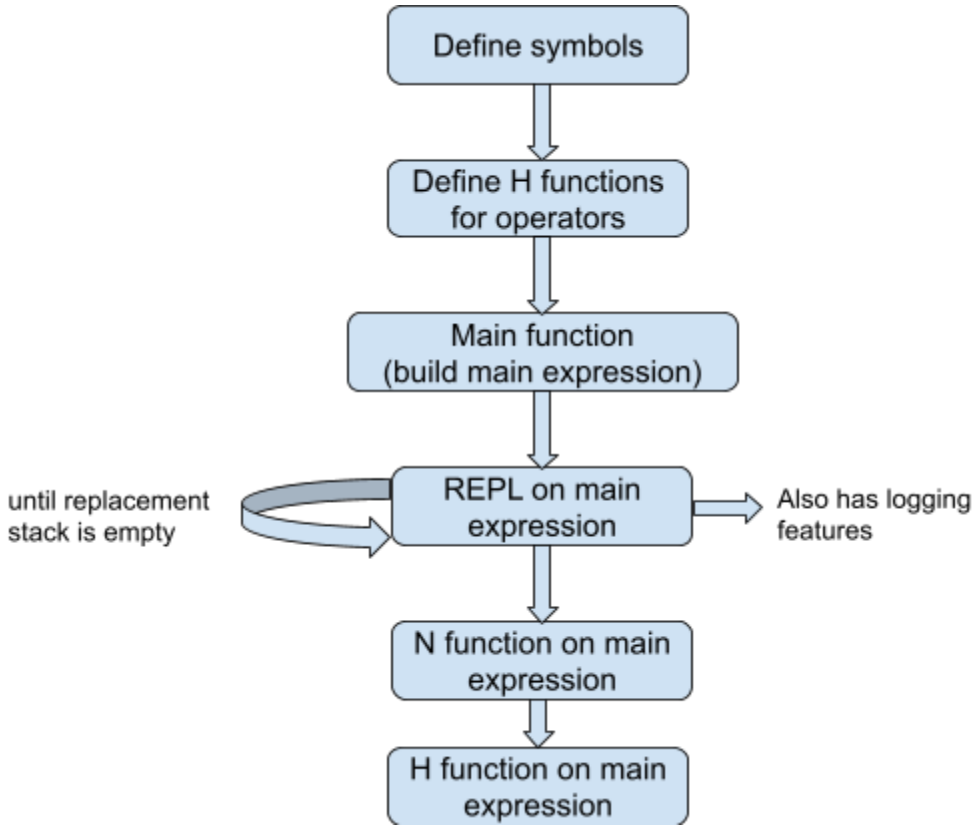
Here we have 2 components - generation of Ruby object code, running the object code

1. Running the object code

Here we discuss the code workflow in the tool whenever the Ruby object code(produced by the compiler) is executed.

Example for use-case : append_objectCode.rb

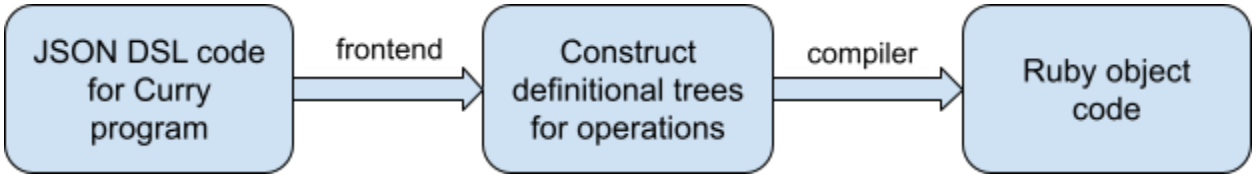
- The object first defines all the symbols in program i.e Constructors and Operators. Example in this case : nil_list, cons, true, false, append, main
- The object code also defines the **H function** for each Operator symbol. Example - append, main



- **REPL function**
    a. The function in a loop
    b. It executes N function on top level expression (on stack)
    c. Checks if backtracking is needed (if CHOICE is found on stack)
    d. Terminates when backtracking is complete i.e replacement stack is empty

- **Backtrack function (src/compiler/repl.rb)**
    a. Requires debugging and redesign
    b. Failed for “permute” curry program with 3 colors

2. Building Object Code



Example : append.json

- **src/compiler.rb**
  Input : JSON file
  Output : object code

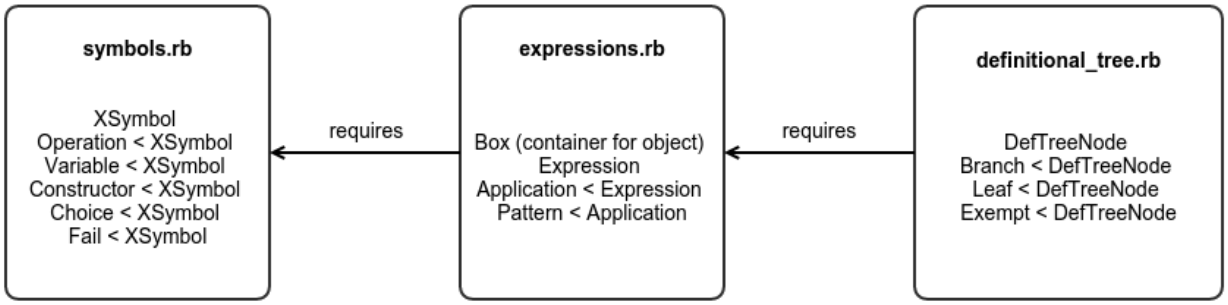
    a. compiler function
        - Parse JSON and create symbol table and operation rules data structures
        - Populate program constructors and program operations hashmaps (these hold the Symbol objects directly)
        - Populate program data types with CurryType objects
        - Construct definitional tree (*handled by modules in* “**frontend”** *directory*)
        - Construct CurryModule object to represent a module in Curry program
        - Build object code for the module object and write into object code file

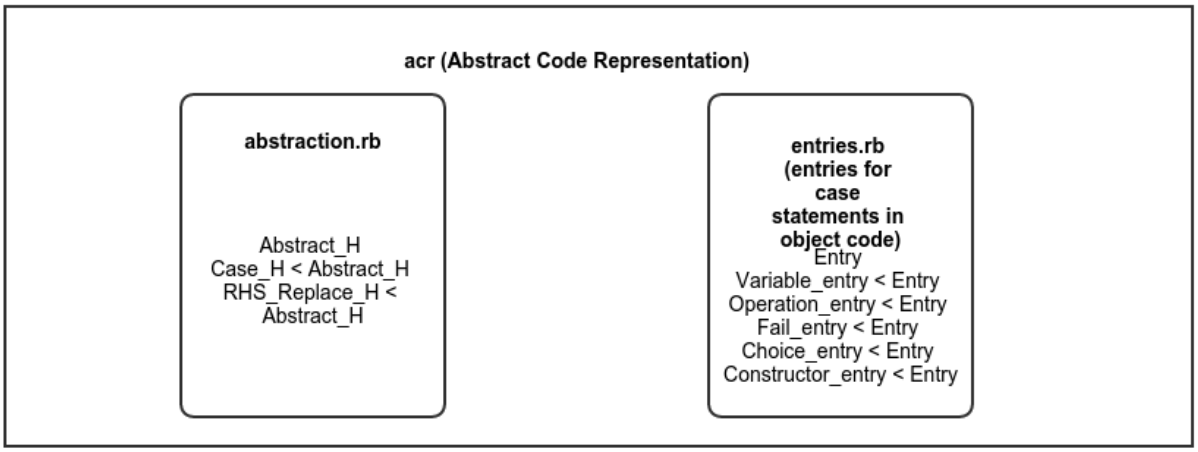
Modules in “frontend”

a. frontend/parsers.rb
Input : JSON file
Output : symbol_table and operations_rules
- modules to parse JSON and convert into in-memory hashmaps
- It populates symbol_table and operations_rules data structures using these hashmaps
- module to parse Curry expressions is also in this file

b. frontend/def_tree_construct.rb

- Implements Bottom-Up strategy for constructing definitional tree (Barry thesis Pg.17)
- Methods and algorithms can be found in Pg. 44 of the thesis





Class Diagram for important classes in ruby_curry

Directory-wise file descriptions

1. bin

cmd.sh --> script to run the compiler for creating object code. Takes program/module name as parameter.

Example : in user dir, run ../bin/cmd.sh append && ruby append_objectCode.rb

2. examples

contains some sample Curry programs represented using the JSON frontend format that ruby_curry uses.
Current examples - append, half, less-than-equal
TODO - arithmetic_replace.json, reverse.json

3. src

Contains the source code files. Contains following sub-directories -

a. compiler
   - Contains source code to implement the compilation component of ruby_curry i.e convert definitional tree to object code
   - Some files not covered in Code workflow in this sub-directory are :-
   1. **generate_h.rb** -> reopens the Branch, Leaf and Exempt classes to specialise and define compile function
   2. **curry_module.rb** -> defines CurryType and CurryModule classes
   3. **utilities.rb** -> defines some general utilities like print, logging,etc. and class specific utility functions for Application, Variable

b. frontend
   - Contains source code to handle the frontend of ruby_curry i.e convert JSON to definitional tree
   - Details about files in this directory are explained the Code Workflow section

c. runtime
   - Source code for runtime support functions/modules used by compiler module
   1. **function_A.rb** -> reopens the Box class to define function A for expressions
   2. **function_N.rb** -> reopens the Box class to define N function for expressions

4. test

- Contains the test suite for ruby_curry
- Uses the Ruby Test::Unit module
  TODO : make complete test suite for all features in ruby_curry

5. user

- Consists of user play area. Allows users to define the JSON files for a Curry program and compile it to object code using ruby_curry