Judah Walker

Programming Language Foundations

December 15, 2016

Project

My submission is basically two separate parts:

First, is my implementation of a system, which on distributed components will process a top level concurrent while command and output the result in a graph.

Second, are grammars, rules, and Forms of Judgments, which describe the above process. Here is a detailed list of what is in the submission.

Theory:

1. Create grammar

2. Create rules

3. Create forms of judgment

Implementation:

1. Parsing

1a. 0,c,sigma -> 0 for root of tree, c for the while command, sigma for mapping. This will be called state and is a common element sent and received from process to process. Code parser for this

1b. Parser for concurrent WHILE -note the shell of this is taken from the github project the meaning of programs online that I sent out in the proposal.

2. Create distributed processes:

2a. Gate: receives initial state, parses input into a process readable code and stores that as new c, sends state to gamma.

2b. Gamma: receives state, sends state to sink, generator

2c. Generator: receives state, creates # of subroutines for count('||') + 1, modifies tree with number corresponding to each branch, sends n different states to n newly generated subroutines.

2d. Subroutine: receives state, reads last number of passed tree. 0.1 would mean do small step of left tree branch. Example: x := 1 of x:=1 || x := 2. While 0.2 would mean do x := 2 of x:=1 || x := 2. If no concurrency is present in command, it will just do a small step with 0...1 as tree. After small step, update c with that step removed while updating mapping. Send new state to gamma. Note: simple command d, would be completely finished when sending new state to gamma.

2e. Sink: receives state, stores state in list, which is permanently in its output queue. Sends list to Omega.

2f. Omega: receives list, displays list on screen. List would be similar to [0,c,sigma; 0.1,c',sigma'; 0.2,c'',sigma''; 0.1.1,c''',sigma'''; ..; ]. I figured out a way to write code that would looks for similar c’s and remove a branch if that occurred anywhere else in the tree.

3. Use some type of library to display output list in a nice pretty graphical tree format.

Following that model, here is the general structure of my submission:

Note that most of the webservers share similar constructions and the main differences are in app/controllers/

foundations/

--parser/ - Contains tree top grammars for parsing

--expression.treetop - Expression parser

--while.treetop - While parser

--small\_step/ - Contains files for instructing the behavior of while elements

--add.rb

--assign.rb

--await.rb

--boolean.rb

--do\_nothing.rb

--equal\_to.rb

--expression\_machine.rb

--greater\_than\_or\_eqaul\_to.rb

--greather\_than.rb

--if.rb

--less\_than\_or\_equal\_to.rb

--less\_than.rb

--multiply.rb

--number.rb

--sequence.rb

--statement\_machine.rb

--variable.rb

--while.rb

--synatx/ - Contains the defined syntax for the commands

--add.rb

--assign.rb

--await.rb

--boolean.rb

--do\_nothing.rb

--equal\_to.rb

--greater\_than\_or\_eqaul\_to.rb

--greather\_than.rb

--if.rb

--less\_than\_or\_equal\_to.rb

--less\_than.rb

--multiply.rb

--number.rb

--sequence.rb

--variable.rb

--while.rb

--parser.rb - Links parser

--small\_step.rb - Links small state\_pre

--syntax.rb - Links syntax

gamma/ - Contains the server setup for gamma

...

--app/

--controllers/

--gamma\_controller.rb

--gamma2\_controller.rb

...

gate/ - Contains the server setup for gate

...

--app/

--controllers/

--gate\_controller.rb

...

generator/ - Contains the server setup for generator

...

--app/

--controllers/

--generator\_controller.rb

...

omega/ - Contains the server setup for omega, also contains some extra graphs for example outputs

...

--app/

--controllers/

--omega\_controller.rb

**#this file has on line 160 opening the MAC OSX preview app, if this is run on a non mac machine, you will need to comment this out or change to the appropriate jpg viewer.**

**graph1.jpg**

**graph2.jpg**

**graph3.jpg**

...

sink/ - Contains the server setup for sink

...

--app/

--controllers/

--sink\_controller.rb

...

subroutine/ - Contains the server setup for subroutine

...

--app/

--controllers/

--subroutine\_controller.rb

...

foundations.rb - Links to while

Gemfile

start.rb - Initial step to start the distributed process

README.docx - This document!

theory.pdf - My scan of the theory portion. It is somewhat hard to read I understand due to additions of more forms of judgment. Hope it makes sense. Let me know if you would like to see it in writing.

Now lets move to actually implement the code.

First, you need to get the gems install, so make sure you have ruby, on the terminal:

IOWACML-T3AEG8W:~ uwalkj6$ ruby -v

ruby 2.1.6p336 (2015-04-13 revision 50298) [x86\_64-darwin12.0]

Next, you need to have 'bundler' installed. http://bundler.io/

Will need a ruby 2.1+ for this to work. Then cd to project/ where Gemfile is and do (you may need to do a 'bundle install' as well)

IOWACML-T3AEG8W:project uwalkj6$ bundle update

Fetching gem metadata from https://rubygems.org/

Fetching version metadata from https://rubygems.org/

Fetching dependency metadata from https://rubygems.org/

Resolving dependencies....

Using rake 12.0.0

...

Using rails 4.2.7.1

Bundle updated!

Then in each server piece do, (you will need to open a seperate terminal for each process):

IOWACML-T3AEG8W:gate uwalkj6$ bin/rails server -p 2000

=> Booting WEBrick

=> Rails 4.2.7.1 application starting in development on http://localhost:2000

=> Run `rails server -h` for more startup options

=> Ctrl-C to shutdown server

[2016-12-08 09:36:51] INFO WEBrick 1.3.1

[2016-12-08 09:36:51] INFO ruby 2.1.6 (2015-04-13) [x86\_64-darwin12.0]

[2016-12-08 09:36:51] INFO WEBrick::HTTPServer#start: pid=3063 port=2000

IOWACML-T3AEG8W:gamma uwalkj6$ bin/rails server -p 3000

=> Booting WEBrick

=> Rails 4.2.7.1 application starting in development on http://localhost:3000

=> Run `rails server -h` for more startup options

=> Ctrl-C to shutdown server

[2016-12-08 09:37:00] INFO WEBrick 1.3.1

[2016-12-08 09:37:00] INFO ruby 2.1.6 (2015-04-13) [x86\_64-darwin12.0]

[2016-12-08 09:37:00] INFO WEBrick::HTTPServer#start: pid=3064 port=3000

IOWACML-T3AEG8W:generator uwalkj6$ bin/rails server -p 4000

=> Booting WEBrick

=> Rails 4.2.7.1 application starting in development on http://localhost:4000

=> Run `rails server -h` for more startup options

=> Ctrl-C to shutdown server

[2016-12-08 09:38:21] INFO WEBrick 1.3.1

[2016-12-08 09:38:21] INFO ruby 2.1.6 (2015-04-13) [x86\_64-darwin12.0]

[2016-12-08 09:38:21] INFO WEBrick::HTTPServer#start: pid=3074 port=4000

IOWACML-T3AEG8W:subroutine uwalkj6$ bin/rails server -p 5000

=> Booting WEBrick

=> Rails 4.2.7.1 application starting in development on http://localhost:5000

=> Run `rails server -h` for more startup options

=> Ctrl-C to shutdown server

[2016-12-08 09:37:25] INFO WEBrick 1.3.1

[2016-12-08 09:37:25] INFO ruby 2.1.6 (2015-04-13) [x86\_64-darwin12.0]

[2016-12-08 09:37:25] INFO WEBrick::HTTPServer#start: pid=3069 port=5000

IOWACML-T3AEG8W:sink uwalkj6$ bin/rails server -p 6000

=> Booting WEBrick

=> Rails 4.2.7.1 application starting in development on http://localhost:6000

=> Run `rails server -h` for more startup options

=> Ctrl-C to shutdown server

[2016-12-08 09:37:29] INFO WEBrick 1.3.1

[2016-12-08 09:37:29] INFO ruby 2.1.6 (2015-04-13) [x86\_64-darwin12.0]

[2016-12-08 09:37:29] INFO WEBrick::HTTPServer#start: pid=3070 port=6000

IOWACML-T3AEG8W:omega uwalkj6$ bin/rails server -p 7000

=> Booting WEBrick

=> Rails 4.2.7.1 application starting in development on http://localhost:7000

=> Run `rails server -h` for more startup options

=> Ctrl-C to shutdown server

[2016-12-08 09:37:33] INFO WEBrick 1.3.1

[2016-12-08 09:37:33] INFO ruby 2.1.6 (2015-04-13) [x86\_64-darwin12.0]

[2016-12-08 09:37:33] INFO WEBrick::HTTPServer#start: pid=3071 port=7000

Now we are ready to start! Open up one more terminal, cd to project, and here are some examples:

ruby start.rb "x := x + 1 || y := x - 2 || z := y - x || await (z = -1) then (x := 100)" "{x->2, y->2, z->3}"

ruby start.rb "if (x > 10) then (while (x > 10) do (x := x - 1; y := y + 1)) else (while (x <= 10) do (x := x + 1; y := y + 1)) || x := 10" "{x->8, y->0}"

Be sure to control-c out of start.rb when you are done processing or it will continue to ping gamma. **Note again, if you can't get this working, there are sample graphs in omega.**

I will include the formal piece in a separate doc pdf.