CS225 Spring 2018—Final Project Write-Up

Aaron Longchamp
github.uvm.edu:@alongcha

Connor Allan
github.uvm.edu@crallan

May 6, 2018

Introduction: We were able to create small step semantics and type checkers for the functions try-with and raise. We also had to create a specific result of error. This result is so that functions like specific try with semantic rules could be used. There is one small step semantic that required the first expression to be an error so that the second expression was expressed. Below all of this text are the semantics that we added into the OCaml final program.

Small-Step Semantics: For small step semantics we determined and defined whether the function stepped to a value, function, another step, or got stuck. To do this we used old functions given to us that we had to create in past homeworks to get a baseline of what we were able to use and how the language was going to work. From there we used the semantics that were given in the book and converted them into OCaml code and made it look like what the other functions looked like. From there we had to do a bit of correction in our syntax and make sure that we use the correct values when using the functions so that the tests that we would implement later would be correct. After all the syntax was corrected and the file would work when trying to make it we moved on to the next part.

Type Checking: During the type checking we had to then convert these functions to be a type checker. We again took old code that we had created from past assignments to use as a baseline and starting point. We referenced the book for the typing semantics, so we would be correct when writing the code in OCaml. The book specified that we needed typing semantics for the two functions try-with and raise. We ran into some issues when determining how to use the typing rule that has to do with expression types. This is seen in the typing rules for raise. In this we must make sure that the type of t1 is of type expression and then return that type. We got around this by creating a new type definition of Exp(). This allowed us to determine the type of t1 and got the correct result of the type within the Exp(). After we got over this hump we were able to move on to the next part.

Test Cases: This final part was a bit more challenging than the other parts. This was because there was not really a framework that we could work off. We

first had to go through and create what the test cases were and to do this we just went down the list creating what we thought would give us what we were looking for. After writing those down we double checked that they would work and wouldnt just end in a stuck. When that was finished we wrote out what the outcomes should be and then changed up the ones that didnt quite work for what we wanted. We then set to the task of using code that you had given us in the last assignments to just add in the tests and check the rules. In this we found a rule or two that didnt quite give us the correct outcome, so we adjusted them. This then finished the coding part of this assignment.

Conclusion: There were many challenges that we had to overcome. Between busy schedules, computer issues, correct syntax, and learning latex. These were all able to be fixed and overcome as shown in the final projects completion. We also have different amounts of commits on Github due to a lot of peer coding. Connor would be working on his part of the code and then Aaron would help and vise versa. To avoid many merge conflicts, we would combine the documents on one computer and then that person would submit it to Github. This was useful because we wouldnt have to run through all of the code looking for small changes that one did and the other didnt notice. Overall this was an interesting experience and was enjoyable for the most part.

1 Small Step

1. $\frac{}{if\ T\ then\ e_2\ else\ e_3 \rightarrow e_2}$
2. $\frac{}{if\ F\ then\ e_2\ else\ e_3 \rightarrow e_3}$
3. $\frac{e_1 \rightarrow e_1'}{if\ e_1\ then\ e_2\ else\ e_3 \rightarrow if\ if e_1'\ then\ e_2\ else\ e_3}$
4. $\frac{}{projl(<v_1,v_2>) \rightarrow v_1}$
5. $\frac{e \rightarrow e'}{projl\ e_1 \rightarrow projl\ e'}$
6. $\frac{}{projr(<v_1,v_2>) \rightarrow v_2}$
7. $\frac{e \rightarrow e'}{projr\ e \rightarrow projr\ e'}$
8. $\frac{}{try\ v\ with\ e \rightarrow v}$
9. $\frac{}{(raise\ v)\ e \rightarrow raise\ v}$
10. $\frac{}{v_1\ (raise\ v_2) \rightarrow raise\ v_2}$
11. $\frac{e \rightarrow e'}{raise\ e \rightarrow raise\ e'}$
12. $\frac{}{raise\ (raise\ v) \rightarrow raise\ v}$

2 Types

1. $\frac{}{\Gamma \vdash T\colon bool}$
2. $\frac{}{\Gamma \vdash F\colon bool}$
3. $\frac{\Gamma \vdash e_1\colon bool \ \Gamma \vdash e_2{:}T \ \Gamma \vdash e_3{:}T}{\Gamma \vdash if \ (e_1)(e_2)(e_3)\colon T}$
4. $\frac{\Gamma \vdash e\colon (T_1 * T_2)}{\Gamma \vdash projl(e)\colon T_1}$
5. $\frac{\Gamma \vdash e\colon (T_1 * T_2)}{\Gamma \vdash projr(e)\colon T_2}$
6. $\frac{\Gamma \vdash e_1\colon T_{exn}}{\Gamma \vdash raise \ e_1\colon T}$