CS51 Final Project

May 2021

Extension: Lexical Dynamics

For my extension, I wrote a lexical evaluator function using the lexically scoped environment semantics rules. The implementation was similar to the dynamic environment evaluator.

There were three differences in how lexical semantics operated from dynamic semantics. The first was in the evaluation of the function. In the lexical semantics, the function evaluates to a "closure" where the the function is returned along with the environment at that point in the evaluation. The purpose is to ensure that previously defined variables that were used in the function definition go unchanged. In the dynamic environment, this kind of updating is allowed. As a result, the second difference between dynamic and lexical environments is their handling of recursive functions. In dynamic environments, since variable updating can occur at any point, when it comes time to evaluate a recursion, the semantics simply use the previous definition of a function. Therefore, "let rec" and "let" statements involve identical implementations in the dynamic environment. In lexical environments, the function definition updates don't occur automatically and are handled manually. The initial evaluation of the function definition is done in an environment that maps the function to the value "Unassigned" to prevent issues when the evaluator encounters the recursion. The result of this evaluation is then manually inserted for the mapping to "Unassigned" in the environment. This must be done with a direct update to the value stored at the reference. The third and final difference between the dynamic and lexical evaluators is in the implementation of function application. This difference is what leads to different evaluations in the two environments, particularly when a variable used in a function is updated. In the lexical environment, variables used within a function that were previously defined will always hold the same value within those functions, even when the variable name is later updated. This is why in lexical environments, the function evaluation returns the function and a "snapshot" of the environment at that point. In the function application stage, it is this lexical environment that is used in evaluating the function. The dynamic environment, however, allows updates to variables in the environment so variables used in a function which are defined after the function itself have the latter value.

Given how similar the two environment evaluators were, I found that it would make sense to combine their definitions. For this purpose, I created the Env_evaluator module to ensure that only the eval_l and eval_d functions that I defined within it would be make accessible to a user. Within the module, I defined a new evaluator type that only has the two options "Lexical" and "Dynamic". I then wrote a env_eval function that takes in this evaluator type in addition to an expression and environment. In the three cases where the lexical and dynamic evaluation differed, I checked which evaluator was called in the function and implemented the appropriate rule. I used this function to define eval_l and eval_d within the module so I would be able to call these functions outside of it to define the actual evaluators to be used in miniml.