Hole instance renumbering for unique holes

Jonathan Lam

2022/01/13

1 Motivation

The original motivation is the performance problem stated in the GitHub issue #536 on the Hazel repository. The problem is that there is a blowup of numbered (tracked) hole instances in the final `HoleInstanceInfo.t` object. These hole instances are counted in `Program.renumber` right after evaluating the expression. The renumbering operation gives a unique instance number to each hole that is encountered, and then recurses through the environments of the holes (renumbering the holes in the substituted expressions).

The problem is that it performs a full DFS through every hole instance, and considers every hole instance distinct from other hole instances. The problem is that it causes an absurdly large number of instances for each hole, even in programs where the evaluation takes many fewer steps. This may confuse the Hazel user: two hole instances that share the same environment, and thus will necessarily have the same value when the hole expression is filled, will have different hole instance numbers.

The solution is to unify all instances of a hole which share the same (physical) environment. This will require changes to hole `InstancePath.ts`. This will be greatly helpful for the fill-and-resume operation described in the POPL 2019 paper.

1.1 Other performance optimizations

When investigating the above issue, I realized a few other inefficiencies that worsen the performance issue. `Program.get_result` is called extraneously many times (four times per `ModelAction.t`). The program result can be stored in the `Model.t`, and the program does not have to be re-evaluated after every action: it should only be re-evaluated after appropriate edit actions and can be memoized with respect to the `UHExp.t` program expression.

The exact memoization and structural sharing characteristics of environments (e.g., the implementation of `Environment.extend`) may slightly affect performance, but are not nearly as important as the exponential slowdown experienced with the hole renumbering.

2 Illustration

Consider the following Hazel program:

```
let a = (* hole 1 *) in
let b = \x.{(* hole 2 *)} in
b 4 + b 5 + (* hole 3 *)
(* yields:
 * (hole 2:1 with env e1) + (hole 2:2 with env e2) + (hole 3:1 with env e3)
 * with e1 = { a = (hole 1:1 with env nil), x = 4 },
 *      e2 = { a = (hole 1:2 with env nil), x = 5 },
 *      e3 = { a = (hole 1:3 with env nil), b = \x.(hole 2) }
 *)
```

3 Proposed solution

4 Implications