{Setɪy}: Optimizing a Set Language — Milestone Report

Adam Blank (adamblan@cs) and Andrey Kostov (akostov@andrew)

Major Changes

The only significant change is that we realized that the proper implementation of set re-use along with the first subgoal of universe inference will automatically fulfill the second subgoal of universe inference (as best as is possible without huge overhead).

Other changes include subtle differences in the implementation from what we anticipated, but those were to be expected.

Accomplished so far

We have finished tracking sets which includes when they are first created, the last location in code at which the set is morphed and the set of last usages of the set (past which its memory can be freed). We also use this information to track sets that we consider constants (we decided these would be only sets that contain only numbers) and reuse them whenever the opportunity arises. Note that any sets that can have more than one value are also considered not constant.

We have also implemented automatic free variable inference (which was the first subgoal of universe inference). The best way to describe what this goal actually means is via a few code examples.

In the following code example, after the implementation of the language feature the compiler can automatically infer the fact that the domain of x is the set $N := [100]$, withouth this being explicitly specified.

```
N := [10]
print {x | y in N minus {0,1} and z in N minus {0,1} and x = y*z}
```

In this following code sample, the compiler can now properly infer the domain of y and thus the code would work properly without the code that specified the domain of y.

```
equivalence_classes(p) :=
   S    := pi1(p)
   dom  := pi1(pi2(p))
   codom := pi2(pi2(p))
   return { {y | y in codom and (x,y) in S} | x in dom}
```

Milestone

We have achieved the milestone we set for the project in the proposal and have even gone slightly over. We are well on track for achieving at least our 125% goal.

Surprises encountered

The first surprise was how long finishing up the compiler took. There are in fact still some minor bugs

left that do not hinder our progress, but finishing up the compiler was definitely more work-intensive than expected.

Something else that was unexpected was the difficulty of keeping proper track of sets (which is our first listed accomplishment). The main difficulty that arose were due to that of determining which sets were not constant, while trying to determine as many constant sets as possible.

Revised schedule

Week	Tasks	
4	Finish set re-use	(**Adam**, **Andrey**)
	Begin purity checking	(**Adam**, **Andrey**)
5	Finish purity checking	(**Adam**, **Andrey**)
	Begin benchmark suite	(**Adam**, **Andrey**)
6	Finish benchmark suite; fix bugs; reporting	(**Adam**, **Andrey**)

Resources needed

We currently posses all the resources that we can think of. For the benchmarking we will be looking at the number of dynamic instructions run by and the memory usage of several {Set1y} programs and we have the tools necessary for both. We do not have any special hardware requirements.