Types, Languages, and Compilers (Spring 2023)
Problem Set #3

Raphael Sofaer - `rjs2247`

9 April 2023

1 Dependently Typed Language

For my final project, I intend to build a dependently typed language able to express graph structures restricted by dependent type relations.

Richly annotated graph structures are commonly encountered in many fields, including program analysis, networking, social media, etc. When constructing these graphs and algorithms which use them, the user must maintain and understand invariants. For instance, a graph of social media data may have both users and organized groups represented as nodes. However, some relations (perhaps 'in a relationship'?) may be restricted to one kind of node or another. By encoding these restrictions in a type system, we can ensure invariants are maintained throughout the program.

To do this I intend to begin by working through the `pi-forall` tutorial recommended by John Hui. The completion of the optional Σ types will be important for accumulating many terms. Then I intend to implement a familiar record type, then the dependent type capabilities.

In addition to the base challenge of having a recursive dependently typed record type, the language must be able to express modifications to graphs, or constructions of new graphs from old graphs, in a way that allows avoiding the creation of interim graphs that cannot be typed.

For instance, if a directed graph $G = (V, E)$ is restricted by the dependent type that no leaf nodes exist, the programmer should be able to add a pair of nodes a, b on to the existing graph such that for some node $n \in V$ $\{(n, a), (a, b), (b, n)\} \subset E$.

This project is driven by recent frustrations dealing with a rich graph format in python, and I look forward to working through it.

References

`pi-forall`: https://arxiv.org/pdf/2207.02129.pdf
`Dependently Typed Knowledge Graphs`: https://arxiv.org/abs/2003.03785