HW3

From System F to Typed Assembly Language

March 13, 2019

1 Short Summary

The contribution of the paper is two-fold. The first one is the design of a typed assembly language (TAL) and the second is type-preserving translation from system F to TAL. The key contribution as specified by the authors is the approach to polymorphic closure conversion.

2 Strengths

1. The requirement of compiler which preserves type information is well motivated. Since the paper is from 1999, I am assuming this was a fresh contribution. The authors mention previous work where 80 percent type info is preserved at best.

2. The authors also take great effort in motivating the need of TAL because this way type checking can be performed independent of the original source code and the compiler. I think this is a very cool idea especially since its almost 2 decades old.

3. Since TAL is typed, it makes sure that well-typed programs do not become stuck. (Yes that is the goal for any typed language) however, here instead of performing some external checks, which the system might have to provide TAL manages all of that.

4. The fact that all these stages are independent of each other is pretty cool.

3 Weakness

1. Using types by introducing annotations in TAL, does check the type errors, however, it does not check for other errors, for eg: logical errors.

2. Though they mention the work from which they derived the order of the steps, I am quite interested in knowing how they concluded that this order is optimal.

3. Had the paper not been already discussed in the lectures, it would have been even more difficult to follow.

4. They don't explain the optimization proofs which they incorporate in the implementation of the compiler. They mention two techniques in CPS section that they eliminate tail recursion and administrative redices and provide the final outcome on the next page.

4 General

1. After reading the paper, it seems it derives a lot of ideas from previous work, the individual translations, the order of the translations, the optimization. However, I am assuming this is what every basic compiler contains, their key contributions were on providing a typed assembly language and a compiler which I think is to support the changing hardware during the 1999 period.

5 Questions:

1. So nowadays are the assembly languages typed or untyped?

2. Won't having assembly typed languages increase the computation time? Isn't that the reason why people prefer untyped languages at a higher level, so we can have a similar argument for assembly level.

3. Can I change the order of the translations?- Yes. Even if I can, will it be helpful? They have provided a calculi at each stage, which I think is more of an implementation detail, which means these steps are independent of each other.