Representing Object-Oriented Languages:

A study of various approaches & A Haskell interpreter for one of them

Shobhit Maheshwari<shmahesh@ucsc.edu>
Aneesh Neelam<aneelam@ucsc.edu>

Representing Procedural and Functional Languages

Lambda Calculus: Functions as primitives.

Trivial to write an interpreter for that, as we've seen in class.

But can we use that to represent Object-Oriented Languages?

Objects need to be primitives right?

Approaches for representing Object-Oriented Languages

Extensive research has already been done in this area:

1. Represent Objects as Functions?
2. Adapt Lambda Calculus to support Objects?
3. A whole new Calculus that complements Lambda Calculus?

A new Calculus? That’s the approach that we liked the most.

Object Calculus

Devised by M. Abadi and L. Cardelli in the 90s.

Designed to be minimal. Typed and untyped Object Calculus devised.

Untyped Object Calculus consists of:

- Objects (*obviously, they're primitives here*)
- Method Invocation
- Method Update
- Object Cloning
- Local Variable Definitions

Haskell Implementation

- Object Calculus implemented as a Haskell data type.
- Objects implemented as a Map from Labels to Methods (Data.Map of Haskell)
- Labels are Strings, distinct within an Object.
- Methods defined as two types: Fields and Function Bodies.
  - Field refers to an object's field (Variable).
  - Function Body is a conventional function that can be evaluated.
- Method Invocation and Method Update are implemented as Haskell functions.
- Evaluation of Object Calculus makes use of the above two functions.

Conclusion

Object Calculus is very vast. Still a long way to go to get up to speed.

A lot of derived research has been done.

Understanding the typing rules and operational semantics takes time.

Haskell Interpreter implemented only for untyped Object Calculus.

Haskell is a typed functional language.

Implementing Object Calculus in Haskell feels odd.

References

1. M. Abadi and L. Cardelli. A Theory of Objects. Springer-Verlag, 1996.

2. M. Abadi and L. Cardelli. An Imperative Object Calculus Basic Typing and Soundness, 1995.

3. Benjamin C. Pierce and David N. Turner. Simple Type-Theoretic Foundations for Object-Oriented Programming. Journal of Functional Programming, 4(2):207-247, April 1994.

4. G. Castagna. Object-oriented programming. Boston: Birkhauser; 1997.

5. K. B. Bruce. Foundations of Object-oriented Languages. Cambridge Mass.: MIT Press; 2002.

6. Raul Rojas. A Tutorial Introduction to the Lambda Calculus, 2015.