Type Inference for Modal Type System

Yiluo Wei (u6227375)

Supervisor: Ranald Clouston

COMP3740 - Project Work in Computing

March 26, 2019
Overview

**Type Inference for Modal type system**

- What is type inference?
- What is modal logic?
- What does modal logic have to do with type system?
- Why does it matter?
Type Inference

- To infer the type for an expression without the expression being explicitly tagged with type annotations
- Widely used in (functional) programming language. (e.g., Haskell, ML)
Type Inference

\((\lambda x. x == 1) \ n\)

1. \(x\) has type \(A\) (a type variable, unknown for now)
2. \(n\) has type \(B\) (a type variable, unknown for now)
3. ‘== 1‘ (as a function) has type \(Int \to Bool\)
4. \(x\) is applied to ‘== 1‘, we can solve for \(A = Int\)
5. The lambda term has type \(Int \to Bool\)
6. \(n\) is applied to the lambda term that has type \(Int \to Bool\)
7. We can solve for \(B = Int\), and the expression has type \(Bool\)
Modal Logic

- It is necessary that $A$
- $A$ always holds
- An agent believes $A$
- ... Different modal logics, different meanings
Proposition as Types

\[
\begin{align*}
\text{Application} & \quad \frac{f : A \rightarrow, \ B \ x : A}{f \ x : B} \\
& \quad \rightarrow \quad \frac{A \rightarrow B, \ A}{B}
\end{align*}
\]
Proposition as Types

- Intuitionistic second order logic: Coq
- Linear logic: linear type system for resource allocation
- Modal logic: monad, homotopy type, staged programming, etc.
Goal and Approach

Implement a program that infers type for lambda calculus with modalities (modal types).

1. Extend lambda calculus with modality notions.
2. Implement an interpreter that infers type as well as evaluates the language.
3. Adding more advanced features to the language (e.g., sum types, more complicated modalities)
End