Theorems for Free

In simple and dependent type theory

Outline

- What are Theorems for free?
- What do they look like in simple polymorphic type systems?
- How are they better in dependent type systems?
- Key takeaway: “Parametric polymorphism is good and you should write polymorphic definitions”

What are Theorems for free?

- A “metatheorem” about a type system saying “if you can’t observe types at runtime, then polymorphic functions behave nicely”
- “Behave nicely” has a formal definition in the theory one uses to talk about the language
- You can build derived semantics of your language and interpret programs as theorems

In practice?

1. Write a polymorphic function (f : forall a, …)
2. Think of types as sets
3. Think of types as relations
4. Read off the theorem
5. ???
6. Profit

Actually in practice?

- Consider (f : forall a, a -> a)
- “Semantically a function taking a set, an element in it, and returning an element”
- “Relationally, an implication taking a relation, two related elements, and proving the outputs are related”
- Constructing different relations gives different theorems

In dependent type theory

- With universes, “relations” are a thing!
- The syntax model has everything we need
- Implementable as a syntactic translation (“[_] : Term -> Term”)
- More principled translation of inductive types
- Metaprogramming!

Example

- Consider (f : (A : Type), A -> A)
- Translate
  [f] : (A A’ : Type) (R : A -> A’ -> Type)
  (a : A) (a’ : A’) (r : R a a’),
  R (f A a) (f A’ a’)
- Instantiate
  λ (A : Type) (a : A) ->
  [f] A A (λ x y -> x = a) a a refl
  : (A : Type) (a : A), f a = a

Applications

- Parlor tricks
- “Proof transfer” (trocq)
- Automatically generating structure and axioms of algebraic gadgets (my PhD)

Resources

- Wadler, “Theorems for Free!”, 1989 link
- Bernardy, Jansson, Paterson, “Proofs for free, Parametricity for dependent types”, 2012 link
- Tassi et al., “coq-elpi” implementation in “app/derive/” link
- Cohen, Crance, Mahboubi, “Trocq: Proof Transfer for Free”, 2025 link