Problem Set 4

Released: February 13, 2023
Due: February 24, 2023, 11:59pm
Last modified: Feb 16, 2023, 11pm

Modifications:

- (Problem 2 part 3) Change part 3 of problem 2 from actions of strings to actions of integers. The previous version was wrong for a subtle reason. The updated version is analogous to the original but correct. For one bonus point, you can figure out what the correct version of the original should be.
- (Problem 3 part 1) Fix the formulation of Lawvere's fixed point theorem in simple type theory.

Submit your solutions to this homework on Canvas in a group of 2 or 3. Your solutions must be submitted in pdf produced using LaTeX.

Problem 1 Product Functor

Let $\mathcal{C}$ be a category with all binary products. We will use the notation $\mathcal{C}^2$ for the product category $\mathcal{C} \times \mathcal{C}$ to avoid confusion between the two notions of product below.

- Show that taking binary products defines a functor $\times : \mathcal{C}^2 \to \mathcal{C}$. That is, show that if we define $\times$ on objects such that $a \times b$ is a product of a and b (with projections $\pi_1 : a \times b \to a$ and $\pi_2 : a \times b \to b$), then you can extend the definition to a functorial action on morphisms.
- Let $\Pi_1 : \mathcal{C}^2 \to \mathcal{C}$ be the functor that projects out the first component of $\mathcal{C}^2$. Prove that π_1 defines a natural transformation from $\times$ to Π_1. Symmetrically, π_2 is also natural.

. . . . . .

Problem 2 Theorems for Free, Naturally

The naturality property of a natural transformation is a very strong property. So strong in fact that for specific functors we can characterize what all of the natural transformations between them are.

Suprisingly this has direct applications to programming languages. The reason is that in a pure polymorphic functional language given type constructors F and G that are functorial, all functions $F(X) \to G(X)$ that are polymorphic/generic in X are natural transformations! Phil Wadler, building on John Reynold's theory of parametricity called these "theorems for free": just from the type of a polymorphic function, the naturality property[1] gives you properties that hold for every function of that type (Reynolds [1983], Wadler [1989]).

We will focus on a very simple example: pick F and G to be the identity functor on types. Then what are the functions of type $\forall X.X \to X$? Obviously the identity function has this type but are there any other examples? If the language is *pure* meaning no side-effects, then it can be proven that the function must be equivalent to the identity function. In effectful languages we need to weaken this statement. For instance in a language where functions might crash but have no other side-effects, the only functions of type $\forall X.X \to X$ are the identity function and the function that always crashes regardless of its input.

Your task is to prove these "free theorems" are true of natural transformations in different categories that model functional languages with different effects.

1. First, prove that the only natural transformation from $\mathrm{id}_{\mathrm{Set}}$ to $\mathrm{id}_{\mathrm{Set}}$ is the identity transformation.

2. Define the category Par as follows:

   - Objects are (small) sets
   - A morphism from X to Y is a *partial* function, i.e., a pair of a subset $D \subseteq X$ and a function $f : D \to Y$.

     The identity partial function id_X is the identity function with $D = X$. To compose $(D, f) : X \to Y$ and $(E, g) : Y \to Z$, the domain is $\{d \in D \mid f(d) \in E\}$ and the function is $g \circ f$

   This category models programs that may crash.

   Prove that the only natural transformations from $\mathrm{id}_{\mathrm{Par}}$ to $\mathrm{id}_{\mathrm{Par}}$ are the identity transformation and the transformation that is everywhere undefined, i.e., has empty domain.

3. Define the category $\mathbb{Z}$-Action to be the following category:

   - Objects are pairs of a set X and a monoid action of the monoid $\mathbb{Z}$ with operation $+$ and unit 0 on X.

[1] more generally, dinaturality or more generally still, parametricity

- A morphism from $X, \otimes$ to $Y, \oplus$ is a function $f : X \to Y$ that is *equivariant*: for all $x \in X$ and $n \in \mathbb{Z}$, $f(n \otimes x) = n \oplus f(x)$.

This category can be used to model programs that can *increment* or *decrement* a counter, but perform no other side-effects

Prove that for any natural transformations $\alpha : \mathrm{id}_{\mathbb{Z}\text{-Action}} \to \mathrm{id}_{\mathbb{Z}\text{-Action}}$, there is a number $n \in \mathbb{Z}$ such that

$$\alpha^{(X,\otimes)}(x) = n \otimes x$$

4. **BONUS**: For one bonus points. Fix a set $\mathcal{A}$ called the "alphabet". Define $\mathcal{S}$ to be the free monoid on $\mathcal{A}$, i.e., the monoid of "strings". Define $\mathcal{S}$-Action to be the category whose objects are actions of $\mathcal{S}$ and morphisms are equivariant maps.

   Provide an analogous concrete description of the natural transformations $\mathrm{id}_{\mathcal{S}\text{-Action}} \to \mathrm{id}_{\mathcal{S}\text{-Action}}$.

. . . . . .

Problem 3 Lawvere's Fixed Point Theorem and Fixed-point Combinators

There are various "diagonalization arguments" employed in logic, set theory and computer science such as the proofs of Gödel's incompleteness theorem, as well as Cantor's Turing's and Rice's theorems. These diagonalization arguments typically show something is impossible to construct.

F. William Lawvere showed that we can abstract over the reasoning in these different proofs by proving a *fixed point* theorem that is valid in an arbitrary cartesian closed category[2]. We will prove the following variant of Lawvere's original formulation:

Theorem 1 (Lawvere's Fixed Point Theorem)**.** *Let x, d be objects in a cartesian closed category $\mathcal{C}$. If there is a pair of morphisms $s : \mathcal{C}(d^x, x)$ and $r : \mathcal{C}(x, d^x)$ such that $r \circ s = id_{d^x}$, then there is a morphism $fix : d^d \to d$ that is a* fixed point combinator *in that for any $f : a \to d^d$,*

$$fix \circ f = app \circ (f, fix \circ f)$$

1. Prove Lawvere's fixed point theorem. You can either do this using cartesian closed categories or perform the construction in simple type theory.

   To prove the theorem in type theory, work in simple type theory with the only connective being function types, with two base types X and D, two function symbols $s : (X \Rightarrow D) \to X$ and $r : X \to (X \Rightarrow D)$ and one equation $g : (X \Rightarrow$

[2] a cartesian closed category is a category with all finite products and exponentials

$D) \vdash r(s(g)) = g : X \Rightarrow D$. Then construct a term $\cdot \vdash \text{fix} : (D \Rightarrow D) \Rightarrow D$ and prove that

$$f : D \Rightarrow D \vdash \text{fix}\, f = f(\text{fix}\, f) : D$$

2. Use Lawvere's fixed point theorem to prove Cantor's theorem as a corollary:

   Corollary 1 (Cantor's Theorem)**.** *There is no surjective function from X to its powerset $\mathcal{P}(X)$.*

   Hint: the powerset of X can be defined as 2^X where $2 = \{\text{true}, \text{false}\}$.

However, while most classical applications of this theorem are for deriving contradictions, we will see later in class that this construction can be used to give semantics to recursive functions.

. . . . . .

References

John C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83, Proceedings of the IFIP 9th World Computer Congress, Paris, France, September 19-23, 1983*, pages 513–523. North-Holland/IFIP, 1983.

Philip Wadler. Theorems for free! In Joseph E. Stoy, editor, *Proceedings of the fourth international conference on Functional programming languages and computer architecture, FPCA 1989, London, UK, September 11-13, 1989*, pages 347–359. ACM, 1989. doi: 10.1145/99370.99404. URL `https://doi.org/10.1145/99370.99404`.