niche computing science

Research Blog of 穆信成 Shin-Cheng Mu

TAG ARCHIVES: AGDA

No Inverses for Injective but Non-Surjective Functions?

"I cannot prove that if a function is injective, it has an inverse," Hideki Hashimoto posed this question to me a while ago. It turned out that this was not the property he really wanted, but it got me into thinking: is it possible at all?

Preliminaries

Let us start with some basic definitions. A relation from A to B is denoted by the wavy arrow:

```
_↝_ : Set → Set → Set1
A ↝ B = A → B → Set
```

Given a relation `R : A ↝ B` we can always take its converse `R ˘ : B ↝ A`, and a function can be lifted to a relation by fun:

```
_˘ : ∀ {A B} → (A ↝ B) → (B ↝ A)
(R ˘) b a = R a b

fun : ∀ {A B} → (A → B) → (A ↝ B)
fun f a b = f a ≡ b
```

A relation `R : A → B` is *simple* if it does not map one input to multiple outputs. It is *entire* if everything in `A` is mapped to something in `B` — a more familiar word may be "total".

```
simple : ∀ {A B} → (A → B) → Set
simple R = ∀ {a b₁ b₂} →  R a b₁ → R a b₂ → b₁ ≡ b₂

entire : ∀ {A B} → (A → B) → Set
entire R = ∀ a → ∃ (λ b → R a b)
```

A function is a relation that is simple and entire. Indeed, one can show that `fun f` is simple and entire for every `f`. Injectivity and surjectivity are similar notions defined for converse of `R`:

```
injective : ∀ {A B} → (A → B) → Set
injective R = ∀ {a₁ a₂ b} → R a₁ b → R a₂ b → a₁ ≡ a₂

surjective : ∀ {A B} → (A → B) → Set
surjective R = ∀ b → ∃ (λ a → R a b)
```

The (constructive variant of the) axiom of choice states that an entire relation `A → B` can be refined to a function `A → B`:

```
ac : ∀ {A B} → (R : A → B) →
         (∀ a → ∃ (λ b → R a b)) → ∃ {A → B} (λ f → ∀ a → R a (f a))
ac R R-entire = ((λ a → proj₁ (R-entire a)) , λ a → proj₂ (R-entire a))
```

See the axiom of choice homepage, or the Stanford Encyclopedia of Philosophy for more information on this axiom.

Inverting Injective and Surjective Functions

Now, let us restate Hashimoto san's challenge:

> *Let fun `f : A → B` be injective. Prove that `f` has a left inverse. That is, some `f⁻¹` such that `f⁻¹ (f a) = a` forall `a`.*

It turned out that he forgot a condition: `f` is also surjective. If `f` is also (provably) surjective, one can pick some `g ⊆ f˘` using the axiom of choice (since `f` is surjective if and only if `f˘` is total) and further prove that `g ∘ f = id` using injectivity:

```
inv-sur : ∀ {A B} → (f : A → B) →
               injective (fun f) → surjective (fun f) →
                  ∃ {B → A} (λ f⁻¹ → (∀ a → f⁻¹ (f a) ≡ a))
inv-sur f f-inj f-sur with ac ((fun f) ˘) f-sur
... | (g , fgb≡b) =  (g , λ a → f-inj {g (f a)} {a} {f a} (fgb≡b (f a)) refl)
```

Like the proof of the constructive axiom of choice, the proof above does not really do much. The proof of surjectivity of `f` has already provided, for every `b : B`, an `a : A` such that `f a ≡ b`. So we simply let `f⁻¹` return that `a`.

Can we lift the restriction that `f` must be surjective? That is, can this be proved?

```
inv : ∀ {A B} → (f : A → B) → injective (fun f) →
                  ∃ {B → A} (λ f⁻¹ → (∀ a → f⁻¹ (f a) ≡ a))
```

To make the scenario clear: we have a (total) function `f : A → B` that is injective but not necessarily surjective. The set `B` could be "larger" than `A` in the sense that there could be some elements `b : B` for which no `f a` equals `b` — that is, `B` may not be "fully covered." Can we construct `f⁻¹ : B → A` such that `f⁻¹ (f a) ≡ a` for all `a : A`?

At the first glance it did not look like something terribly difficult to do. Given `b`, if it equals some `f a`, let `f⁻¹` simply return that `a`. Otherwise `f⁻¹` could just map `b` to any element in `A`, since this `b` is not used in any invocation of `f⁻¹ (f a)` anyway. It should be possible as long as `A` is not empty, right?

I tried to construct this f^{-1} but could not help noticing something funny going on. It turns out that had this function existed, we could, again, prove the law of excluded middle. That is, for any predicate `P : B → Set` and any `b : B`, there would be a decision procedure telling us whether `P b` is true or not.

Provided that we assume proof irrelevance, that is.

Excluded Middle, Again

Here we go. Let `B` be some type having decidable equality. That is, there exists some `eqB`:

```
eqB : (b₁ b₂ : B) → (b₁ ≡ b₂) ⊎ (¬ (b₁ ≡ b₂))
```

where `_⊎_` is disjoint sum.

Now take some predicate `P : B → Set`. Let `A` be defined by:

```
A : (B → Set) → Set
A P = Σ B (λ b → P b)
```

That is, `A P` is the subset of `B` for which `P` holds. Each element of `A P` is a pair `(b, Pb)` where `Pb` is a proof of `P b`.

Finally, take

```
f : ∀ {P} → A P → B
f = proj₁
```

Thus `f (b, Pb) = b`.

The function `f` is injective *if we assume proof irrelevance*. That is, if `f (b, Pb) = b` and `f (b', Pb') = b`, we must have `b = b'` and (due to proof irrelevance) `Pb = Pb'`, and therefore `(b, Pb) = (b', Pb')`. Indeed, if we postulate proof irrelevance:

```
postulate irr : (P : B → Set) → ∀ {b} → (p₁ : P b) → (p₂ : P b) → p₁ ≡ p₂
```

We can construct a proof that `f` is injective:

```
f-inj : ∀ {P} → injective (fun (f {P}))
f-inj {P} {(.b , Pb₁)} {(.b , Pb₂)} {b} refl refl = cong (λ p → (b , p)) (irr P Pb₁
 Pb₂)
```

Assume that we have proved `inv`. We can now apply `inv` and obtain some `f⁻¹`, the left inverse of `f`.

However, with this particular choice of `A`, `f`, and `f⁻¹`, we can construct a deciding procedure for `P`. That is, for any `P` and `b`, we can determine `P b` holds or not:

```
em : {P : B → Set} → ∀ b → P b ⊎ ¬ (P b)
```

This is how `em` works. Given some `b`, let's apply `f⁻¹` to `b`. The result is a pair `(b', Pb')`. Let's compare `b` and `b'` using `eqB`:

```
em {P} b with inv f (f-inj {P})
...      | (f⁻¹ , f⁻¹fa≡a) with inspect (f⁻¹ b)
...                        | (b' , Pb') with-≡ _          with eqB b b'
```

If `b ≡ b'`, `Pb'` is a proof of `P b'` and also a proof of `P b`. Let us just return it (after some type casting):

```
em {P} b | (f⁻¹ , f⁻¹fa≡a) | (b' , Pb') with-≡ _          | inj₁ b≡b'  =
              inj₁ (subst P (sym b≡b') Pb')
```

Consider the case that `b` does not equal `b'`. We want to show that `P b` is not true. That is, a proof of `P b` leads to contradiction. Assume we have a proof `Pb` of `P b`. Since `f (b , P b) ≡ b`, we have `f⁻¹ b ≡ (b , Pb)`:

```
em {P} b | (f⁻¹ , f⁻¹fa≡a) | (b' , Pb') with-≡ b'Pb'≡f⁻¹b | inj₂ ¬b≡b' =
   inj₂ (λ Pb →
           let f⁻¹b≡bPb : f⁻¹ b ≡ (b , Pb)
               f⁻¹b≡bPb = f⁻¹fa≡a (b , Pb)
```

The assumption says that `f⁻¹ b = (b' , Pb')`. By transitivity we have `(b , Pb) ≡ (b' , Pb')`.

```
               bPb≡b'Pb' : (b , Pb) ≡ (b' , Pb')
               bPb≡b'Pb' = sym (trans b'Pb'≡f⁻¹b f⁻¹b≡bPb)
```

But if we take the first component of both sides, we get `b ≡ b'`. That contradicts our assumption that `b` does not equal `b'`:

```
               b≡b' : b ≡ b'
               b≡b' = cong proj₁ bPb≡b'Pb'
           in ¬b≡b' b≡b')
```

which completes the proof.

In retrospect, `f⁻¹` cannot exist because for it to work, it has to magically know whether `b` is in the range of `f`, that is, whether `P b` is true or not.

Nakano's Challenge

When I talked about this to Keisuke Nakano he posed me another related challenge. Set-theoretically, we understand that if there exists an injective function `f : A → B` and another injective function `g : B → A`, the sets `A` and `B` are of the same cardinality and there ought to be a bijection `A → B`. Can we construct this bijection? That is, can we prove this theorem?

```
nakano : {A B : Set} →
           (f : A → B) → injective (fun f) →
           (g : B → A) → injective (fun g) →
             ∃ {A → B} (λ h → injective (fun h) × surjective (fun h))
```

I do not yet have a solution. Any one wanna give it a try?

Programs

- Hashimoto.agda

This entry was posted in Research Blog and tagged Agda, Dependent Type, Program Inversion on May 11, 2009 [http://www.iis.sinica.edu.tw/~scm/2009/no-inverses-for-injective-but-non-surjective-functions/] .

Proof Irrelevance, Extensional Equality, and the Excluded Middle

It was perhaps in our first lesson in constructive logic when we learnt about the absence of the law of excluded middle, which in a constructive interpretation would imply a decision procedure for every proposition. Therefore I was puzzled by the fact, stated in a number of places including the Stanford Encyclopedia of Philosophy (SEP), that axiom of choice, proof irrelevance, and extensional equality (definitions to be given later) together entail the law of excluded middle. Since a constructive variant of axiom of choice is provable in intuitionistic logic, and both proof irrelevance and extensional equality are properties we would like to have in a type system, it is worrying that they lead to implausible consequences. Thus I was curious to find out what happened.

The observational type theory promises us the best of everything — extensional equality without losing canonicity (please do implement it in Agda soon!), and it does rely on proof irrelevance. There is even an Agda embedding (or, with the information given it is not hard to reconstruct one), so I conducted some experiments in the embedding. For this post, however, it is sufficient to do everything in plain Agda and postulate the missing properties.

Decidable Equality for All Types?

The construction in SEP makes use of functions on propositions, which is not available in the Agda embedding of observational type theory. Instead I experimented with another construction from Benjamin Werner's On the Strength of Proof-Irrelevant Type Theories: with (a particular interpretations of) axiom of choice and proof irrelevance, one can construct a function that, given *a* and *b* of any type `A`, decides whether *a* equals *b*.

This could also lead to horrifying consequences — we could even compare whether two infinite structure, or two functions are equal, in a finite amount of time.

Axiom of Choice

The axiom of choice, as described on the very informative homepage for the axiom maintained by Eric Schechter, is considered by some the "last great controversy of mathematics." The axiom comes in many forms, and one of the simplest could be:

> *Given a collection of non-empty sets, we can choose a member from each set in that collection.*

A set of `B` is usually represented by a characteristic function `B → Set`. Let the collection of non-empty sets of `B`'s be indexed by `A`, the collection can be modelled by a function mapping indexes in `A` to sets of `B`s, that is, a relation `A → B → Set`. The action of "choosing" is modelled by the existence of a function returning the choice, that is, a function taking an index in `A` and returning a element in `B` that is chosen. One possible formulation of the axiom of choice would thus be:

```
ac : {A B : Set} → (R : A → B → Set) →
       (∀ x → ∃ (λ y → R x y)) →
         (∃ {A → B} (λ f → ∀ x → R x (f x)))
```

In words: given a collection of sets represented by `A → B → Set`, if `R x` is non-empty for every `x : A`, there exists a function `f : A → B`, such that `f x` is in `R x` for every `x : A`. Another way to see it is that the axiom simply says we can refine a relation `R : A → B → Set` to a function `A → B` provided that the former is total.

It is surprising to some that in constructive logic, the axiom of choice is in fact provable: