Appendix A

Example Proof

Herein we explain in detail all the tactics used during the proof of preservation of isomorphisms by a functor. This proof is light but is illustrative of the type of results that we obtain in this work.

We have to show that given two categories `c` and `d` and a functor `fF:(Functor c d)`, if `c1`,`c2`$\in$ `c` are isomorphic then `(fF c1)`,`(fF c2)`$\in$ `d` are isomorphic. In `Coq` this is simply stated by the lemma `F_Preserve_Iso`.

```
Coq < Lemma F_Preserve_Iso :
        (c,d:Category)(fF:(Functor c d))
        (c1,c2:c)(Iso c1 c2)→(Iso (fF c1) (fF c2)).

1 subgoal

  ============================
   (c,d:Category)
    (fF:(Functor c d))
     (c1,c2:(c))(Iso c1 c2)→(Iso (F0 fF c1) (F0 fF c2))
```

We switch to the proof editing mode by declaring the lemma `F_Preserve_Iso`. Here we give as input a command that begins after the symbol `<` and finishes with a dot. After, the `Coq` system output shows up. Remark that in editing proof mode the prompt `Coq <` is changed to `F_Preserve_Iso <`.

Our goal is now to prove

```
(c,d:Category)(fF:(Functor c d))
(c1,c2:(c))(Iso c1 c2)→(Iso (F0 fF c1) (F0 fF c2)).
```

We start by using the tactic `Intros`. This tactic introduces as hypothesis the variables `c`, `d`, `fF`, `c1`, `c2` and `H`. To these hypothesis we call the local context of the goal.

```
F_Preserve_Iso < Intros c d fF c1 c2 H.
```

```
1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  ============================
   (Iso (F0 fF c1) (F0 fF c2))
```

Now our goal is to prove `(Iso (F0 fF c1) (F0 fF c2))`. The hypothesis `H` states that `c1` and `c2` are isomorphic. Thus there is a morphism `f:(Hom c1 c2)` and `g:(Hom c2 c1)` such that one is the inverse of the other. We can refer directly to these two morphisms using the projections `(IsoMor H)` and `(InvIso H)`. However, for the sake of simplicity, we would like to have them in the local context. That can be achieved with the tactic `Elim`. This tactic must be applied to an inductive type destroying it into its constituents. In our particular case, by applying `Elim` to `H`, we obtain:

- a morphism `IsoMor:(Hom c1 c2)`;
- a morphism `InvIso:(Hom c2 c1)`;
- a proof that `IsoMor` is the inverse of `InvIso`.

```
F_Preserve_Iso < Elim H.

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  ============================
   (IsoMor:(Hom c1 c2))
    (InvIso:(Hom c2 c1))
     (IsoLaw IsoMor InvIso)→(Iso (F0 fF c1) (F0 fF c2))
```

We call `f`, `g` and `H0` to `IsoMor`, `InvIso` and the proof that one is the inverse of the other, respectively.

```
F_Preserve_Iso < Intros f g H0.

1 subgoal
```

```
c :  Category
d :  Category
fF : (Functor c d)
c1 :  (c)
c2 :  (c)
H : (Iso c1 c2)
f :  (Hom c1 c2)
g :  (Hom c2 c1)
H0 :  (IsoLaw f g)
============================
 (Iso (F0 fF c1) (F0 fF c2))
```

Now we are able to give the candidates for the morphisms between `(F0 fF c1)` and `(F0 fF c2)`. Clearly these morphisms are `(FMor fF f)` and `(FMor fF g)`. How can we provide this information to the Coq system? Note that `Build_Iso` has type

```
(IsoMor:(Hom c1 c2))(InvIso:(Hom c2 c1))
(IsoLaw IsoMor InvIso)->(Iso c c1 c2).
```

To show `(Iso (F0 fF c1) (F0 fF c2))` is sufficient to give:

- `IsoMor: (Hom (F0 fF c1) (F0 fF c2))`;
- `InvIso: (Hom (F0 fF c2) (F0 fF c1))`;
- `(IsoLaw IsoMor InvIso)`.

This can be done using a variant of the tactic `Apply`. The tactic `Apply` matches the current goal against the conclusion of the type of `Build_Iso` and returns as many subgoals as the instantiations of its premises. So `Apply Build_Iso` matches

- c with d;
- c1 with `(F0 fF c1)`;
- c2 with `(F0 fF c2)`;

However in this case the Coq system can not find `IsoMor` and `InvIso`. Hence we use the variant `Apply ... with ...` that provides `Apply` with explicit instantiations for all premises of the type of `Build_Iso` which do not occur in the conclusion (and consequently can not be found). So `Apply Build_Iso` **with** `(FMor fF f) (FMor fF g)` also matches

- c with d;
- c1 with `(F0 fF c1)`;
- c2 with `(F0 fF c2)`;

and instantiate

- `IsoMor` with `(FMor fF f)`;
- `InvIso` with `(FMor fF g)`.

```
F_Preserve_Iso < Apply Build_Iso with (FMor fF f) (FMor fF g).

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  (IsoLaw f g)
  ============================
   (IsoLaw (FMor fF f) (FMor fF g))
```

Our goal becomes `(IsoLaw (FMor fF f) (FMor fF g))`, that is by definition

```
(InverseLaw (FMor fF f) (FMor fF g))
∧(InverseLaw (FMor fF g) (FMor fF f)).
```

To replace each occurrence of `IsoLaw` by its definition (δ-reduction) we use the tactic **`Unfold`**.

```
F_Preserve_Iso < Unfold IsoLaw.

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  (IsoLaw f g)
  ============================
   (InverseLaw (FMor fF f) (FMor fF g))
  ∧(InverseLaw (FMor fF g) (FMor fF f))
```

We do the same but now for `InverseLaw`.

```
F_Preserve_Iso < Unfold InverseLaw.

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  (IsoLaw f g)
  ============================
   (((FMor fF g) o (FMor fF f)) =%S (Id (F0 fF c2)))
   ∧(((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

In a variant of the tactic **`Unfold`** we refer to an hypothesis of the current goal to make the unfolding in that hypothesis.

```
F_Preserve_Iso < Unfold IsoLaw InverseLaw in H0.

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  ============================
   (((FMor fF g) o (FMor fF f)) =%S (Id (F0 fF c2)))
   ∧(((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

The tactic `Inversion` should be applied to an hypothesis of the current goal. When this hypothesis is a conjunction it splits this conjunction into two new hypothesis.

```
F_Preserve_Iso < Inversion H0.

1 subgoal
```

```
  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   (((FMor fF g) o (FMor fF f)) =%S (Id (F0 fF c2)))
   ∧(((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

The tactic `Split` is applied to the current goal typically in the case of conjunctions. It splits our goal into two subgoals.

```
F_Preserve_Iso < Split.

2 subgoals

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   (((FMor fF g) o (FMor fF f)) =%S (Id (F0 fF c2)))

subgoal 2 is:
 (((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

Now we have to show that `(FMor fF g)` is left and right inverse of `(FMor fF f)`. We start by showing that `(FMor fF g)` is left inverse of `(FMor fF f)`. We apply a transitivity that splits our goal into two subgoals.

```
F_Preserve_Iso < Apply Equal_trans with (FMor fF (g o f)).

3 subgoals
```

```
  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   (((FMor fF g) o (FMor fF f)) =%S (FMor fF (g o f)))

subgoal 2 is:
 ((FMor fF (g o f)) =%S (Id (F0 fF c2)))
subgoal 3 is:
 (((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

Our goal is simply the proof of preservation of the composition by a functor. Since it is written in the symmetric way we have first to apply symmetry to the goal and then apply the proof of preservation of the composition, by a functor `fF`, of the morphisms `g` and `f`.

```
F_Preserve_Iso < Apply Equal_sym; Apply (Prf_comp fF g f).

2 subgoals

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   ((FMor fF (g o f)) =%S (Id (F0 fF c2)))

subgoal 2 is:
 (((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

We apply again a transitivity that splits our goal into two new subgoals.

```
F_Preserve_Iso < Apply Equal_trans with (FMor fF (Id c2)).

3 subgoals

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   ((FMor fF (g o f)) =%S (FMor fF (Id c2)))

subgoal 2 is:
 ((FMor fF (Id c2)) =%S (Id (F0 fF c2)))
subgoal 3 is:
 (((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

Since `FMor` is a mapoid (remark that `FMor` is just `F1`) preserves the equality between morphisms. We only have to apply the proof of this preservation for the equality `H1`.

```
F_Preserve_Iso < Apply (Prf_pres (F1 fF c2 c2) H1).

2 subgoals

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   ((FMor fF (Id c2)) =%S (Id (F0 fF c2)))

subgoal 2 is:
 (((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

Our goal is simply the proof of preservation of the identity by a functor. So we only have to apply the proof of preservation of the identity, by a functor `fF`, in `c2`.

```
F_Preserve_Iso < Apply (Prf_id fF c2).

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   (((FMor fF f) o (FMor fF g)) =%S (Id (F0 fF c1)))
```

Our goal states now that `(FMor fF g)` is right inverse of `(FMor fF f)`. The proof from now on is similar to the left inverse done before.

```
F_Preserve_Iso < Apply Equal_trans with (FMor fF f o g).

2 subgoals

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   (((FMor fF f) o (FMor fF g)) =%S (FMor fF (f o g)))

subgoal 2 is:
 ((FMor fF (f o g)) =%S (Id (F0 fF c1)))
```

```
F_Preserve_Iso < Apply Equal_sym; Apply (Prf_comp fF f g).

1 subgoal

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   ((FMor fF (f o g)) =%S (Id (F0 fF c1)))


F_Preserve_Iso < Apply Equal_trans with (FMor fF (Id c1)).

2 subgoals

  c :  Category
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   ((FMor fF (f o g)) =%S (FMor fF (Id c1)))

subgoal 2 is:
 ((FMor fF (Id c1)) =%S (Id (F0 fF c1)))


F_Preserve_Iso < Apply (Prf_pres (F1 fF c1 c1) H2).

1 subgoal

  c :  Category
```

```
  d :  Category
  fF : (Functor c d)
  c1 :  (c)
  c2 :  (c)
  H : (Iso c1 c2)
  f :  (Hom c1 c2)
  g :  (Hom c2 c1)
  H0 :  ((g o f) =%S (Id c2))∧((f o g) =%S (Id c1))
  H1 :  ((g o f) =%S (Id c2))
  H2 :  ((f o g) =%S (Id c1))
  ============================
   ((FMor fF (Id c1)) =%S (Id (F0 fF c1)))


F_Preserve_Iso < Apply (Prf_id fF c1).

Subtree proved!
```

When the proof is complete the command `Qed` add to the environment the name of the original goal.

```
F_Preserve_Iso < Qed.

Intros c d fF c1 c2 H.
Elim H.
Intros f g H0.
Apply Build_Iso with (FMor fF f) (FMor fF g).
Unfold IsoLaw.
Unfold InverseLaw.
Unfold IsoLaw InverseLaw in H0.
Inversion H0.
Split.
Apply Equal_trans with (FMor fF g o f).
(Apply Equal_sym; Apply (Prf_comp fF g f)).
Apply Equal_trans with (FMor fF (Id c2)).
Apply (Prf_pres (F1 fF c2 c2) H1).
Apply (Prf_id fF c2).
Apply Equal_trans with (FMor fF f o g).
(Apply Equal_sym; Apply (Prf_comp fF f g)).
Apply Equal_trans with (FMor fF (Id c1)).
Apply (Prf_pres (F1 fF c1 c1) H2).
Apply (Prf_id fF c1).
F_Preserve_Iso is defined

Coq <
```