Chapter 1

Library Curry_Howard

```
Require Import Setoid.
```

First, let us start with a small definition of what it means for a "Set/Type" to be inhabited.

Let us also recall that we can hoist "Set/Type" to "Prop" (in some sense the very essence of Curry-Howard) with the help of Setoid library

```
Definition inhabited (A : Set) := ∃ a : A, True.

Example exists_nat :
  inhabited nat.
eexists; eauto.
eapply 1.
Qed.

Example exists_string :
  inhabited bool.
eexists; eauto.
eapply false.
Qed.

Lemma not_exists_false :
  ¬ inhabited False.
Proof.
  intros HC.
  inversion HC.
  eapply x.
Qed.
```

Here, we show how standard propositional logic holds for our inhabited definition

```
Lemma both_ways_split_inhabited : ∀ (P Q : Prop),
  inhabited (P ∧ Q) ↔
  inhabited P ∧ inhabited Q.
Proof.
  split; intros.
  - inversion H; destruct x; split; cbv; eauto.
  - destruct H. inversion H; inversion H0; cbv; eexists; eauto.
Qed.

Lemma either_way_split_inhabited : ∀ (P Q : Prop),
  inhabited (P ∨ Q) ↔
  inhabited P ∨ inhabited Q.
Proof.
  split; intros.
  - inversion H; destruct x; cbv; eauto.
  - destruct H; inversion H; cbv; eexists; eauto.
Qed.

Lemma not_exists_inhabited_not :
  ∀ (P : Prop),
  ¬ P ↔
  inhabited (~ P).
Proof.
  split; intros.
  - cbv; eauto.
  - inversion H; eauto.
Qed.

Example inhabitation_preservation :
  ∀ (P C : Prop),
  (P → C) →
  inhabited P →
  inhabited C.
Proof.
  intros. inversion H0.
  cbv.
  eexists; eauto.
Qed.

Example not_inhabitation_reverse :
```

```
  ¬ (
    ∀ (P C : Prop),
    (P → C) →
    inhabited C →
    inhabited P
  ).
Proof.
  intros HC.
  pose proof (HC False True).
  assert (False → True). eauto.
  assert (inhabited True). cbv. eauto.
  intuition.
  inversion H. eauto.
Qed.
```

Here is the big one, any inhabited prop must be true

```
Theorem curry_howard : ∀ (P : Prop),
  inhabited P ↔ P.
Proof.
  split; intros.
  - destruct H; eauto.
  - cbv; eauto.
Qed.
```

See how much this can simplify our proofs that proposition logic holds now that we have proven that inhabitation is equivalent to prop

```
Ltac CH :=
  split; repeat erewrite curry_howard; eauto.

Lemma both_ways_split_inhabited' : ∀ (P Q : Prop),
  inhabited (P ∧ Q) ↔
  inhabited P ∧ inhabited Q.
Proof.
  CH.
Qed.

Lemma either_way_split_inhabited' : ∀ (P Q : Prop),
  inhabited (P ∨ Q) ↔
  inhabited P ∨ inhabited Q.
Proof.
  CH.
```

```
Qed.

Lemma not_exists_inhabited_not' :
  ∀ (P : Prop),
  ¬ P ↔
  inhabited (~ P).
Proof.
  CH.
Qed.

Example inhabitation_preservation' :
  ∀ (P C : Prop),
  (P → C) →
  inhabited P →
  inhabited C.
Proof.
  intros; rewrite curry_howard in *; eauto.
Qed.
```