Illustrating Intuitionistic logic using Coq

J. L. Freire

January 9, 2005

Abstract

The Curry–Howard correspondence states the equivalence between the constructions implicit in intuitionistic logic and those described in the simply-typed lambda-calculus. It is an insight of great importance in theoretical computer science, and is fundamental in modern approaches to constructive type theory.

In this paper we try to illustrate this process using type theory.

We use some material from [2] and the compiler Coq [7]of the Calculus of Inductive Constructions. A good reference of Coq is [8].

1 Implicational Propositional logic

1.1 Definitions

Given an infinite set of *propositional variables* V we define the set of formulas as the least set containing V and such that: if A and B are formulas, then $A \to B$ is also a formula.

Let us define now a proof system called *minimal natural deduction*:

- A *judgement* or *sequent* $\Gamma \vdash A$ (Γ proves A) consists of a finite multiset[1] of formulas Γ and a formula A.
- An axiom is a judgement of the form $\Gamma,\ A \vdash A$.
- A proof of $\Gamma \vdash A$ is a tree of judgements such that:
  - The root label is $\Gamma \vdash A$
  - All the leaves are axioms The label of each mother node is obtained from the labels of its daughters using only rules from the following:

$$\frac{}{\Gamma, A \vdash A}\ \textbf{A}\text{ssumption}$$

$$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \Rightarrow B} \Rightarrow \textbf{I}\text{ntroduction}$$

$$\frac{\Gamma \vdash A \Rightarrow B \quad \Gamma \vdash A}{\Gamma \vdash B} \Rightarrow \text{Application or } \textbf{E}\text{limination}$$

[1] a set where repetitions are allowed and no account of order is taken

1.2 Examples

1.2.1

The judgement $\vdash (P \Rightarrow Q) \Rightarrow (Q \Rightarrow R) \Rightarrow P \Rightarrow R$ has the following proof:

$$\dfrac{\dfrac{P,\ P\Rightarrow Q,\ Q\Rightarrow R \vdash R}{P\Rightarrow Q,\ Q\Rightarrow R \vdash P\Rightarrow R}\Rightarrow I}{\dfrac{P\Rightarrow Q \vdash (Q\Rightarrow R)\Rightarrow P\Rightarrow R}{\vdash (P\Rightarrow Q)\Rightarrow (Q\Rightarrow R)\Rightarrow P\Rightarrow R}\Rightarrow I}\Rightarrow I$$

where

$$\dfrac{P,\ P\Rightarrow Q,\ Q\Rightarrow R \vdash Q\Rightarrow R \qquad \dfrac{P,\ P\Rightarrow Q,\ Q\Rightarrow R\vdash P \quad P,\ P\Rightarrow Q,\ Q\Rightarrow R\vdash P\Rightarrow Q}{P,\ P\Rightarrow Q,\ Q\Rightarrow R\vdash Q}\Rightarrow I}{P,\ P\Rightarrow Q,\ Q\Rightarrow R \vdash R}\Rightarrow E$$

Therefore, we say that $(P \Rightarrow Q) \Rightarrow (Q \Rightarrow R) \Rightarrow P \Rightarrow R$ is a theorem in this logic.

1.2.2

The formula $P \Rightarrow P$ is also a theorem.

$$\dfrac{P \vdash P}{\vdash P \Rightarrow P} \Rightarrow \mathbf{I}$$

1.2.3

Let us prove now $\vdash P \Rightarrow Q \Rightarrow P$

$$\dfrac{\dfrac{P,\ Q \vdash P}{P \vdash Q \Rightarrow P}\Rightarrow \mathbf{I}}{\vdash P \Rightarrow Q \Rightarrow P}\Rightarrow \mathbf{I}$$

1.3 The BKH interpretation

The rules of deduction can be seen as a formalization of the BHK[2] interpretation of the intuitionistic notion of "truth" of a formula which no longer relies on truth tables but in our ability to justify it via an explicit proof or "construction".

Because there exists a correspondence between proofs and terms of lambda-calculus, known as the Curry-Howard isomorphism [5], Coq stores proofs as terms of the calculus of inductive constructions. Those terms are called proof terms.

In the Figure 1 some "constructions" are builded up by using Coq

Unfortunately some formulas which are true in classical propositional logic, fails to be provable in this minimal natural deduction system of Implicational propositional logic . An example is the Peirce's law $((P \Rightarrow Q) \Rightarrow P) \Rightarrow P$. To explain why, lets take a look at the semantics.

[2] Brouwer–Heyting–Kolmogoroff

```
Section Min_prop_logic.
 Variables P Q R T : Prop.

 Theorem imp_trans :
 (P->Q)->(Q->R)->P->R.
 Proof.
  intros H H' p.
  apply H'.
  apply H.
  assumption.
  Show Tree.
  (*
  |- (P -> Q) -> (Q -> R)
               -> P -> R
 BY intros H H' pp
    H : P -> Q  H' : Q -> R
                pp : P |- R
   BY apply H'
     |- Q  BY apply H
       |- P  BY assumption
  *)
 Qed.

(* Print imp_trans.
 imp_trans =
 fun (H : P -> Q)
     (H' : Q -> R)
     (p : P) => H' (H p)
     : (P -> Q) -> (Q -> R)
                 -> P -> R
*)

Theorem ident:P->P.
Proof.
 intro p.
 assumption.
Qed.

(* Print ident.
ident = fun p : P => p
     : P -> P
*)
```

```
Theorem first:P->Q->P.
Proof.
 intro p.
 intro q.
 assumption.
Qed.

(* Print first.
first = fun (p : P)
            (_ : Q) => p
     : P -> Q -> P
*)

Goal (P->Q->R)->(P->Q)
               ->P->R.
Proof.
 intro uno.
 intro dos.
 intro p.
 apply uno.
 assumption.
 apply dos.                  (*)
 Show  Tree.
 assumption.
 Show Tree.
 (*
uno : P -> Q -> R |-
     (P -> Q) -> P -> R
   BY intro dos
      dos : P -> Q |- P -> R
     BY intro p
        p : P |- R
       BY apply uno
         |- P  BY assumption

         |- Q  BY apply dos
           |- P  BY assumption
*)


Save p.
End Min_prop_logic.
```

Figure 1: This is the Coq code

```
 Goal forall (P Q R : Prop),
(P->Q->R)->(P->Q) ->P->R.
Proof.
tauto.
(* Proof completed.*)

Show Tree.
(* |- forall P Q R : Prop,
 (P -> Q -> R) ->
     (P -> Q) -> P -> R  BY tauto
*)

Show Proof.
LOC:
Proof:
 fun
(H : P -> Q -> R)
(H0 : P -> Q)
(H1 : P) =>
let H2 := H H1 in
let H3 := H0 H1 in
let H4 := H2 H3 in H4
```

Let Γ = {P–>Q–>R, (P–>Q),P} the environmment. Then, following the proof structure we can obtain:

$$\dfrac{\dfrac{\Gamma \mid\!\!-\!\!- P\text{->}(Q\text{->}R) \quad \Gamma \mid\!\!-\!\!- P}{\Gamma \mid\!\!-\!\!- (Q\text{->}R)} \qquad \dfrac{\Gamma \mid\!\!-\!\!- P \quad \mid\!\!-\!\!- P\text{->}Q}{\Gamma \mid\!\!-\!\!- Q}}{\Gamma \mid\!\!-\!\!- R}$$

Figure 2: Showing `tauto` and the correspondence between the `Coq` proof and the proof in the natural deduction system.

A model[3] of this minimal natural deduction system is the following: let T a topological space and P, Q open sets. Let us define, as representation of $\Rightarrow$ the operation $P \multimap Q := Int(\overline{P} \cup Q)$ i.e. the interior of the complement of P union Q. Here the validity of a formula P corresponds with $v(P) = T$.

Let us take the topological space $\mathbb{R}$ of real numbers and a valuation v such that $v(P) = \mathbb{R} - \{0\}$ and $v(Q) = \emptyset$. Then $v(P \Rightarrow Q) = (v(P) \multimap v(Q)) = Int(\overline{\mathbb{R} - \{0\}} \cup \emptyset) = \emptyset$. Now, $v((P \Rightarrow Q) \Rightarrow P) = (\emptyset \multimap \mathbb{R} - \{0\}) = Int(\mathbb{R} \cup \mathbb{R} - \{0\}) = \mathbb{R}$.

Finally, $v(((P \Rightarrow Q) \Rightarrow P) \Rightarrow P) = \mathbb{R} \multimap \mathbb{R} - \{0\}) = Int(\emptyset \cup \mathbb{R} - \{0\}) = \mathbb{R} - \{0\} \neq \mathbb{R}$.

2 Intuitionistic Propositional Calculus

2.1 Definitions.

We extend the implicational logic with new formulas: a constant $\perp$ and if P and Q are formulas, then $P \vee Q$, $P \wedge Q$ are also formulas.

[3] The algebraic semantics of intuitionistic propositional logic is defined in a similar style as for classical logic replacing Boolean Algebras by Heyting Algebras. In particular, the family of open sets of a topological space is a Heyting Algebra. As the two valued Boolean algebra plays the role of test of truth in the classical case the real line topological space $\mathbb{R}$ do in the intuitionistic one in which there is not any finite model.

The symbol $\neg P$ abbreviates $P \Rightarrow \bot$.

In Coq we define inductive types to represent this new formulas:

```
Coq < Print False.
Inductive False : Prop :=

Coq < Print and.
Inductive and (A : Prop) (B : Prop) : Prop :=  conj : A -> B -> A /\ B
For conj: Arguments A, B are implicit
For and: Argument scopes are [type_scope type_scope]
For conj: Argument scopes are [type_scope type_scope _ _]

Coq < Print or.
Inductive or (A : Prop) (B : Prop) : Prop :=
    or_introl : A -> A \/ B | or_intror : B -> A \/ B
For or_introl: Argument A is implicit
For or_intror: Argument B is implicit
For or: Argument scopes are [type_scope type_scope]
For or_introl: Argument scopes are [type_scope type_scope _]
For or_intror: Argument scopes are [type_scope type_scope _]
```

The rules of natural deduction are those of the implicational case and:

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge\textbf{I}\text{ntroduction}$$

$$\frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash P} \wedge\textbf{E}\text{limination-l} \qquad \frac{\Gamma \vdash P \wedge Q}{\Gamma \vdash Q} \wedge\textbf{E}\text{limination-r}$$

$$\frac{\Gamma \vdash P}{\Gamma \vdash P \vee Q} \vee\textbf{I}\text{ntroduction-l} \qquad \frac{\Gamma \vdash Q}{\Gamma \vdash P \vee Q} \wedge\textbf{I}\text{ntroduction-r}$$

$$\frac{\Gamma, P \vdash R \quad \Gamma, Q \vdash R \quad \Gamma \vdash P \vee Q}{\Gamma \vdash R} \vee\textbf{E}\text{limination}$$

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash R} \bot\textbf{E}\text{limination}$$

In Coq

```
oq < Check conj.
conj
     : forall A B : Prop, A -> B -> A /\ B
```

```
Coq < Check and_ind.
and_ind
     : forall A B P : Prop, (A -> B -> P) -> A /\ B -> P

  Coq < Check or_introl.
or_introl
     : forall A B : Prop, A -> A \/ B

Coq < Check or_intror.
or_intror
     : forall A B : Prop, B -> A \/ B

Coq < Check False_ind.
False_ind
     : forall P : Prop, False -> P
```

2.2 Decision procedure

Based on [1], the Coq system provides a decision procedure for intuitionistic propositional tautologies called tauto [3].

```
Section prueba.
Variables a b c:Prop.
Goal ((b<->c)->(a/\b/\c)) /\ ((c<->a)->(a/\b/\c)) /\ ((a<->b)->(a/\b/\c))
-> (a/\b/\c).
tauto.
Save uno.

Goal ~ ~(~a \/ a).
try tauto.
Save dos.

Goal  ~ ~(~a \/ ~b \/ ~c \/ (a /\ b /\ c)).
try tauto.
Save tres.

Coq < Goal a \/ ~a.
1 subgoal

  a : Prop
  b : Prop
  c : Prop
  ============================
   a \/ ~ a

Unnamed_thm < Require Export Classical.

Unnamed_thm < Check NNPP.
NNPP
     : forall p : Prop, ~ ~ p -> p
```

```
Unnamed_thm < apply NNPP.
1 subgoal

  a : Prop
  b : Prop
  c : Prop
  ============================
   ~ ~ (a \/ ~ a)

Unnamed_thm < tauto.
Proof completed.

Unnamed_thm < Save cuatro.
apply NNPP.
tauto.
cuatro is defined

Coq < Goal ~(a /\ b) -> (~a \/ ~b).
1 subgoal

  a : Prop
  b : Prop
  c : Prop
  ============================
   ~ (a /\ b) -> ~ a \/ ~ b

Unnamed_thm < tauto.
Toplevel input, characters 409-414
> tauto.
> ^^^^^
User error: Tauto failed

Unnamed_thm < apply NNPP.
1 subgoal

  a : Prop
  b : Prop
  c : Prop
  ============================
   ~ ~ (~ (a /\ b) -> ~ a \/ ~ b)

Unnamed_thm < tauto.
Proof completed.

Unnamed_thm < Save cinco.
apply NNPP.
tauto.
cinco is defined
```

3 Glivenko's theorem

Theorem 1 *If a formula P is provable in classical propositional calculus, then $\neg\neg P$ is probable in intuitionistic propositional calculus.*

Note that this theorem cannot be extended onto intuitionistic predicate logic. This theorem is a trivialconsequencee of Seldin's normalization strategy [4].

3.1 Using `tauto` as a classical tautologies decision procedure and satisfiability

```
Coq < Goal forall x y z:Prop, ~((x \/ ~y \/ z) /\ ~z /\ y /\ ~x).
1 subgoal

  ============================
   forall x y z : Prop, ~ ((x \/ ~ y \/ z) /\ ~ z /\ y /\ ~ x)

Unnamed_thm < tauto.
Proof completed.
```

This means that `(x\/~y\/z)/\ ~z/\y/\ ~x` is no satisfiable.

```
Goal ((P:Prop)P\/~P->x)->~~x.
Proof.
Intros.
Unfold not.
Intros.
Apply H0.
Apply (H x).
Right.
Auto.
Qed.


Section foo.

Variable p:Prop.

Goal ~~(p\/~p).
tauto.
Save int.

Goal ((p\/~p)->p)->~~p.
try tauto.
Save cero.

Lemma uno :((P:Prop)P\/~P)->(p\/~p).
intro H.
exact (H p).
Qed.
```

```
Goal ((p\/~p)->p)->p.
1 subgoal

  p : Prop
  ============================
   (p\/~p->p)->p

Unnamed_thm < try tauto.
1 subgoal

  p : Prop
  ============================
   (p\/~p->p)->p

Lemma dos:(((P:Prop)P\/~P)->p)->((p\/~p)->p).

dos < Goal ((p\/~p)->~p)->p.
1 subgoal

  p : Prop
  ============================
   (p\/~p->~p)->p

Unnamed_thm < try tauto.
1 subgoal

  p : Prop
  ============================
   (p\/~p->~p)->p


Goal ~~~p->~p.
try tauto
Qed.

Variables (A:Set) (P:A->Prop).

Goal forall a:A, ~~((P a) \/ ~(P a)).
tauto.
Save triv.

(*
Goal ~~ (forall a:A, (P a \/ ~ P a)).
tauto.
User error: Tauto failed
*)
Variable te:forall a:A, (P a \/ ~ P a).
Goal ~~ (forall a:A, (P a \/ ~ P a)).
tauto.
Save metriv.
```

```
End k.
```

There is no need of a tactic dedicated to classical logic. As a consequence of the Glivenko's theorem : P is a classic thautology if and only if the following tactic of Coq builds a proof of it.

```
Require Classical.
 Goal P.
 apply NNPP; tauto.
```

where NNPP is defined in the Classical Module as `NNPP : (p:Prop)~~p -> p`. When you apply NNPP to P, you get the new goal P, which is instuitionistically valid. So you can use `tauto` to solve automatically your problem. Foer example:

```
Goal (a->b) -> ~a\/b.
1 subgoal

  a : Prop
  b : Prop
  c : Prop
  ============================
   (a -> b) -> ~ a \/ b

Unnamed_thm < tauto.
Toplevel input, characters 492-497
> tauto.
> ^^^^^
User error: Tauto failed

Unnamed_thm < apply NNPP.
1 subgoal

  a : Prop
  b : Prop
  c : Prop
  ============================
   ~ ~ ((a -> b) -> ~ a \/ b)

Unnamed_thm < tauto.
Proof completed.

Unnamed_thm < Save seis.
apply NNPP.
tauto.
seis is defined
-
 End prueba.
```

Ufortunately, Gilvenko's theorem, hence this estrategy, can not be extended to predicate calculus:

```
Coq < Goal (forall a : A, P a \/ ~ P a).
1 subgoal

  ============================
   forall a : A, P a \/ ~ P a

Unnamed_thm < apply NNPP.
1 subgoal

  ============================
   ~ ~ (forall a : A, P a \/ ~ P a)

Unnamed_thm < tauto.
Toplevel input, characters 556-561
> tauto.
> ^^^^^
User error: Tauto failed
```

But however `forall a : A, P a \/ ~ P a` is true in classical logic. In fact restarting the anterior proof:

```
Coq < Goal (forall a : A, P a \/ ~ P a).
Proof.
intro a.
apply classic.
Qed.
```

prove it.

4 Some remarks.

The possibility of a similar Curry–Howard formulae-as-types correspondence for classical logic looks to be a seminal development in this area, but whilst promising results have been achieved, there does not appear to be much agreement of what is at stake in claiming that such a correspondence exists [6]

Contents

References

[1] R. Dyckhoff. Contraction-free sequent calculi for intuitionistic logic. *J. Symb. Logic*, 57:795–807, 1992.

[2] P. Urzyczyn M.H. Sorensen. Logika intuicjonistyczna. http://www.mimuw.edu.pl/ urzy/Int.

[3] C. Muñoz. *Démonstration automatique dans la logique propositionnelle intuitionniste*. PhD thesis, Université Paris 7, 1994.

[4] J Seldin. Normalization and excluded middle. *Studia logica*, 1989.

[5] W.A. Howard. Ed.: J.P. Seldin and J.R. Hindley. *H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, chapter The formulae-as-types notion of constructions. Academic Press, 1980.

[6] C. A. Stewart. *On the formulae-as-types correspondence for classical logic.* PhD thesis, Oxford University, 2000.

[7] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V8.0*, April 2004.

[8] P. Casteran Y. Bertot. *Interactive Theorem Proving and Program Development. Coq'Art : the Calculus of Inductive Constructions.* Springer Verlag:Texts in Theoretical Computer Science, 2004.