Coq Formalization of a Tableau for the Classical-Intuitionistic Propositional Fragment of Ecumenical Logic

Renato R. Leme[1], **Giorgio Venturi**[1], **Bruno Lopes**[2]

[1]Instituto de Filosofia e Ciências Humanas
Universidade Estadual de Campinas (UNICAMP)
Campinas, SP – Brazil

[2]Instituto de Computação
Universidade Federal Fluminense (UFF)
Niterói, RJ – Brazil

{rntreisleme,gio.venturi}@gmail.com, bruno@ic.uff.br

Abstract. *In this paper, we describe a tableau system for reasoning about ecumenical propositional logic, and introduce the central definitions of its implementation in the Coq proof assistant.*

1. Introduction

Ecumenical logic studies the notion of logical consequence from the perspective of a neutral observer. From this point of view, one can argue about logical relations between propositions of two (or more) different logic systems. The idea of such a system goes back to the nineties, with the work of Krauss [Krauss 1992], but only recently its proof theory has been reviewed and received new advances. Notably, after Prawitz, in [Pereira and Rodriguez 2017], the semantics for ecumenical logic was formulated in terms of Kripke models, and the normalization of the correspondent natural deduction system was proved. Following Pereira's work, sequent calculi were proposed [Pimentel et al. 2021].

To illustrate how one could reason about ecumenical propositions, consider, for a moment, the following symbols: $p \to q$. If two logicians (one classical and the other intuitionistic) intend to read that as a proposition, they should first agree with the meaning of $\to$ (or, at least, agree to disagree!), as it means different things if they read the implication as a classical logician or as an intuitionistic. However, if they assume that the implication is intuitionistic, then both could be confident (either as an intuitionistic or as a classical logician) that the very same implication holds classically.

This fact could be symbolized in ecumenical logic as theorems (1) and (2):

$$(p \to_i q) \to_i (p \to_c q) \tag{1}$$
$$(p \to_i q) \to_c (p \to_c q) \tag{2}$$

To make it possible to reason about derivability interaction between such different systems, classical and intuitionistic logicians agree to share the semantics of the conjunction, negation, and bottom in the ecumenical environment. Those are called neutral

operators. With that setup, classical operators are then recovered in semantic level, creating an environment in which peace would finally reign. Based on [Pimentel et al. 2021], we formulate a tableau system for the propositional fragment of ecumenical logic and implemented it on Coq. The overall structure of the paper goes as follows: in Section 2, we review the notation and central definitions of ecumenical environment, in Section 3 we define the system TE_{pci} and introduce the rules of the tableau, and in Section 4 we present the central definitions of its implementation on Coq.

2. The language

2.1. Syntax

The language of ecumenical propositional logic comprehends a set Z of atomic formulas in addition with the logical symbols $\neg, \wedge, \vee_i, \vee_c, \rightarrow_i, \rightarrow_c, \bot$ and the non-logical symbols $(,)$. The grammar goes inductively as follows

1. Every atomic formula is an ecumenical formula;
2. If φ and ψ are ecumenical formulas, then $\neg\varphi$, $\varphi \wedge \psi$, $\varphi \vee_i \psi$, $\varphi \vee_c \psi$, $\varphi \rightarrow_i \psi$ and $\varphi \rightarrow_c \psi$ are ecumenical formulas.

2.2. Semantics

The semantic for ecumenical logic, proposed by Prawitz and followed by [Pereira and Rodriguez 2017], is intuitionistic. In terms of Kripke semantics, it is like in each possible world one could make use of classical axioms, even though the universe (understood as the set of worlds with its relations) would remain intuitionistic. On one hand, one could think about that as a systematic way to provide an intuitionistic framework where classical logicians could work without giving up their principles. On the other hand, on the intuitionistic side, one could think of ecumenism as a way to use classical inferences without sacrificing constructability.

A Kripke frame is a partially ordered set (poset) $\langle \mathcal{M}, \leq \rangle$ equipped with a valuation function $V : Z \rightarrow \mathcal{P}(\mathcal{M})$. A valuation is intuitionistic if it respects persistency: if $\omega \in V(p)$ and $\omega \leq \omega'$ then $\omega' \in V(p)$. A poset equipped with an intuitionistic valuation and whose $\leq$ is reflexive and transitive is an intuitionistic Kripke frame.

Definition 1. *Ecumenical Kripke Model. An Ecumenical Kripke Model EKM is a pair $\langle \mathcal{W}, V \rangle$ such that $\mathcal{W} = \langle \mathcal{M}, \leq \rangle$ is an intuitionistic Kripke frame where V is its corresponding intuitionistic valuation.*

Finally, ecumenical entailment $\Vdash$ is defined as follows.

$$
\begin{array}{ll}
\mathcal{M}, \omega \nVdash \bot & \\
\mathcal{M}, \omega \Vdash p & \Leftrightarrow \omega \in V(p) \\
\mathcal{M}, \omega \Vdash \neg\psi & \Leftrightarrow \forall v \in \mathcal{M} \text{ such that } \omega \leq v : v \nVdash \psi \\
\mathcal{M}, \omega \Vdash \varphi \wedge \psi & \Leftrightarrow \omega \Vdash \varphi \text{ and } \omega \Vdash \psi \\
\mathcal{M}, \omega \Vdash \varphi \vee_i \psi & \Leftrightarrow \omega \Vdash \varphi \text{ or } \omega \Vdash \psi \\
\mathcal{M}, \omega \Vdash \varphi \vee_c \psi & \Leftrightarrow \omega \Vdash \neg(\neg\varphi \wedge \neg\psi) \\
\mathcal{M}, \omega \Vdash \varphi \rightarrow_i \psi & \Leftrightarrow \forall v \in \mathcal{M} \text{ such that } \omega \leq v : v \Vdash \varphi \text{ implies } v \Vdash \psi \\
\mathcal{M}, \omega \Vdash \varphi \rightarrow_c \psi & \Leftrightarrow \omega \Vdash \neg(\varphi \wedge \neg\psi)
\end{array}
$$

If $\mathcal{M}, \omega \Vdash \varphi$ is the case, then we will say that ω satisfies φ in $\mathcal{M}$, or simply that φ is satisfiable, or $\omega \Vdash \varphi$ whenever the context is clear enough. As an important consequence of persistency, the entailment relation is *monotonic*, i.e, if $\omega \Vdash p$ and $\omega \leq \omega'$ then $\omega' \Vdash p$.

Definition 2. *A formula φ is an ecumenical consequence of a set of formulas Γ if, and only if, there is no ecumenical Kripke model $\mathcal{K}$ which entails every formula of Γ and does not entail φ. We will represent this relation as $\Gamma \Vdash \varphi$ and use $\Vdash \varphi$ when $\Gamma = \emptyset$.*

To illustrate model construction, let's show that this simple one-dot model

ω

$\omega \in V(p)$

satisfies the arbitrary atomic instance of *tertium non datur*: $p \vee_c \neg p$.

First, notice that, if $\omega \Vdash p \vee_c \neg p$, then $\omega \Vdash \neg(\neg p \wedge \neg(\neg p))$. But, if that is the case, then, for every ν such that $\omega \leq \nu$, $\nu \nVdash \neg p \wedge \neg(\neg p)$. Well, the only ν in our model is ω itself. Then, we have to show that $\omega \nVdash \neg p \wedge \neg(\neg p)$, which means that $\omega \nVdash \neg p$ or $\omega \nVdash \neg(\neg p)$. Conveniently, let's choose the left side. Thus, we have to show that

$$\omega \nVdash \neg p$$

i.e, exists a ν such that $\omega \leq \nu$ and $\nu \Vdash p$. It is sufficient to observe that $\omega \in V(p)$ and then, by definition, $\omega \Vdash p$. $\square$

3. The deductive system

The Ecumenical tableau system TE_{pci} is based on the ecumenical sequent system $LEci$ [Pimentel et al. 2021]. The rule for intuitionistic implication is the same as proposed by Fitting [Fitting 2013], according to which, if one finds a node $F\varphi \rightarrow_i \psi$, one can add both $T\varphi$ and $F\psi$, but *first*, they have to delete every F signed formula on that branch.

3.1. Basic definitions

1. A signed formula is an ecumenical proposition prefixed with T or F;
2. A *node* w is a signed formula;
3. The *root* of a chain C is a node w of C such that, for every w_1 of C, if $w_1 \leq w$ then $w_1 = w$;
4. A *branch* B is a chain of nodes with a root;
5. If S is a branch, then S_T is a subset of S with only T-signed formulas.

Definition 3. *A tree τ is a set of branches such that for every $B_n, B_m \in \tau$, if r_1 is the root of B_n and r_2 is the root of B_m, then $r_1 = r_2$.*

Definition 4. *A branch S is closed if, and only if, for an atomic p, both $Tp \in S$ and $Fp \in S$ or $\bot \in S$.*

In our definitions, $\alpha \parallel \beta$ denotes a choice between α and β. This choice is concurrent, but not exclusive: one can choose between α and β and then come back, later, to choose the other side, if necessary.

Now, it will be necessary to turn back and review our decision only if we fail to close the tree, which means that our choice has produced at least one open branch. In this case, we know that we made the wrong choice. In this situation, the tableau system allows the computer to come back to the point that it made that decision, review it, and attempt to close the tree again via a different road. In this process, we will say that a "bad" choice, which produces at least one open branch, is obliterated concerning closure. With that in mind, we define a version of closure that does not consider obliterated branches to decide if the tree is closed or not. According to the definition below, to know that one of the two branches *is not* obliterated is a sufficient and necessary condition to conclude that the sub-tree below $\alpha \parallel \beta$ is closed.

Definition 5. *A tree τ is closed if, and only if, every non-obliterated branch is closed.*

For this definition, we consider that all rules (except for the special β) produces only non-obliterated branches. In what follows, we introduce the rules for the system. We have chosen to keep, as much as possible, Fitting style for defining tableaux.

3.2. The system TE_{pci}

Regular α rules

$$\frac{S, T\,(\neg\psi)}{S, F\psi} \qquad \frac{S, T(\varphi \wedge \psi)}{S, T\varphi, T\psi}$$

$$\frac{S, F(\varphi \rightarrow_c \psi)}{S, T\varphi, T(\neg\psi)} \qquad \frac{S, F\,(\varphi \vee_c \psi)}{S, T(\neg\varphi), T(\neg\psi)}$$

Regular β rules

$$\frac{S, F\,(\varphi \wedge \psi)}{S, F\varphi \mid S, F\psi} \qquad \frac{S, T\,(\varphi \rightarrow_i \psi)}{S, F\varphi \mid S, T\psi} \qquad \frac{S, T\,(\varphi \vee_i \psi)}{S, T\varphi \mid S, T\psi}$$

$$\frac{S, T\,(\varphi \rightarrow_c \psi)}{S, F\varphi \mid S, F(\neg\psi)} \qquad \frac{S, T\,(\varphi \vee_c \psi)}{S, F(\neg\varphi) \mid S, F(\neg\psi)}$$

Special rules

Special α

$$\frac{S, F\,(\neg\psi)}{S_T, T\psi}$$

Special α

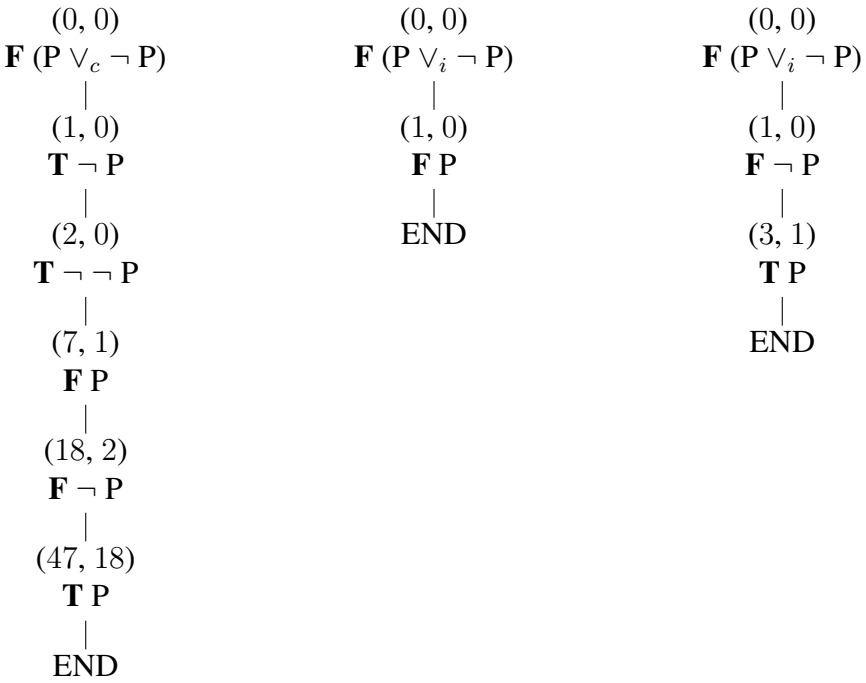
$$\frac{S, F(\varphi \rightarrow_i \psi)}{S_T, T\varphi, F\psi}$$

Special β

$$\frac{S, F\,(\varphi \vee_i \psi)}{S, F\varphi \parallel S, F\psi}$$

3.3. Examples

Let's take a look at an example. The first tree below is the complete tableau for $F(P \vee_c \neg P)$. The labels were added to make it easier to understand the behavior of the tree. They are generated automatically by our implementation and represents a relation between two nodes: (m, n) means that the node m is an expansion of node n. One can check that the first tree closes because of nodes 7 and 47, whereas the last two exemplify the application of the special β rule in an attempt to close $F(P \vee_i \neg P)$, which, as one can expect, fails for both possibilities.



4. Coq implementation

From now on, we present the central definitions of our implementation of TE_{pci} on Coq. The structure of the tree is a bit complex in virtue of the need to keep track of the history of node expansion. A *checkpoint* is created after each expansion of a special β rule, and a *metabranch* store information about the expansion of special α ones.

```
Inductive logicalFormula : Set :=
| Atom : string → logicalFormula
| Neg : logicalFormula → logicalFormula
| And : (logicalFormula × logicalFormula) → logicalFormula
| iOr : (logicalFormula × logicalFormula) → logicalFormula
| cOr : (logicalFormula × logicalFormula) → logicalFormula
| iImp : (logicalFormula × logicalFormula) → logicalFormula
| cImp : (logicalFormula × logicalFormula) → logicalFormula.

Inductive node : Type :=
| Empty : node
| Node : bool → logicalFormula → node.
```

```
Inductive tree : Type :=
| Root : tree
| Leaf : tree
| Unary : tree → tree → (bool × (Z×Z) × bool × node) → tree → tree
| Alpha : tree → tree → ((bool × (Z×Z) × bool × node) × (bool × (Z×Z) × bool × node)) → tree → tree
| Beta : (tree × tree × (bool × (Z×Z) × bool × node) × tree) → (tree × tree × (bool × (Z×Z) × bool × node) × tree) → tree.

Inductive checkpoint : Type := Checkpoint : Z → tree → checkpoint.

Inductive state : Type := State : tree → list checkpoint → bool → state.

Inductive pair : Type := ZZ : (Z × Z) → pair.

Inductive metabranch : Type := Metabranch : tree → Z → metabranch.
```

4.1. Notational conventions

```
Notation "[ A ]" := (Atom A) (at level 50).
Notation "~ A" := (Neg A).
Notation "A ∧ B" := (And (A, B)).
Notation "A ∨_i B" := (iOr (A, B)) (at level 100).
Notation "A ∨_c B" := (cOr (A, B)) (at level 100).
Notation "A →_i B" := (iImp (A, B)) (at level 90).
Notation "A →_c B" := (cImp (A, B)) (at level 90).
```

We omit function definitions of tree expanding rules[1].

4.2. Closure

In order to make closure function cleaner, the tableau tree is previously converted into a list of branches. The following functions presupposes that this procedure was already done.

```
Open Scope string_scope.

Definition neg := "~".

Definition contradiction (P Q : string) :=
  if eqb P (append neg Q) then true
  else if eqb Q (append neg P) then true
       else false.

Definition nodeToString (n : node) :=
  match n with
    Empty ⇒ EmptyString
  | Node t op ⇒
      match op with
        [P] ⇒ if t then P else (append neg P)
      | _ ⇒ EmptyString
```

[1]The complete implementation can be found on GitHub: https://github.com/renatoleme/TEpci_Coq.

```
    end
  end.
```

Special α rules demand that the closure algorithm ignores F signed formulas above them. We use the notion of meta branch to store the information that such a branch occurs in the tree. When a special α formula in a node x is expanded, we store the information according to which that branch has a special formula at node x in a *meta* branch. Here, this information is used to decide if a tree with such branches is closed or not, i.e, if it is closed modulo deleted F signed formulas. If a tree is meta closed, then it is closed in the sense of *Definition 5*.

```
Fixpoint meta_cmp (n : node) (branch : tree) (tag : Z) :=
  match branch with
  | ( Root | Leaf | Alpha _ _ _ _ | Beta _ _ ) ⇒ false
  | Unary hT pT (_, (ptag, ctag), t, r) nT ⇒
      if andb (Z.gtb tag ctag) (negb t) then
        orb (meta_cmp n pT tag) (meta_cmp n nT tag)
      else
        orb (orb (contradiction (nodeToString n) (nodeToString r)) (meta_cmp n pT
tag)) (meta_cmp n nT tag)
  end.

Fixpoint isBranchMetaClosed (branch : tree) (tag : Z) :=
  match branch with
  | ( Root | Leaf | Alpha _ _ _ _ | Beta _ _ ) ⇒ false
  | Unary hT pT (_, (ptag, ctag), t, r) nT ⇒
      orb (meta_cmp r nT tag) (orb (orb (meta_cmp r pT tag) (isBranchMetaClosed pT
tag)) (isBranchMetaClosed nT tag))
  end.

Fixpoint isMetaClosed (l : list metabranch) :=
  match l with
  | nil ⇒ true
  | h::tl ⇒
      let branch := getBranchFromMeta h in
      let tag := getInfoFromMeta h in
      andb (isBranchMetaClosed branch tag) (isMetaClosed tl)
  end.
```

4.3. Usage example

First of all, we define our atomic propositions as strings. Then, we use some auxiliary functions to setup the initial tree. Finally, we construct the tree for $F((P \rightarrow_c Q) \rightarrow_c P) \rightarrow_c P$ and $F((P \rightarrow_i Q) \rightarrow_i P) \rightarrow_i P$ and prove via *reflexivity* that *isMetaClosed* evaluates to *true* in the first case but, as one would expect, evaluates to *false* in the second.

```
Definition P := "P".
Definition Q := "Q".

Definition peirce_c :=
  makeInitialTree
```

```
    Root ((Node false ((( [P] →c [Q] ) →c [P]) →c [P]))::nil).
Definition peirce_i :=
  makeInitialTree
    Root ((Node false ((( [P] →i [Q] ) →i [P]) →i [P]))::nil).
Definition paths_peirce_c :=
  genPaths (upto 100) (pop (computeTableau peirce_c)).
Definition paths_peirce_i :=
  genPaths (upto 100) (pop (computeTableau peirce_i)).

Example peirce_c_theorem :
  isMetaClosed (postCheck paths_peirce_c) = true.
Proof.
  reflexivity.
Qed.

Example peirce_i_not_theorem :
  isMetaClosed (postCheck paths_peirce_i) = false.
Proof.
  reflexivity.
Qed.
```

5. Conclusion

In this paper, we have presented the tableau for ecumenical propositional logic, as well central aspects of our implementation of it. In future works, we intend to present the correspondent soundness and completeness proofs of the system, extend our rules to include predicate calculus, and expand the system to embrace further developments of ecumenical logic, such as ecumenical modalities.

6. Acknowledgments

The first author thanks the generous support of FAPESP, via grant 21/01025-3, São Paulo Research Foundation (FAPESP), through which this work is being possible. We would also like to thank the reviewers for their thoughtful comments and efforts towards improving our manuscript.

References

Fitting, M. (2013). *Proof methods for modal and intuitionistic logics*, volume 169. Springer Science & Business Media.

Krauss, P. (1992). A constructive refinement of classical logic.

Pereira, L. C. and Rodriguez, R. O. (2017). Normalization, soundness and completeness for the propositional fragment of prawitz'ecumenical system. *Revista Portuguesa de Filosofia*, 73(3):1153–1168. Publisher: JSTOR.

Pimentel, E., Pereira, L. C., and de Paiva, V. (2021). An ecumenical notion of entailment. *Synthese*, 198(22):5391–5413.