Natural deduction and semantic models of justification logic in the proof assistant Coq

JESÚS MAURICIO ANDRADE GUZMÁN*, *Posgrado en Filosofía de la Ciencia, Universidad Nacional Autónoma de México, 04510 Ciudad Universitaria, Mexico.*

FRANCISCO HERNÁNDEZ QUIROZ**, *Facultad de Ciencias, Universidad Nacional Autónoma de México, 04510 Ciudad Universitaria, Mexico.*

Abstract

The purpose of this paper is to present a formalization of the language, semantics and axiomatization of justification logic in Coq. We present proofs in a natural deduction style derived from the axiomatic approach of justification logic. Additionally, we present possible world semantics in Coq based on Fitting models to formalize the semantic satisfaction of formulas. As an important result, with this implementation, it is possible to give a proof of soundness for LP with respect to Fitting models.

Keywords: Justification logic, natural deduction, Coq, modal logic, soundness

1 Introduction

Justification logic (2) is a family of logics, resulting from a general interpretation of the logic of proofs (LP), first developed by Artemov in (1). In general terms, justification logic makes an explicit analysis of assertions, compared with the implicit approach of traditional modal logic. There are many examples of justification logics in the literature (5; 9; 12), designed with different purposes, most of the time as an alternative of a modal logic previously defined. More work is currently in progress following this line of thought, and we see this as an opportunity to present a computational approach in the context of the proof assistant Coq[1] since it has an important role in the formalization of theories.

The purpose of this paper is to give a formalization of an inference system in the style of natural deduction using justification logic. The formalization follows a deep embedding for the language and rules of inference of natural deduction for J_0, J and LP. With respect to the semantics, the approach is hybrid between a deep and a shallow embedding, where semantic satisfaction is defined in terms of the internal logic of Coq. Having the language and rules in place, we present the construction of justification polynomials of valid formulas in modal logic, which correspond to their counterparts in LP. Additionally, using our formalization for semantic models, we present a proof of soundness for LP. The source code of our implementation can be downloaded from https://sites.google.com/view/lp-coq/.

The implementation was developed with version 8.7 of Coq and tested with version 8.9. It is inspired in part by the work of de Wind (6), which presents natural deduction of modal logic in Coq version 6.3. However, our semantic definition and the rules of inference are considerably

*E-mail: guzmandrade@gmail.com
**E-mail: fhq@ciencias.unam.mx
[1] More information about Coq can be consulted in https://coq.inria.fr.

Vol. 00, No. 0,
doi:10.1093/jigpal/jzaa007

different from de Wind's. Also previously, Melvin Fitting presented an implementation of the realization theorems in Prolog (8), which calculates the justification polynomial from a valid formula in S4 to a justified formula in LP. The objective of that approach was to obtain the justification polynomial algorithmically, whereas our implementation in COQ can be used to obtain the justification polynomial step by step in the proof. The interactive nature of the proof assistant COQ allows us to reveal information about the structure of a proof that is not present in an algorithmic approach, such as Fitting's implementation.

Structure of the article

In Section 2, we present the syntax of LP and its COQ implementation. In Section 3, we lay out the semantic definition of satisfaction and validity in COQ based on Fitting models. In Section 4, we offer the core axioms of the theory, including the constant specification $\mathcal{CS}$, and in particular, how the assignment of justification constants to each axiom was addressed. Section 5 describes the rules of inference involved and how the notion of natural deduction and the set of axioms accepted in the theory are integrated. In Section 6, we discuss some examples of proofs using this implementation, both in terms of natural deduction and semantic models. Finally, in Section 7, we present some conclusions.

2 Language

For countably many justification variables e_i and justification constants c_i, a *justification term* t is defined by the grammar:

$$t ::= e_i \mid c_i \mid t \cdot t \mid t + t \mid !\, t, \text{ where } i \in \mathbb{N}.$$

A *justification polynomial* or *proof polynomial* is a justification term that follows this grammar, and each justification term t is member of the set Tm.

A formula φ in the language of LP follows the following BNF grammar, where p_i is a propositional atom, $\bot$ is the constant *false* and t is a justification term:

$$\varphi ::= p_i \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \varphi \rightarrow \varphi \mid t : \varphi, \text{ where } i \in \mathbb{N}.$$

Each formula φ produced by this grammar is member of the set Fm.

In order to define the language of justification logic in COQ, we use inductive products. For the propositional variables, elements of the set `var`, we have

```
Inductive var: Set := p : nat -> var.
```

where `nat` is the set of natural numbers (a native type of COQ); e.g. `p 0` represents the first propositional variable, `p 1` the second and so forth.

TABLE 1 Notation used.

Operation	Notation	Operation	Notation
And	p /\' q	Just	s $ F
Or	p \/' q	App	s @ t
Not	¬ p	Sum	s # t
If	p -> q	Chk	! t

For justification terms, we define the set Tm:

```
Inductive Tm: Set :=
| e     : nat -> Tm
| c     : nat -> Tm
| jsum  : Tm -> Tm -> Tm
| japp  : Tm -> Tm -> Tm
| jchk  : Tm -> Tm.
```

where the constructor e applied to zero, e 0, represents the justification variable e_0, c 0 represents the justification constant c_0 and the binary constructors japp and jsum and the unary constructor jchk represent the functional operators $(\cdot)$, $(+)$ and (!), respectively.

Accordingly, a formula in COQ is defined as the set Fm, as follows:

```
Inductive Fm: Set :=
| Fal   : Fm
| Atom  : var -> Fm
| And   : Fm -> Fm -> Fm
| Or    : Fm -> Fm -> Fm
| If    : Fm -> Fm -> Fm
| Just  : Tm -> Fm -> Fm.
```

where Atom is the constructor for propositional atoms; Fal is the constant *false* ($\bot$); And, Or and If are the propositional operators with the usual meaning; and Just is the binding of a justification with a formula, e.g. Just (e 0) (Atom (p 1)) represents $e_0 : p_1$. Using these operators, negation is defined as $\neg\varphi \equiv \varphi \to \bot$, the constant *truth* as $\top \equiv \neg\bot$, and we use $(\varphi \leftrightarrow \psi)$ as a translation of $(\varphi \to \psi) \wedge (\psi \to \varphi)$.

COQ allows us to define a special notation for the operators defined above to improve the readability of formulas. For example, for And, Just and japp we have p /\' q, s $ F and s @ t, respectively. Table 1 has the operations and their corresponding notation used to represent formulas in the source code of the project.

3 Semantics

The standard semantics of justification logic is due to Fitting (7). A Fitting model is the tuple $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{L}, \mathcal{E} \rangle$, in which the first three elements are a traditional Kripke model, i.e. $\mathcal{W}$ is a non-empty set of possible worlds, $\mathcal{R}$ is the accessibility relation between worlds, $\mathcal{L}$ is the labelling function for propositional variables and the last element is the evidence function $\mathcal{E} : Tm \times Fm \mapsto \wp(\mathcal{W})$ that maps a justification term and a formula to a subset of $\mathcal{W}$.

3.1 Fitting models

Let $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{L}, \mathcal{E} \rangle$ be a Fitting model, for a world $w \in \mathcal{W}$ and a formula φ; the truth of φ in w, or $\mathcal{M}, w \models \varphi$, is given by

For each $w \in \mathcal{W}$,

1. $\mathcal{M}, w \models p$, iff[2] $p \in \mathcal{L}(w)$, for the propositional variable p;
2. $\mathcal{M}, w \not\models \bot$, *false* is never the case;
3. $\mathcal{M}, w \models \varphi \wedge \psi$, iff $\mathcal{M}, w \models \varphi$ and $\mathcal{M}, w \models \psi$;
4. $\mathcal{M}, w \models \varphi \vee \psi$, iff $\mathcal{M}, w \models \varphi$ or $\mathcal{M}, w \models \psi$;
5. $\mathcal{M}, w \models \varphi \rightarrow \psi$, iff $\mathcal{M}, w \models \varphi$ implies $\mathcal{M}, w \models \psi$;
6. $\mathcal{M}, w \models t : \varphi$, iff $w \in \mathcal{E}(t, \varphi)$, and for every $v \in \mathcal{W}$, such that if $w\mathcal{R}v$[3] , then $\mathcal{M}, v \models \varphi$.

From (1) to (5) we have propositional logic, and (6) adds justification logic.

To express this in COQ, we need to first define the structures involved.

```
Record frame:Type := {
  W : Set;
  R : W -> W -> Prop }.
```

```
Record kripke: Type := {
  F : frame;
  L : (W F) -> var -> Prop }.
```

```
Record fitting: Type := {
  K: kripke;
  E: Fm -> Tm -> (W (F K)) -> Prop }.
```

that correspond to the usual semantic structures based on possible worlds. And finally, we define the function `sat` recursively as follows:

```
Fixpoint sat (M:fitting) (x:(W (F (K M)))) (phi:Fm): Prop :=
match phi with
| Fal => False
| (Atom v) => (L (K M) x v)
| phi1 /\' phi2 => (sat M x phi1) /\ (sat M x phi2)
| phi1 \/' phi2 => (sat M x phi1) \/ (sat M x phi2)
| phi1 --> phi2 => (sat M x phi1) -> (sat M x phi2)
| s $ phi =>
    (E M phi s x) /\ forall y:(W (F (K M))),
        (R (F (K M)) x y) -> (sat M y phi)
end.
```

with the custom notation `||= phi`, representing the semantic validity of the formula `phi` for every Fitting model.

```
Notation " ||= phi" := (valid phi) (at level 30).
```

[2] if and only if

[3] $(w, v) \in \mathcal{R}$

3.2 Accessibility relation

It is straightforward to impose restrictions to the accessibility relation $\mathcal{R}$ to express the acceptance of axioms semantically, in the same way as in modal logic. The following are the properties reflexivity and transitivity of $\mathcal{R}$ in Coq. The rest of the properties used can be consulted in the source code provided.

```
Hypothesis Rref: forall (M:fitting) (x:(W (F (K M)))),
(R (F (K M)) x x).

Hypothesis Rtra: forall (M:fitting) (x y z:(W (F (K M)))),
(R (F (K M)) x y) /\ (R (F (K M)) y z) ->
(R (F (K M)) x z).
```

4 Axiomatization

4.1 Propositional axioms

Usually, when a modal or justification logic is defined, the propositional part is assumed as given. However, for this implementation, we are going to explicitly list the propositional axioms accepted, in order to assign a justification constant to each axiom. The following is a list of axioms (13), describing a logic in which we do not accept the tautology $\varphi \vee \neg\varphi$, a form of **LEM**[4].

```
Definition AxThen1 (P Q:Fm) :=
(P --> Q --> P).

Definition AxThen2 (P Q R:Fm) :=
((P --> Q --> R) -->
((P --> Q) --> P --> R)).

Definition AxAnd1 (P Q:Fm) :=
((P /\' Q) --> P).

Definition AxAnd2 (P Q:Fm) :=
((P /\' Q) --> Q).

Definition AxAnd3 (P Q:Fm) :=
(P --> (Q --> (P /\' Q))).
```

```
Definition AxOr1 (P Q:Fm) :=
(P --> (P \/' Q)).

Definition AxOr2 (P Q:Fm) :=
(Q --> (P \/' Q)).

Definition AxOr3 (P Q R:Fm) :=
((P --> R) -->
((Q --> R) -->
((P \/' Q) --> R))).

Definition AxFalse (P:Fm) :=
(Fal --> P).
```

[4]It should be clear that if the system accepts the tautology $\varphi \vee \neg\varphi$, then we simply add that or some other equivalent e.g. $\neg\neg\varphi \rightarrow \varphi$, to the list. We chose this list because it does not require to define the negation as primitive, and we are following the internal logic of Coq that would require an additional library (`Coq.Logic.Classical_Prop`) in order to interact with proofs that involve **LEM** (law of the excluded middle).

4.2 Justification axioms

4.2.1 Application It is an explicit form of the axiom **K**, $\Box(\varphi \to \psi) \to \Box\varphi \to \Box\psi$, from modal logic. Application establishes the behaviour of the operator $(\cdot)$. Let s and t be two justification terms and φ and ψ two formulas. Application states that

$$s : (\varphi \to \psi) \to t : \varphi \to (s \cdot t) : \psi.$$

In COQ, application is presented as

```
Definition AxApp (s t:Tm) (P Q: Fm) :=
(s$(P --> Q)) --> (t$P)  --> ((s @ t)$Q).
```

4.2.2 Sum These two axioms represent the idea that adding new evidence to an accepted assertion does not affect the acceptance of that assertion. The sum operation $(+)$ between two justification terms s and t, associated to a formula φ, states that

$$s : \varphi \to (s + t) : \varphi$$

$$t : \varphi \to (s + t) : \varphi.$$

In COQ, the axioms of sum are

```
Definition AxSum1 (s t:Tm) (P: Fm) :=
(s$P) --> ((s # t)$P).

Definition AxSum2 (s t:Tm) (P: Fm) :=
(t$P) --> ((s # t)$P).
```

4.2.3 Proof check It is the explicit form of the *positive introspection* axiom (sometimes called **4**) of modal logic, $\Box\varphi \to \Box\Box\varphi$, which states that

$$t : \varphi \to !\, t : (t : \varphi).$$

It can be read as if we accept the evidence t for a formula φ, then we are in position to accept another piece of evidence $(!\, t)$ that justifies $t : \varphi$.

In COQ, proof check is presented as

```
Definition AxChk (t:Tm) (P:Fm) :=
(t$P) --> ((!t)$t$P).
```

4.2.4 Factivity It is the explicit form of the modal logic axiom **T**, $\Box\varphi \to \varphi$. If we accept the factivity principle, for a justification term t and a formula φ, then we have:

$$t : \varphi \to \varphi.$$

If we have evidence t for a formula φ, then φ must true. The factivity principle in COQ is presented as

```
Definition AxFact (t:Tm) (P:Fm) :=
t$P --> P.
```

4.3 Properties of the evidence function $\mathcal{E}$

In order to guarantee the validity of the main axioms of justification logic, $\mathcal{E}$ needs to satisfy the following properties.

- Application closure, $\mathcal{E}(s, \varphi \to \psi) \cap \mathcal{E}(t, \varphi) \subseteq \mathcal{E}(s \cdot t, \psi)$
- Monotony, $\mathcal{E}(s, \varphi) \cup \mathcal{E}(t, \varphi) \subseteq \mathcal{E}(s + t, \varphi)$
- Verification, $\mathcal{E}(t, \varphi) \subseteq \mathcal{E}(!\, t, t : \varphi)$

In COQ, for the properties of function $\mathcal{E}$, we have

```
Hypothesis Eapp: forall (M:fitting) (x:(W (F (K M))))
  (phi psi:Fm) (s t: Tm),
  (E M (phi --> psi) s x) /\  (E M phi t x) ->
  (E M psi (s @ t) x).


Hypothesis Esum: forall (M:fitting) (x:(W (F (K M))))
  (phi:Fm) (s t: Tm),
  (E M phi s x) \/ (E M phi t x)  ->
  (E M phi (s # t) x).


Hypothesis Echk: forall (M:fitting) (x:(W (F (K M))))
  (phi:Fm) (t: Tm),
  (E M phi t x) ->
  (E M (t$phi) (!t) x).
```

4.4 Constant specification

The set of axioms that a particular justification logic accepts is represented with the constant specification, $\mathcal{CS}$. Our particular $\mathcal{CS}$ includes the propositional and justification axioms listed in 4.1 and 4.2. The structure of $\mathcal{CS}$ is given by

$$\mathcal{CS} = \{c_o : A_0, c_1 : A_1, \dots\},$$

where each $c_i : A_i$ is a justified axiom for the constant c_i and the axiom A_i. In terms of semantics, we say that for every world $w \in \mathcal{W}$, we assume $\mathcal{M}, w \models c_i : A_i$. As we can see, being member of $\mathcal{CS}$ is what characterizes the axioms in a justification logic. A particular $\mathcal{CS}$ can be

- *Empty*, i.e. $\mathcal{CS} = \varnothing$, the justification logic with an empty $\mathcal{CS}$ is called J_0. Corresponds to the case in which an agent does not accept axiomatically any proposition.

- *Finite*. If $\mathcal{CS}$ is a finite set of formulas, then we can track the steps in the inference up until a finite number of derivations.
- *Axiomatically appropriate*. This characterization corresponds with the case in which, for an axiom A and a constant justification c_1, such that $c_1 : A \in \mathcal{CS}$, there exists an additional justification c_2, such that $c_2 : c_1 : A \in \mathcal{CS}$. In general, if the sequence of justification terms $c_n, c_{n-1}, \ldots, c_1$, is such that

$$c_n : c_{n-1} : \cdots : c_1 : A \in \mathcal{CS},$$

then

$$c_{n+1} : c_n : c_{n-1} : \cdots : c_1 : A \in \mathcal{CS}.$$

This means that for every axiom A in $\mathcal{CS}$, exists an additional justification c_{n+1}, for every member of $\mathcal{CS}$.
- *Total*. A constant specification is total ($\mathcal{TCS}$) if for every axiom A, there must exists a sequence of justifications $c_n, c_{n-1}, \ldots, c_1$, such that

$$c_n : c_{n-1} : \cdots : c_1 : A \in \mathcal{CS}.$$

A justification logic is characterized by its constant specification $\mathcal{CS}$, i.e. $\mathsf{J}_{\mathcal{CS}} = \mathsf{J}_0 + \mathcal{CS}$. Justification logic J_0 admits application and sum, but it does not include factivity nor proof check. Justification logic J corresponds with a total $\mathcal{CS}$, i.e. $\mathsf{J} = \mathsf{J}_0 + \mathcal{TCS}$. The original justification logic LP, the logic of proofs, corresponds with J plus factivity and proof check.

4.4.1 $\mathcal{CS}$ in Coq The Coq implementation of the set $\mathcal{CS}$ needs to explicitly assign a particular justification term to each axiom listed in 4.1 and 4.2. To illustrate the inductive structure, the assignment of justification constant has the following definition:

```
Inductive CS: Fm -> Prop :=
| cs1: forall (P Q:Fm),
  CS ((c 1)$(AxThen1 P Q))
| cs2: forall (P Q R:Fm),
  CS ((c 2)$(AxThen2 P Q R))
...
| cs10: forall (s t:Tm) (P Q: Fm),
  CS ((c 10)$(AxApp s t P Q))
| cs11: forall (s t:Tm) (P: Fm),
  CS ((c 11)$(AxSum1 s t P))
...
```

This inductive structure is sufficient to show the main properties of justification logic. For this, we have justified axioms $c_i : A_i$, such that one constant is assigned to each axiom. Given that proof check is a valid axiom for LP, it is possible to emulate the properties either in an axiomatically appropriate or in a total $\mathcal{CS}$. For example, the prefixed axiom $e_3 : e_2 : e_1 : A \in \mathcal{CS}$ can be derived from $c : A \in \mathcal{CS}$ and proof check as $!\,!\,c :!\,c : c : A$.

5 Natural deduction

Conventionally, rules of inference for a natural deduction system are given instead of listing the axioms accepted. In justification logic, we want to track the axioms used in a derivation, and for that goal the rules of inference are going to be derived from the axioms.

To distinguish between formulas asserted and formulas proved, we use the parameter `Assert`:

```
Parameter Assert: Fm -> Prop.
```

The following code is the definition of the inductive product `Provable` that formalizes in COQ the notion of provability to establish that an axiom is provable in the system:

```
Inductive Provable: Fm -> Prop :=
| then1:  forall (P Q:Fm),
  Provable(AxThen1 P Q)
...
| jApp: forall (s t:Tm) (P Q: Fm),
    Provable (AxApp s t P Q)
| jS1: forall (s t:Tm) (P: Fm),
    Provable (AxSum1 s t P)
...
```

We use the customary notation `|- phi` to represent that the formula `phi` is provable:

```
Notation " |- phi" := (Provable phi) (at level 30).
```

5.1 Core rules

We need to formalize the notion of provability of asserted formulas, commonly known as assumption, but in order to distinguish it from the COQ tactic `assumption` we will call it **Hyp** (from hypotheses).

```
Hypotheses Hyp: forall (P:Fm),
  Assert P -> Provable P.
```

We also define modus ponens (**MP**) in the usual way:

```
Hypotheses MP: forall (P Q: Fm),
  Provable (P --> Q) -> Provable P -> Provable Q.
```

And now the deduction theorem (**DT**):

```
Hypotheses DT: forall (P Q: Fm),
  (Assert P -> Provable Q) -> Provable (P --> Q).
```

An additional rule is needed for including in the deduction system the list of axioms associated with justifications constants in $\mathcal{CS}$. The rule **CSprov** states that if $\varphi \in \mathcal{CS}$, then φ is provable:

```
Hypotheses CSprov: forall (P:Fm),
  CS (P) -> Provable (P).
```

5.2 Internalization

Internalization is an explicit form of the necessitation rule (**NEC**) of modal logic, $\vdash \varphi \Rightarrow \vdash \Box\varphi$, which states that if $\vdash \varphi$, then there exists a proof polynomial t, such that $\vdash t : \varphi$.

Given a constant specification $\mathcal{CS}$, we can prove internalization using the base case `CSprov`, by induction on the structure of the formula:

```
Theorem internalization (P:Fm):
  Provable (P) -> exists t, (Provable (t$P)).
Proof.
intros.
induction H.
eexists; apply CSprov; apply cs1.
eexists; apply CSprov; apply cs2.
...
Qed.
```

Otherwise, the principle of internalization can be integrated as an additional rule of inference:

```
Hypotheses RInt: forall (P:Fm),
  Provable (P) -> exists t:Tm, Provable(t$P).
```

5.3 Deriving rules of inference from the axioms

As we have said before, this is an axiomatic proof system. In order to use the characteristics of COQ as proof assistant, we have derived the rules of inference in the style of natural deduction from the axioms of $\mathcal{CS}$.

To illustrate this idea, we just expand the proofs for the rule of introduction of $\wedge$ and elimination of $\vee$. The rest of the proofs can be consulted in the source code.

- Introduction of ($\wedge$)

```
1  Theorem andI: forall (P Q: Fm),
2    Provable P -> Provable Q ->
3    Provable(P /\' Q).
4  Proof.
5  intros.
6  apply (MP Q); apply (MP P).
7  apply and3.
8  auto.
9  auto.
10 Qed.
```

Line 7 (`apply and3`) is the key step that is the use of the axiom `AxAnd3` to complete the proof.

- Elimination of ($\vee$)

```
1  Theorem orE: forall (P Q R: Fm),
2    Provable (P \/' Q) ->
3    (Assert P -> Provable R) ->
4    (Assert Q -> Provable R) ->
5    Provable R.
6  Proof.
7  intros.
8  eapply MP; eapply MP; eapply MP.
9  apply or3.
10 apply DT; exact H0.
11 apply DT; exact H1.
12 trivial.
13 Qed.
```

In this proof, the key step is on line 9 (`apply or3`), which corresponds with the use of the axiom `AxOr3`, after interacting with the proof goal using the rules of inference to give it the form of the axiom.

6 Examples of derivations using the formalization

With our Coq formalization, it is possible to perform derivations in J_0, which amounts to emulate justification constants with only justification variables, and derivations in J and LP using its constant specification. In the remainder of this section, we present natural deduction proofs and we give a proof of soundness for the system with respect to the semantics defined in Section 3.

6.1 Realizations

With our implementation, it is possible to obtain realizations from a valid formula in modal logic S4 to a justified formula in LP. For example, the well-known property in modal logic $\Box\varphi \wedge \Box\psi \rightarrow \Box(\varphi \wedge \psi)$ has a justification polynomial that we can construct step by step. In (4, Example 3.1), Artemov offers this derivation.

1. $A \rightarrow (B \rightarrow (A \wedge B))$, propositional axiom;
2. $a : (A \rightarrow (B \rightarrow (A \wedge B)))$, from 1, by internalization of a propositional axiom;
3. $x : A \rightarrow (a \cdot x) : (B \rightarrow (A \wedge B))$, from 2, by application and modus ponens;
4. $x : A \rightarrow (y : B \rightarrow ((a \cdot x) \cdot y) : (A \wedge B))$, from 3, by application and propositional reasoning;
5. $x : A \wedge y : B \rightarrow ((a \cdot x) \cdot y) : (A \wedge B)$, from 4, by propositional reasoning.

Therefore, the realization for $\Box\varphi \wedge \Box\psi \rightarrow \Box(\varphi \wedge \psi)$, given the justification variables x and y, for $\Box\varphi$ and $\Box\psi$, respectively, and $a : (\varphi \rightarrow (\psi \rightarrow (\varphi \wedge \psi))) \in \mathcal{CS}$ is $x : A \wedge y : B \rightarrow ((a \cdot x) \cdot y) : (A \wedge B)$. In Coq, the proof is as follows:

```
1  Example andJust: forall (A B: Fm),
2    exists (a b c: Tm), Provable(a$A /\' b$B --> c$(A /\' B)).
3  Proof.
4  intros.
5  exists x; exists y.
6  eexists ((fun _ _:Tm => _) x y).
7
8  apply DT; intro.
9
10 eapply (App _ y B (A /\' B)).
11 eapply (App _ x A (B --> A /\' B)).
12
13 eapply CSprov.
14 apply cs5.
15
16 apply Hyp in H; apply andE1 in H.
17 trivial.
18 apply Hyp in H; apply andE2 in H.
19 trivial.
20 Qed.
```

Some notes about this derivation:

1. Line 5 assigns the variables `x` and `y` to the justification terms `a` and `b`, respectively.
2. In line 6, the third justification term `c` is defined in terms of the previous two, i.e. $c = f(x, y)$.
3. In lines 13 and 14, after applying instances of the application axiom, we assign the justification constant c_5 (`(c 5)` that corresponds with the propositional axiom $A \rightarrow B \rightarrow (A \wedge B)$).
4. From line 16 to line 20, we use the natural deduction rules to reason propositionally.

Once completed the proof, the value of justification term `c` is `((c 5 @ x) @ y)`, which corresponds with the derivation on paper, and the justification constant `(c 5)` corresponds to the propositional axiom `(P -> (Q -> (P /\' Q)))` in $\mathcal{CS}$.

It is possible to give this proof without using a constant specification, emulating justification logic J_0 that reasons in terms of only justification variables. To do that, the derivation can be expressed in terms of justification variables:

$$z : (A \to B \to (A \wedge B)), x : A, y : B \vdash t : (A \wedge B)$$

for some justification term $t = f(x, y, z)$, in which z is a justification variable assigned to a propositional axiom. In Coq, the proof is as follows:

```
1  Example J0andJust: forall (A B:Fm),
2      Assert (z$(A --> B --> (A /\' B))) ->
3      Assert (x$A) -> Assert (y$B) ->
4      exists (t:Tm), Provable(t$(A /\' B)).
5  Proof.
6  intros.
7  eexists ((fun _ _:Tm => _) x y).
8  eapply (App _ y B (A /\' B)).
9  eapply (App _ x A (B --> A /\' B)).
10 apply Hyp.
11 exact H.
12 apply Hyp.
13 trivial.
14 apply Hyp.
15 trivial.
16 Qed.
```

This proof is very similar to the previous one, with the difference that this time instead of using $\mathcal{CS}$ to complete the proof, in line 11 the premise $z : (A \to B \to (A \wedge B))$ (labelled as `H`) fulfils the same role.

6.2 A semantic example: soundness of LP

To exemplify the semantic capabilities of this formalization, we present the proof of soundness for justification logic LP. We want to prove that every formula that can be derived in LP is logically valid with respect to the semantic models implemented. The statement for the proof is

```
Theorem Soundness: forall (P:formula),
|- P -> ||= P.
```

The proof in Coq begins by manually unfold the predicates: validity in general `valid` or $||=$, validity given a frame `valid_in_frame`, validity given a Kripke model `valid_in_kripke`

and validity given a Fitting model `valid_in_fitting`.

```
unfold valid; intros.
unfold valid_in_frame; intros.
unfold valid_in_kripke; intros.
unfold valid_in_fitting; intros.
```

This unfolding allow us to use an arbitrary model to prove the semantic validity of the axioms. We proceed by induction on the structure of the predicate `|-`, or `Provable`, i.e. the provability of the formula `P`. Since the predicate `Provable` is defined by the axioms of the system, the induction consists in giving a proof of the semantic validity of each axiom.

The semantic validity of the propositional part is straightforward since the rules are defined in terms of the internal logic of COQ. In the case of the application axiom (`AxApp`), the corresponding semantic property is used for the evidence function, $\mathcal{E}(s, \varphi \to \psi) \cap \mathcal{E}(t, \varphi) \subseteq \mathcal{E}(s \cdot t, \psi)$ (`Eapp`), as follows:

```
unfold AxApp.
...
replace E0 with (E (Build_fitting (Build_kripke F0 L0) E0)).
eapply Eapp.
...
```

With the tactic `replace`, we are instructing the proof assistant to substitute `E0` with the function of evidence that is part of the fitting model formed by the frame `F0` and the valuation `L0` in order to apply the semantic property Eapp.

Similarly, the proof of the semantic validity for the sum axioms (`AxSum1` and `AxSum2`) uses the semantic property $\mathcal{E}(s, \varphi) \cup \mathcal{E}(t, \varphi) \subseteq \mathcal{E}(s + t, \varphi)$ (`Esum`)

```
unfold AxSum1.
...
replace E0 with (E (Build_fitting (Build_kripke F0 L0) E0)).
eapply Esum.

unfold AxSum2.
...
replace E0 with (E (Build_fitting (Build_kripke F0 L0) E0)).
eapply Esum.
...
```

and so forth.

7 Conclusion

Justification logic is gaining attention these days, among other things, because it is a revision of well-established ideas presented in traditional modal logic. Although there are many computer

formalizations of modal logic, our formalization was able to capture the style of natural deduction proofs and to combine it with the axiomatic structure of justification logic. There are already multi-agent justification logics in the literature, e.g. (12), but the language described in this formalization is limited to a single agent case. Our intention was to give a formal language simple enough to make it easier to identify errors or to serve as the basis for a further expansion to a more complex version.

Being an axiomatic proof system, one of the main challenges about expressing justification logic in COQ was to think of a structure that could manage the axioms used in the proofs and preserve the natural deduction style of the derivations. We believe that this solution, i.e. preserving the mechanic approach of natural deduction and explicitly listing the axioms and their justification constants in $\mathcal{CS}$, could be used in future applications.

Justification terms encode the derivation of formulas and they do so in the object language of the formal system (its internalized proof), a feature absent in other representations of knowledge, such as those based on modal logic. This would allow adapt our approach in other fields, such as the representation of knowledge in artificial intelligence. For example, the proof assistant COQ was used for this purpose in (10), work that is based on the representation of a formal ontology.

Funding

This work was supported by the Consejo Nacional de Ciencia y Tecnología [grant number CB-2013-01/221341]; and the Departamento de Matemáticas, Facultad de Ciencias, Universidad Nacional Autónoma de México.

Acknowledgements

We thank the anonymous referees for the careful reading and their valuable comments which help us to enhance the content of this paper.

References

[1] S. Artemov. Operational modal logic. *Technical Report 95-29*. Mathematical Sciences Institute, Cornell University, 1995. https://sartemov.ws.gc.cuny.edu/files/2014/01/download-3.pdf.

[2] S. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, **7**, 1–36, 2001.

[3] S. Artemov and E. Nogina. Introducing justification into epistemic logic. *Journal of Logic and Computation*, **15**, 1059–1073, 2005. https://doi.org/10.1093/logcom/exi053.

[4] S. Artemov. The logic of justification. *Review of Symbolic Logic*, **1**, 477–513, 2008.

[5] S. Bucheli, M. Ghari and T. Studer. Temporal justification logic. In *Proceedings of the Ninth Workshop on Methods for Modalities (M4M9 2017)*, S. Ghosh and R. Ramanujam eds, pp. 59–74. Indian Institute of Technology, Kanpur, India, 8th to 10th January 2017. Electronic Proceedings in Theoretical Computer Science 243, 2017.

[6] P. de Wind. *Modal logic in coq*. Master's thesis, Vrije Universiteit in Amsterdam, 2001.

[7] M. Fitting. The logic of proofs, semantically. *Annals of Pure and Applied Logic*, **132**, 1–25, 2005. http://dx.doi.org/10.1016/j.apal.2004.04.009.

[8] M. Fitting. *TR–2013005: Realization Implemented*, 2013. http://academicworks.cuny.edu/gc_cs_tr/380.

[9] M. Ghari. Pavelka-Style duzzy justification logics. *Logic Journal of the IGPL*, **24**, 743–773, 2016. https://doi.org/10.1093/jigpal/jzw019.

[10] V. Lenko, V. Pasichnyk, N. Kunanets and Y. Shcherbyna. Knowledge representation and formal reasoning in ontologies with coq. In *Advances in Computer Science for Engineering and Education*, Z. Hu, S. Petoukhov, I. Dychka and M. He eds, pp. 759–770. Springer International Publishing, Cham, 2019.
[11] L. Menasché Schechter. A logic of plausible justifications. In *Logic, Language, Information and Computation*, L. Ong and R. de Queiroz eds, pp. 306–320. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
[12] B. Renne. Multi-agent justification logic: communication and evidence elimination. *Synthese*, **185**, 43–82, 2012. https://doi.org/10.1007/s11229-011-9968-7.
[13] A. S. Troelstra and H. Schwichtenberg. *Basic Proof Theory*, 2nd edn. Cambridge University Press, New York, NY, USA, 2000.

Received 15 January 2018