CSE203 - Project

Timestamp: Wednesday 12^{th} December, 2018 - 13:58

2018-2019

We are here interested in proving facts about propositional logic. The purpose of this project is the proof of the 2 following facts:

1. natural deduction is correct w.r.t. the interpretation of assertions;
2. it is decidable to check that an assertion is universally valid. We are going to check that by implementing a sound normalization algorithm for assertions, and then to write, in Coq, a simple decision for the universal validity of the normalized assertions.

We provide a Coq skeleton file `prop.v` and we ask you to fill the missing definitions & proofs.

Assertions - we assume given an infinite countable set of propositional variables $\mathcal{X}$. In the formalization, we take $\mathcal{X} \triangleq \mathbb{N}$. The set of assertions $\mathcal{A}$ is given by

$$\begin{array}{rcll} \phi, \psi, \xi \in \mathcal{A} & ::= & p \in \mathcal{X} & \text{propositional variable} \\ & | & \bot & \text{false} \\ & | & \phi \vee \psi & \text{disjunction} \\ & | & \phi \wedge \psi & \text{conjunction} \\ & | & \phi \Rightarrow \psi & \text{implication} \end{array}$$

We write $\top$ (resp. $\neg\phi$, $\phi \Leftrightarrow \psi$) for $\bot \Rightarrow \bot$ (resp. $\phi \Rightarrow \bot$, $(\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$).

The set of assertions is defined in Coq by the type `prop`.

Denotation of assertions - we now define the denotation of an assertion w.r.t. a valuation. A value is any function ν from $\mathcal{X}$ to $\mathbb{B}$ ($\triangleq \{\top, \bot\}$). The denotation of an assertion ϕ w.r.t. a valuation ν, written $[\![\phi]\!]_\nu$ is defined as follows:

$$\left|\begin{array}{rl} [\![p]\!]_\nu = & \nu(p) \\ [\![\bot]\!]_\nu = & \bot \\ [\![\phi \vee \psi]\!]_\nu = & \begin{cases} \top & \text{if } [\![\phi]\!]_\nu = \top \text{ or } [\![\psi]\!]_\nu = \top \\ \bot & \text{otherwise} \end{cases} \\ [\![\phi \wedge \psi]\!]_\nu = & \begin{cases} \top & \text{if } [\![\phi]\!]_\nu = \top \text{ and } [\![\psi]\!]_\nu = \top \\ \bot & \text{otherwise} \end{cases} \\ [\![\phi \Rightarrow \psi]\!]_\nu = & \begin{cases} \top & \text{if } [\![\phi]\!]_\nu = \bot \text{ or } [\![\psi]\!]_\nu = \top \\ \bot & \text{otherwise} \end{cases} \end{array}\right.$$

We say that an assertion ϕ is satisfiable under a valuation ν if $[\![\phi]\!]_\nu = \top$. We say that an assertion is valid if it is satisfiable under any valuation.

Q1. Fill the Coq definition `sem : valuation → prop → bool` s.t. `sem v p` returns the denotation of `p` w.r.t the valuation `v`.

Base rules

$$\frac{p \in \Gamma}{\Gamma \vdash p}\ \textsc{Axiom} \qquad \frac{\neg p, \Gamma \vdash \bot}{\Gamma \vdash p}\ \textsc{Absurd}$$

Introduction rules

$$\frac{\Gamma \vdash p \qquad \Gamma \vdash q}{\Gamma \vdash p \wedge q}\ \wedge\text{-I} \qquad \frac{\Gamma \vdash p}{\Gamma \vdash p \vee q}\ \vee\text{-L-I} \qquad \frac{\Gamma \vdash q}{\Gamma \vdash p \vee q}\ \vee\text{-R-I} \qquad \frac{p, \Gamma \vdash q}{\Gamma \vdash p \Rightarrow q}\ \Rightarrow\text{-I}$$

Elimination rules

$$\frac{\Gamma \vdash \bot}{\Gamma \vdash p}\ \bot\text{-E} \qquad \frac{\Gamma \vdash p \wedge q}{\Gamma \vdash p}\ \wedge\text{-L-E} \qquad \frac{\Gamma \vdash p \wedge q}{\Gamma \vdash q}\ \wedge\text{-R-E} \qquad \frac{\Gamma \vdash p \vee q \qquad p, \Gamma \vdash r \qquad q, \Gamma \vdash r}{\Gamma \vdash r}\ \vee\text{-E}$$

$$\frac{\Gamma \vdash p \qquad \Gamma \vdash p \Rightarrow q}{\Gamma \vdash q}\ \Rightarrow\text{-E}$$

Figure 1: Natural deduction inference rules

Natural deduction - we describe description a proof calculus for assertions called *Natural Deduction*. A judgment of natural deduction is of the form $\Gamma \vdash \phi$ where Γ is a list of assertions $(\phi_1, \phi_2, \ldots)$ called an *environment*.

Derivation of judgment in natural deduction is described by a set of inference rules that we give in Figure 1. It is defined in Coq using the inductive predicate `nd : list prop → prop → Prop`.

We say that an assertion ϕ is provable under Γ if $\phi \vdash \Gamma$. If Γ is empty, we simply say that ϕ is provable. We also extend the notion of satisfiability to environments: we say that a valuation ν satisfies an environment Γ if it satisfies all its assertions, i.e. if ν satisfies any assertion $\phi \in \Gamma$.

We start by proving a weakening lemma for natural deduction derivations. We say that an environment Γ is weaker than an environment Δ (written $\Gamma \preceq \Delta$) if $\forall \phi.\, \phi \in \Gamma \Rightarrow \phi \in \Delta$.

Q2. Prove that $\cdot \vdash \cdot$ is monotonous w.r.t. $\preceq$, i.e. if $\Gamma \preceq \Delta$ and $\Gamma \vdash \phi$, then $\Delta \vdash \psi$. (Lemma `subenv_nd` in the file)

We now prove the correctness of natural deduction w.r.t. the denotation of assertions that is expressed as follows: if ϕ is provable under Γ, then any valuation that satisfies Γ must satisfy ϕ.

Q3. Prove the correctness of natural deduction:

```
Lemma correctness (env : list prop) (p : prop) :
    nd env p
  → forall v, (forall q, In q env → sat v q)
  → sat v p.
```

Deciding validity of assertions - The aim of that section is to write and prove correct a program (or decision procedure) for deciding if an assertion is valid. For that, we will write two normalization procedures for transforming assertions from their general form to a more restricted one. All these transformations will preserve the satisfiability of assertions. Then, we will write and prove correct (and complete) a decision procedure for the satisfiability of assertions in restricted form. Finally, tying all up, we will derive a correct procedure for the satisfiability of assertions in general form.

The set of $\mathbb{I}$-assertions is given by

$$\begin{array}{rcll} \Phi, \Psi, \Xi \in \mathbb{I} & ::= & p \in \mathcal{X} & \text{propositional variable} \\ & | & b \in \mathbb{B} & \text{propositional constant} \\ & | & \text{if } \Phi \text{ then } \Psi \text{ else } \Xi & \text{if assertion} \end{array}$$

The set of $\mathbb{I}$-assertions is defined in Coq by the type `ifForm`.

As for general assertions, we define a notion of denotation of a $\mathbb{I}$-assertion Φ w.r.t a valuation ν (denoted by $[\![\Phi]\!]_\nu$):

$$\left|\begin{aligned} [\![p]\!]_\nu &= \nu(p) \\ [\![\top]\!]_\nu &= \top \\ [\![\bot]\!]_\nu &= \bot \\ [\![\text{if } \Phi \text{ then } \Psi \text{ else } \Xi]\!]_\nu &= \begin{cases} [\![\Psi]\!]_\nu & \text{if } [\![\Phi]\!]_\nu = \top \\ [\![\Xi]\!]_\nu & \text{otherwise} \end{cases} \end{aligned}\right.$$

Q4. Fill the Coq definition `ifsem : valuation → ifForm → bool` s.t. `ifsem v p` returns the denotation of the $\mathbb{I}$-assertion `p` w.r.t the valuation `v`.

Q5. Write a function `ifForm_of_prop : prop → ifForm` that transforms a general assertion to an $\mathbb{I}$-assertion. Keep in mind that this transformation should keep satisfiability of assertions.

Q6. Prove the correctness of your transformation, i.e.

```
Lemma ifForm_correct (v : valuation) (p : prop) :
  sem v p = ifsem v (ifForm_of_prop p).
```

An $\mathbb{I}$-assertion Φ is said to be *normalized* if all the conditions of the if-then-else constructs are propositional variables, i.e. if it is of the form

$$\begin{array}{rcll} \hat{\Phi}, \hat{\Psi} \in \mathbb{K} & ::= & p \in \mathcal{X} & \text{propositional variable} \\ & | & b \in \mathbb{B} & \text{propositional constant} \\ & | & \text{if } p \text{ then } \hat{\Phi} \text{ else } \hat{\Psi} & \text{normalized if assertion} \end{array}$$

We write $\mathbb{K}$ for the set of normalized $\mathbb{I}$-assertions. The notion of denotation is unchanged from $\mathbb{I}$-assertions to $\mathbb{K}$-assertions. The set of $\mathbb{K}$-assertions is defined in Coq by the type `nifForm`. (Note that it is not a subtype of `ifForm`)

Q7. Fill the Coq definition `nifsem : valuation → nifForm → bool` s.t. `nifsem v p` returns the denotation of the $\mathbb{K}$-assertion `p` w.r.t the valuation `v`.

We now define a procedure for normalizing $\mathbb{I}$-assertions. This procedure relies of two inductive functions. One ($(\!|\Phi|\!)$) that normalized a $\mathbb{I}$-assertion, and one ($\lfloor\!\lfloor \text{if } \hat{\Phi} \text{ then } \hat{\Psi} \text{ else } \hat{\Xi} \rfloor\!\rfloor$) that normalized if-then-else constructs whose sub-formulas are already $\mathbb{K}$-assertions.

$$\begin{aligned} (\!|p|\!) &= p \\ (\!|b|\!) &= b \\ (\!|\text{if } \Phi \text{ then } \Psi \text{ else } \Xi|\!) &= \lfloor\!\lfloor \text{if } (\!|\Phi|\!) \text{ then } (\!|\Psi|\!) \text{ else } (\!|\Xi|\!) \rfloor\!\rfloor \end{aligned}$$

$$\begin{aligned} \lfloor\!\lfloor \text{if } p \text{ then } \hat{\Phi} \text{ else } \hat{\Psi} \rfloor\!\rfloor &= \text{if } p \text{ then } \hat{\Phi} \text{ else } \hat{\Psi} \\ \lfloor\!\lfloor \text{if } \top \text{ then } \hat{\Phi} \text{ else } \hat{\Psi} \rfloor\!\rfloor &= \hat{\Phi} \\ \lfloor\!\lfloor \text{if } \bot \text{ then } \hat{\Phi} \text{ else } \hat{\Psi} \rfloor\!\rfloor &= \hat{\Psi} \\ \lfloor\!\lfloor \text{if } (\text{if } \hat{\Phi} \text{ then } \hat{\Psi} \text{ else } \hat{\Xi}) \text{ then } \hat{\Psi}' \text{ else } \hat{\Xi}' \rfloor\!\rfloor &= \\ \text{if } \hat{\Phi} \text{ then } \lfloor\!\lfloor \text{if } \hat{\Psi} \text{ then } \hat{\Psi}' \text{ else } \hat{\Xi}' \rfloor\!\rfloor &\text{ else } \lfloor\!\lfloor \text{if } \hat{\Xi} \text{ then } \hat{\Psi}' \text{ else } \hat{\Xi}' \rfloor\!\rfloor \end{aligned}$$

Q8. Define in Coq the two normalization procedures:

```
Fixpoint normif (c t f : nifForm) {struct c} : nifForm.

Fixpoint norm (p : ifForm) {struct p} : nifForm.
```

Q9. Prove that the normalization procedure is correct, i.e.

```
Lemma normif_correct (v : valuation) (c t f : nifForm) :
  nifsem v (normif c t f) =
    if nifsem v c then nifsem v t else nifsem v f.

Lemma norm_correct (v : valuation) (p : ifForm) :
  nifsem v (norm p) = ifsem v p.
```

The decision procedure - we here give the Coq code that decide if a $\mathbb{K}$-assertion is valid or not w.r.t a partial valuation:

```
Definition xt (v : nat → option bool) (x : nat) (b : bool) :=
  fun y ⇒ if beq_nat x y then Some b else v y.

Fixpoint nifForm_tauto_r (v : nat → option bool) (p : nifForm) :=
  match p with
  | PNIVar x ⇒ match v x with Some true ⇒ true | _ ⇒ false end
  | PNIConst b ⇒ b

  | PNIIf x t f ⇒
    match v x with
    | Some true ⇒ nifForm_tauto_r v t
    | Some false ⇒ nifForm_tauto_r v f
    | None ⇒
          nifForm_tauto_r (xt v x true ) t
       && nifForm_tauto_r (xt v x false) f
    end
  end.

Definition nifForm_tauto p := nifForm_tauto_r (fun _ ⇒ None) p.
```

We ask you to prove the correctness and completeness of the procedure.

Q10. Prove the correctness of the procedure:

```
Lemma nifForm_tauto_r_correct (xv : nat → option bool) (p : nifForm) :
     nifForm_tauto_r xv p = true
  → forall v, (forall x b, xv x = Some b → v x = b)
  → nifsem v p = true.

Lemma nifForm_tauto_correct (p : nifForm) :
  nifForm_tauto p = true → forall v, nifsem v p = true.
```

Q11. Prove the completeness of the procedure:

```
Lemma nifForm_tauto_r_complete (xv : nat → option bool) (p : nifForm) :
     nifForm_tauto_r xv p = false
  → exists v, (forall x b, xv x = Some b → v x = b)
             /\ nifsem v p = false.

Lemma nifForm_tauto_complete (p : nifForm) :
  nifForm_tauto p = false → exists v, nifsem v p = false.
```

We can now all tie up, writing and proving correct a decision procedure for the validity of assertions.

Q12. Write a Coq function `is_tautology : prop → bool` that decides if a assertion is valid or not.

Q13. Prove that your decision procedure is correct and complete:

```
Lemma is_tautology_correct (p : prop) :
  is_tautology p = true → valid p.

Lemma is_tautology_complete (p : prop) :
  is_tautology p = false → exists v, sem v p = false.
```