Le Raisonnement Logique dans l'Assistant de Preuve Coq

Travaux Pratiques, Automates et Logiques, Licence Info, Université de Lille 1

Reynald Affeldt

National Institute of Advanced Industrial Science and Technology

3 mars 2016

Prouver les lemmes de ce document, sans avoir recours aux tactiques automatiques de Coq (`assumption`, `trivial`, `auto`, `intuition`, `tauto`, `firstorder`, etc.). Les tactiques vues en cours suffisent (on les rappellent brièvement Table 1, voir le cours pour une syntaxe plus précise). En particulier, on préfèrera `apply` à `cut` pour éliminer l'implication. Dans un premier temps, on peut ignorer les exemples de la logique du premier ordre. On pourra utiliser la logique classique en cas de besoin (en particulier le lemme `bottom_c` vu en cours).

Déduction Naturelle (une seule règle pour illustration)	Définition	Tactique	
		Introduction	Élimination
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to_i$		`intros`	`apply`
$\dfrac{\Gamma \vdash \bot}{\Gamma \vdash A} \bot_e$	`Inductive False : Prop := .`		`destruct`

$\dfrac{\Gamma \vdash A \qquad \Gamma \vdash B}{\Gamma \vdash A \land B} \land_i$	`Inductive and`
	`(A B : Prop) : Prop :=`
	`| conj : A -> B -> A /\ B.`
		`split`	`destruct as [...]`

$\dfrac{\Gamma \vdash A}{\Gamma \vdash A \lor B} \lor_i^g$
$\dfrac{\Gamma \vdash B}{\Gamma \vdash A \lor B} \lor_i^d$
	`Inductive or`
	`(A B : Prop) : Prop :=`
	`| or_introl : A -> A \/ B`
	`| or_intror : B -> A \/ B`
		`left`
		`right`
			`destruct as [...|...]`

$\dfrac{\Gamma, A \vdash \bot}{\Gamma \vdash \neg A} \neg_i$	`Definition not (A:Prop) :=` `A -> False.`	`intros`	`apply`

$\dfrac{\Gamma \vdash P[x := t]}{\Gamma \vdash \exists x, P\, x} \exists_i$	`Inductive ex (A : Type)`
	`(P : A -> Prop) : Prop :=`
	`| ex_intro :`
	`forall x : A, P x ->`
	`exists x, P x`
		`exists`	`destruct as [...]`

$\dfrac{\Gamma \vdash A \quad x \text{ n'est pas libre dans } \Gamma}{\Gamma \vdash \forall x. A} \forall_i$		`intros`	`apply`
$\dfrac{\Gamma, \neg A \vdash \bot}{\Gamma \vdash A} \bot_c$	`Lemma bottom_c (A : Prop) :` `((~A) -> False) -> A.`	`apply bottom_c.` `intros na.`	

Table 1 – Rappel des tactiques élémentaires de Coq

1 Familiarisation avec l'interface de CoqIDE

Reproduire l'exemple de l'axiome S de Hilbert vu en cours. L'usage est de commencer un script de preuve par la commande `Proof`. Une commande se termine par un point. Pour exécuter une commande, clicker . Pour revenir une étape en arrière, clicker . Pour exécuter les commandes jusqu'au curseur, clicker . Un script de preuve se termine en général par `Qed`.

Les entrées sont essentiellement en ASCII (même si elles apparaissent sous forme de symboles LaTeX dans ce document). On écrit `->` pour $\to$, `/\` pour $\wedge$, `~` pour $\neg$, etc.

`Lemma` *hilbertS* $(A\ B\ C : \texttt{Prop})$:
$(A \to B \to C) \to (A \to B) \to A \to C.$

2 Exemples tirés du cours

2.1 [Pie16, Slide 58]

`Lemma` *exo1* $(P\ Q : \texttt{Prop}) : P \to (Q \to P).$

`Lemma` *exo2* $(P\ Q : \texttt{Prop}) : P \to (\neg P \to Q).$

`Lemma` *exo3* $(P\ Q\ R : \texttt{Prop}) : (P \to Q) \to ((Q \to R) \to (P \to R)).$

Contraposée :

`Lemma` *exo4* $(P\ Q : \texttt{Prop}) : (P \to Q) \to (\neg Q \to \neg P).$

On prépare un lemme correspondant à l'absurdité classique. On charge d'abord la librairie standard `Classical` de Coq :

`Require Import` *Classical*.

On peut utiliser la loi de double négation (lemme `NNPP`) ou le tiers exclus (lemme `classic`) provenant de la librairie `Classical`.

`Lemma` *bottom_c* $(A : \texttt{Prop}) : ((\tilde{}A) \to \mathit{False}) \to A.$

Utiliser l'absurdité classique pour le lemme suivant :

`Lemma` *exo5* $(P\ Q : \texttt{Prop}) : (\neg Q \to \neg P) \to (P \to Q).$

Loi de double négation (une caractérisation de la logique classique) :

`Lemma` *exo6* $(P : \texttt{Prop}) : \neg \neg P \to P.$

`Lemma` *exo7* $(P : \texttt{Prop}) : P \to \neg \neg P.$

`Lemma` *exo8* $(P\ Q\ R : \texttt{Prop}) : (P \to (Q \to R)) \to (P \wedge Q \to R).$

`Lemma` *exo9* $(P\ Q\ R : \texttt{Prop}) : (P \wedge Q \to R) \to (P \to (Q \to R)).$

On peut utiliser les lemmes précédents pour l'exemple suivant :

`Lemma` *exo10* $(P : \texttt{Prop}) : P \wedge \neg P \to \mathit{False}.$

`Lemma` *exo11* $(P : \texttt{Prop}) : \mathit{False} \to P \wedge \neg P.$

2.2 [Pie16, Slide 68]

Loi de De Morgan. Le sens $\leftarrow$ est équivalent à la logique classique. Utiliser l'absurdité classique.

`Lemma` *exo12* $(P\ \ Q : \texttt{Prop}) : P \vee Q \leftrightarrow \neg\ (\neg\ P \wedge \neg\ Q)$.

`Lemma` *exo13* $(P : \texttt{Prop}) : \neg\ P \leftrightarrow (P \rightarrow \mathit{False})$.

`Lemma` *exo14* $(P\ \ Q : \texttt{Prop}) : (P \leftrightarrow Q) \leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$.

3 Exemples tirés de [DNR03]

On reproduit les exemples de démonstration de [DNR03, Section 1.3.4] en structurant les scripts à l'aide des bullets `-`, `+`, `*` pour faire apparaître la structure d'arbre.

3.1 [DNR03, Exemple 1.3.4, p. 33].

Rappel : `A <-> B` est défini comme `(A -> B) /\ (B -> A)`.

`Lemma` *exemple134* $(A\ \ B\ \ C : \texttt{Prop}) : (A \wedge B \rightarrow C) \leftrightarrow (A \rightarrow B \rightarrow C)$.

3.2 [DNR03, Exemple 1.3.5, p. 34].

`Lemma` *exemple135* $(A\ \ B\ \ C : \texttt{Prop}) : (C \rightarrow A) \vee (C \rightarrow B) \rightarrow (C \rightarrow A \vee B)$.

3.3 [DNR03, Exemple 1.3.6, p. 34].

`Lemma` *exemple_136* $(X : \texttt{Type})\ (A\ \ B : X \rightarrow \texttt{Prop})$:
$((\forall\ x,\ A\ x) \vee (\forall\ x,\ B\ x)) \rightarrow \forall\ x,\ A\ x \vee B\ x$.

3.4 [DNR03, Exemple 1.3.7, p. 34].

`Lemma` *exemple_137* $(X : \texttt{Type})\ (A\ \ B : X \rightarrow \texttt{Prop})$:
$(\exists\ x,\ A\ x \wedge B\ x) \rightarrow ((\exists\ x,\ A\ x) \wedge (\exists\ x,\ B\ x))$.

3.5 [DNR03, Exemple 1.3.8, p. 35].

Pour ce nouvel exemple de la loi de De Morgan, on utilise l'absurdité classique :

`Lemma` *exemple_138* $(A\ \ B : \texttt{Prop}) : \neg\ (A \wedge B) \rightarrow (\neg\ A \vee \neg\ B)$.

On peut aussi réutiliser des lemmes déjà prouvés.

`Lemma` *exemple_138'* $(A\ \ B : \texttt{Prop}) : \neg\ (A \wedge B) \rightarrow (\neg\ A \vee \neg\ B)$.

3.6 [DNR03, Exemple 1.3.9, p. 35].

Pour rester dans l'esprit du livre, on utilisera la fonction `eq_ind` plutôt que `rewrite`.

`Lemma` *exemple_139* $(X : \texttt{Type}) : \forall\ (x1\ \ x2 : X),\ x1 = x2 \rightarrow x2 = x1$.

3.7 [DNR03, Exemple 1.3.10, p. 36].

Même remarque que ci-dessus.

`Lemma` *exemple_140* $(X : \texttt{Type}) : \forall\ (x1\ x2\ x3 : X),\ x1 = x2 \wedge x2 = x3 \rightarrow x1 = x3.$

4 Encodage des connectives logiques sans types inductifs

On a utilisé la théorie des types pour encoder l'implication (comme le type d'une fonction) et les types inductifs pour encoder les connectives de la logique. On peut en fait encoder ces connectives sans avoir recours aux types inductifs. On va encoder les connectives logiques avec le produit de la théorie des types et montrer l'équivalence avec les définitions de la librairie standard. Utiliser la tactique `unfold` pour développer une définition.

4.1 Exemple du faux

On peut définir faux comme une proposition qui rend toutes les propositions vraies :

`Definition` *FALSE* $: \texttt{Prop} := \forall\ (P : \texttt{Prop}),\ P.$

```
Goal FALSE.
unfold FALSE.
intros p.
Abort.
```

`Lemma` *FALSE_False* $: FALSE \leftrightarrow False.$

4.2 Définitions sans types inductifs

Encodage de second-ordre (à cause de la quantification sur toutes les propositions) de la conjonction. Il y aussi l'encodage de premier ordre $A \wedge B = \neg(A \rightarrow \neg B)$ mais la logique devient classique.

`Definition` *AND* $(A\ B : \texttt{Prop}) := \forall\ (P : \texttt{Prop}),\ (A \rightarrow B \rightarrow P) \rightarrow P.$

`Definition` *OR* $(A\ B : \texttt{Prop}) := \forall\ (P : \texttt{Prop}),\ ((A \rightarrow P) \rightarrow (B \rightarrow P) \rightarrow P).$

`Definition` *EX* $(A : \texttt{Type})\ (P : A \rightarrow \texttt{Prop}) := \forall\ (Q : \texttt{Prop}),\ (\forall\ a,\ P\ a \rightarrow Q) \rightarrow Q.$

`Definition` *EQ* $(A : \texttt{Type})\ (a\ a' : A) := \forall\ (P : A \rightarrow \texttt{Prop}),\ P\ a \rightarrow P\ a'.$

4.3 Équivalence avec la librairie standard

On retrouve les règles d'introduction et d'élimination sous forme de lemmes.

`Lemma` *SPLIT* $(A\ B : \texttt{Prop}) : A \rightarrow B \rightarrow AND\ A\ B.$

`Lemma` *PROJ1* $(A\ B : \texttt{Prop}) : AND\ A\ B \rightarrow A.$

`Lemma` *PROJ2* $(A\ B : \texttt{Prop}) : AND\ A\ B \rightarrow B.$

`Lemma` *ORINTROL* $(A\ B : \texttt{Prop}) : A \rightarrow OR\ A\ B.$

`Lemma` *ORINTROR* $(A\ B : \texttt{Prop}) : B \rightarrow OR\ A\ B.$

On en déduit l'équivalence avec les définitions inductives.

`Lemma` *AND_and* $(A\ B : \texttt{Prop}) : AND\ A\ B \leftrightarrow A \wedge B.$

`Lemma` OR_or $(A\ B : \texttt{Prop}) : OR\ A\ B \leftrightarrow A \vee B.$

`Lemma` EX_exists $(A : \texttt{Type})\ (P : A \rightarrow \texttt{Prop}) : EX\ A\ P \leftrightarrow \exists\ a,\ P\ a.$

Rappel : La réécriture s'effectue avec la tactique `rewrite`.

`Lemma` EQ_eq $(A : \texttt{Type})\ (a\ a' : A) : EQ\ \_\ a\ a' \leftrightarrow a = a'.$

5 Question subsidiaire

`Lemma` $subsidiaire$ $(A : \texttt{Prop}) : \neg\ \neg\ (A \vee \neg\ A).$

Références

[DNR03] René David, Karim Nour, and Christophe Raffalli, *Introduction à la logique*, 2ème ed., Dunod, 2003.

[Pie16] Thomas Pietrzak, *Logique—logique propositionnelle—*, Licence Informatique, Université de Lille 1 Sciences et Technologies, 2016, http://www.thomaspietrzak.com/download.php?f=CoursLogique0.pdf.