Le Raisonnement Logique dans l'Assistant de Preuve Coq (draft)

Licence Info, Automates et Logiques, Travaux Pratiques

Reynald Affeldt
National Institute of Advanced Industrial Science and Technology

19 février 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 (voir Table 1). 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.

Déduction Naturelle (Règle d'introduction uniquement)	Définition inductive	Tactique	
		Introduction	Élimination
$\dfrac{\Gamma, A \vdash B}{\Gamma \vdash A \to B} \to_i$		`intros`	`apply`
	`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 [...|...]`

$\dfrac{\Gamma \vdash P[x := t]}{\Gamma \vdash \exists x, P\,x} \exists_i$	`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`

Table 1 – Rappel des tactiques élémentaires de Coq

On pourra utiliser la logique classique en cas de besoin (en particulier le lemme `NNPP` qui correspond à la loi de double négation), d'où la commande suivante :

```
Require Import Classical.
```

1 Exemples tirés du cours

1.1 [Pie16, Slide 58]

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

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

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

Contraposée :

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

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

Une caractérisation de la logique classique :

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

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

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

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

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

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

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

1.2 [Pie16, Slide 68]

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

`Lemma` *exo12* $(P\ Q : \texttt{Prop}) : P \lor Q \leftrightarrow \neg\ (\tilde{}\ P \land \neg\ Q)$.

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

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

2 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.

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

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

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

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

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

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

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

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

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

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

Pour ce nouvel exemple de la loi de De Morgan, on utilise le lemme NNPP de la librairie standard de Coq :

Lemma *exemple_138* $(A\ B : \texttt{Prop}) : \neg\ (A \land B) \to (\tilde{}\ A \lor \neg\ B).$

On peut aussi réutiliser des lemmes déjà prouvés.

Lemma *exemple_138'* $(A\ B : \texttt{Prop}) : \neg\ (A \land B) \to (\tilde{}\ A \lor \neg\ B).$

2.6 [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* $(X : \texttt{Type}) : \forall\ (x1\ x2 : X),\ x1 = x2 \to x2 = x1.$

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

Même remarque que ci-dessus.

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

3 Encodage des connectives logiques sans types inductifs

On a utilisé la théorie 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.

3.1 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* : $FALSE \leftrightarrow False.$

3.2 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 $A \wedge B = \neg(A \to \neg B)$ mais la logique devient classique.

`Definition` AND (A B : `Prop`) := $\forall$ (P : `Prop`), ($A \to B \to P$) $\to P$.

`Definition` OR (A B : `Prop`) := $\forall$ (P : `Prop`), (($A \to P$) $\to$ ($B \to P$) $\to P$).

`Definition` EX (A : `Type`) (P : $A \to$ `Prop`) := $\forall$ (Q : `Prop`), ($\forall$ a, P $a \to Q$) $\to Q$.

`Definition` EQ (A : `Type`) (a a' : A) := $\forall$ (P : $A \to$ `Prop`), P $a \to P$ a'.

3.3 Équivalence avec la librairie standard

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

`Lemma` $SPLIT$ (A B : `Prop`) : $A \to B \to AND$ A B.

`Lemma` $PROJ1$ (A B : `Prop`) : AND A $B \to A$.

`Lemma` $PROJ2$ (A B : `Prop`) : AND A $B \to B$.

`Lemma` $ORINTROL$ (A B : `Prop`) : $A \to OR$ A B.

`Lemma` $ORINTROR$ (A B : `Prop`) : $B \to OR$ A B.

On en déduit l'équivalence avec les définitions inductives.

`Lemma` AND_and (A B : `Prop`) : AND A $B \leftrightarrow A \wedge B$.

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

`Lemma` EX_exists (A : `Type`) (P : $A \to$ `Prop`) : EX A $P \leftrightarrow \exists$ a, P a.

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

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

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.