

定理証明支援系 CoQ での 理論的なリーゾニング

演習問題, 集中講義 千葉大学大学院

アフエルト レナルド

産業技術総合研究所

2017年01月25

本ファイルの補題を証明する. ただし, CoQ の自動タクティクを使わない (`assumption`, `trivial`, `auto`, `intuition`, `tauto`, `firstorder`, etc.). 授業で見たタクティクは十分である (まとめ: 表 1, 細かいシンタクスに関して授業に参考). 特に, 含意の除去の際, `cut` より `apply` を使う. 必要なら, 古典論理を使うことができる (特に, 授業で見た `bottom_c` 補題).



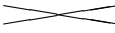



自然演繹 (ルールを一つしかい表示しない)	定義	タクティク	
		導入	除去
$\frac{\Gamma, A \vdash B}{\Gamma \vdash A \rightarrow B} \rightarrow_i$		<code>intros</code>	<code>apply</code>
$\frac{\Gamma \vdash \perp}{\Gamma \vdash A} \perp_e$	<code>Inductive False : Prop := .</code>		<code>destruct</code>
$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \wedge B} \wedge_i$	<code>Inductive and (A B : Prop) : Prop := conj : A -> B -> A /\ B.</code>	<code>split</code>	<code>destruct as [...]</code>
$\frac{\Gamma \vdash A}{\Gamma \vdash A \vee B} \vee_i^l$ $\frac{\Gamma \vdash B}{\Gamma \vdash A \vee B} \vee_i^r$	<code>Inductive or (A B : Prop) : Prop := or_introl : A -> A \/ B or_intror : B -> A \/ B</code>	<code>left</code> <code>right</code>	<code>destruct as [... ...]</code>
$\frac{\Gamma, A \vdash \perp}{\Gamma \vdash \neg A} \neg_i$	<code>Definition not (A:Prop) := A -> False.</code>	<code>intros</code>	<code>apply</code>
$\frac{\Gamma \vdash P[x := t]}{\Gamma \vdash \exists x, P x} \exists_i$	<code>Inductive ex (A : Type) (P : A -> Prop) : Prop := ex_intro : forall x : A, P x -> exists x, P x</code>	<code>exists</code>	<code>destruct as [...]</code>
$\frac{\Gamma \vdash A}{\Gamma \text{ の中に } x \text{ は自由変数ではない}} \Gamma \vdash \forall x. A \quad \forall_i$		<code>intros</code>	<code>apply</code>
$\frac{\Gamma, \neg A \vdash \perp}{\Gamma \vdash A} \perp_c$	<code>Lemma bottom_c (A : Prop) : ((~A) -> False) -> A.</code>	<code>apply bottom_c.</code> <code>intros na.</code>	

表 1: CoQ の基本的なタクティクのまとめ

1 CoqIDE のインタフェースの基本的な使い方

授業で見た Hilbert の S 公理を再現する. 通常, スクリプトはコマンド `Proof` で開始する. コマンドはピリオドで終わる.  をクリックすると, コマンドは実行される.  をクリックすると, 前のステップに戻る.  をクリックすると, カーソルまでのコマンドを実行する. 通常, スクリプトは `Qed` で終わる.

(本資料では \LaTeX のシンボルとして表示するが,) 入力基本的には ASCII で書く. \rightarrow は `->`, \wedge は `\wedge`, \neg は `~`, 等と書く.

```
Lemma hilbertS (A B C : Prop) :  
  (A → B → C) → (A → B) → A → C.
```

2 [Pie16] からの例

2.1 [Pie16, Slide 58]

```
Lemma exo1 (P Q : Prop) : P → (Q → P).
```

```
Lemma exo2 (P Q : Prop) : P → (¬ P → Q).
```

```
Lemma exo3 (P Q R : Prop) : (P → Q) → ((Q → R) → (P → R)).
```

対偶:

```
Lemma exo4 (P Q : Prop) : (P → Q) → (¬ Q → ¬ P).
```

古典矛盾に当たる補題を用意する. まず, Coq の標準ライブラリから `Classical` を設定する:

```
Require Import Classical.
```

`Classical` ライブラリからの二重否定 (NNPP 補題) または排中律 (`classic` 補題) を利用できる.

```
Lemma bottom_c (A : Prop) : ((¬ A) → False) → A.
```

古典矛盾を用いて, 次の補題を証明する:

```
Lemma exo5 (P Q : Prop) : (¬ Q → ¬ P) → (P → Q).
```

二重否定 (古典論理の一つの定義):

```
Lemma exo6 (P : Prop) : ¬ ¬ P → P.
```

```
Lemma exo7 (P : Prop) : P → ¬ ¬ P.
```

```
Lemma exo8 (P Q R : Prop) : (P → (Q → R)) → (P ∧ Q → R).
```

```
Lemma exo9 (P Q R : Prop) : (P ∧ Q → R) → (P → (Q → R)).
```

次の補題を証明するために, 上記の補題を利用できる:

```
Lemma exo10 (P : Prop) : P ∧ ¬ P → False.
```

```
Lemma exo11 (P : Prop) : False → P ∧ ¬ P.
```

2.2 [Pie16, Slide 68]

De Morgan の法則. \leftarrow の方向は古典論理による. 古典矛盾を利用する.

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

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

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

3 [DNR03] からの例

[DNR03, Section 1.3.4] の証明を再現する. ただし, ビュレット $\neg, +, *$ を用いて, 証明木の構造を明確にする.

3.1 [DNR03, 例 1.3.4, p. 33].

メモ: $A \leftrightarrow B$ は $(A \rightarrow B) \wedge (B \rightarrow A)$ として定義されている.

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

3.2 [DNR03, 例 1.3.5, p. 34].

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

3.3 [DNR03, 例 1.3.6, p. 34].

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

3.4 [DNR03, 例 1.3.7, p. 34].

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

3.5 [DNR03, 例 1.3.8, p. 35].

次の De Morgan 法則の例を証明するために, 古典矛盾を利用する:

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

証明済みの補題を利用することもできる.

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

3.6 [DNR03, 例 1.3.9, p. 35].

[DNR03] から離れないように, “`!rewrite!`”の代わりに, 関数“`!eq_ind!`”を利用する.

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

3.7 [DNR03, 例 1.3.10, p. 36].

上記と同様.

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

4 帰納的型を利用せず, 論理結合子を形式化

含意を表現するために, 型理論による関数の型を使った. 論理結合子を表現するために, 帰納的型を使った. 実は, 帰納的型を使わない形式化もある.

今回, 型理論の `product` を用いて論理結合子を表現し, 標準ライブラリの定義と比較する. 定義を展開するために, タクティク `unfold` を利用できる.

4.1 矛盾の例

矛盾は何でもの命題の証明を返すとして定義できる:

```
Definition FALSE : Prop :=  $\forall (P : Prop), P$ .
```

```
Goal FALSE.
```

```
unfold FALSE.
```

```
intros p.
```

```
Abort.
```

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

4.2 帰納的型を使わない定義

論理積の二階形式化 (任意の命題に対する全称記号のため). 一階形式化 $A \wedge B = \neg(A \rightarrow \neg B)$ もあるが, 論理が古典になる.

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

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

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

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

4.3 標準ライブラリとの比較

導入と除去ルールは補題として再現できる.

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

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

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

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

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

帰納的型を用いた定義と同値であることを証明できる.

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

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

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

メモ: 書き換えをタクティク `rewrite` で行う.

Lemma *EQ_eq* ($A : \text{Type}$) ($a a' : A$) : $\text{EQ } a a' \leftrightarrow a = a'$.

5 補足

Lemma *subsidaire* ($A : \text{Prop}$) : $\neg \neg (A \vee \neg A)$.

参考文献

[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>.