Coqゼミ　第4回

浅井研究室（担当：廣田知子）

2008年 7月 7日

1　命題論理式の真と偽

Coqでは、True, False で表します。

```
Coq < Check True.
True : Prop.
Coq < Check False.
False : Prop.
```

2　implication以外の論理演算

以下では、「ならば →」以外の論理演算子の使い方を説明します。各論理演算子には以下の二つの規則が存在します;

introduction の規則
　論理演算を（矢印の右側に）導入する規則

elimination の規則
　論理演算を消去する規則

2.1　and

A /\ Bはand A Bと書いているのと同じ。

introduction の規則：

```
Coq < Check conj.
conj : forall A B : Prop,
       A -> B -> A /\ B
```

Goalが ... -> A /\ Bだったときに、
intros; apply conj.
とすると、GoalがAとBになる。(...の部分はintrosで仮定に組み込まれている。)これはよく使われるので、**split** というtacticで行える。

elimination の規則：

```
Coq < Check and_ind.
and_ind : forall A B P : Prop,
          (A -> B -> P) -> A /\ B
                        -> P
```

仮定にA /\ Bがあって、それの名前がHだったとき、
elim H.
とすると、GoalのPがA -> B -> Pになる。(その後、introsを使ってA, Bというばらばらになった仮定の下でPを証明すれば良い。)

2.2　or

A \/ Bはor A Bと書いているのと同じ。

introduction の規則：

```
Coq < Check or_introl.
or_introl : forall A B : Prop, A -> A \/ B

Coq < Check or_intror.
or_intror : forall A B : Prop, B -> A \/ B
```

Goalが ... -> A \/ Bだったときに、
intros; apply or_introl.
とすると、Goal が A になる。(... の部分は intros で仮定に組み込まれている。) これはよく使われるので、**left** という tactic で行える。同様に、intros; apply or_intror. は **right** である。

elimination の規則：

```
Coq < Check or_ind.
or_ind : forall A B P : Prop,
         (A -> P) -> (B -> P) -> A \/ B -> P
```

仮定にA \/ Bがあって、それの名前がHだったとき、
elim H.
とすると、GoalのPがA -> PとB -> Pという二つのGoalになる。

2.3　not

~Aはnot Aと書いているのと同じ。

introduction の規則はない。~A の Goal を証明するには、(~A は A -> False と同じことなので) intro を使って (背理法で) 証明する。

elimination の規則：

```
Coq < Check False_ind.
False_ind : forall P : Prop,
            False -> P
```

仮定に~A があって、それの名前が H だったとき、elim H とすると、Goal の P が A になる。

2.4 ある〜が存在して

「ある A 型の x が存在して Q」を

exists x : A, Q と書く。これは、

ex (fun x : A => Q) と書いても同じ。特に ex P は P が A 上の述語なら (つまり P が例えば fun x : A => Q のような、A -> Prop 型の関数だったら) exists x : A, P x のこと。

introduction の規則：

```
Coq < Check ex_intro.
ex_intro :
  forall (A : Type) (P : A -> Prop)
         (x : A),
  P x -> ex P
```

Goal が exists x : A, Q で、かつ仮定に t : A があるなら、exists t とすると、Goal が Q になる。ここで指定している t には、存在すると言っている A 型の値を具体的に指定する。

elimination の規則：

```
Coq < Check ex_ind.
ex_ind :
  forall (A : Type) (P : A -> Prop)
         (P0 : Prop),
  (forall x : A, P x -> P0) -> ex P
                            -> P0
```

仮定に ex P があって、それの名前が H だったとき、elim H とすると、Goal の P0 が forall x : A, P x -> P0 になる。

3 等式の証明

reflexivity.

Goal が A = A の形のときに使う tactic。

rewrite 仮定の名前.

例えば仮定に a = b があって、その名前が H のとき、rewrite H. とすると、Goal である P に含まれている a が全て b になる。逆に、P に含まれている b を全て a にしたいときには

rewrite <- H.

とする。

練習問題

問 1 以下を証明せよ。

```
(1) ~False
(2) forall P : Prop, ~~~P -> ~P
(3) forall P Q : Prop, ~~~P -> P -> Q
(4) forall P Q : Prop, (P -> Q) -> ~Q -> ~P
(5) forall A B C : Prop,
              A /\ (B /\ C) -> (A /\ B) /\ C
(6) forall A : Prop, ~(A /\ ~A)
(7) forall A B C : Prop,
              A \/ (B \/ C) -> (A \/ B) \/ C
(8) forall A : Prop, ~~(A \/ ~A)
(9) forall A B : Prop, (A \/ B) /\ ~A -> B
```

問 2

Variables (A : Set) (P Q : A -> Prop).
と宣言したセクションの中で以下を証明せよ。

```
(1) (exists x : A, P x \/ Q x)
             -> (ex P) \/ (ex Q)
(2) (ex P) \/ (ex Q)
            -> exists x : A, P x \/ Q x
(3) (exists x : A,
      (forall R : A -> Prop, R x)) -> 2 = 3
(4) (forall x : A, P x)
             -> ~(exists y : A, ~ P y)
```

問 3 以下を証明せよ

```
forall (A : Set) (x y z : A),
        x = y -> y = z -> x = z
```