Aula 14 - Lógica em Coq 2

Rafael Castro - rafaelcgs10.github.io/coq

06/06/2018

Equivalência lógica

- Uma equivalência lógica acontece quando duas proposições são verdadeiras para a mesma valoração.
- Conhecido como "se, e somente se,".
- É apenas uma conjunção de implicações

```
Module MyIff.

Definition iff (P Q : Prop) := (P -> Q) /\ (Q -> P).

Notation "P <-> Q" := (iff P Q)
      (at level 95, no associativity)
      : type_scope.

End MyIff.
```

Exemplo de equivalência

- Se P e Q são equivalentes, então Q e P são equivalentes.
- Como a bi-implicação é apenas uma conjunção, utiliza-se *split*.

```
Theorem iff_sym : forall P Q : Prop,
  (P <-> Q) -> (Q <-> P).
Proof.
  intros P Q [HAB HBA].
  split.
  - (* -> *) apply HBA.
  - (* <- *) apply HAB.  Qed.
```

Quantificação existencial no objetivo

- Para dizer que existe um x de tipo T, tal que alguma propriedade P é verdade para x, escreve-se: *exist* $x : T, P$.
- Utiliza-se o quantificador *exists* de maneira similar ao *forall*.
- A tática *exists* escolhe qual elemento existe.
- Exemplo de prova de existência:

```
Lemma four_is_even : exists n : nat, 4 = n + n.
Proof.
  exists 2.
  reflexivity.
Qed.
```

Quantificação existencial nas hipóteses

- Se há uma hipótese que afirma uma existencia *exists x, Px*, pode-se utilizar *destruct* para obter as hipóteses *x* e *P* sobre *x*.

```
Theorem exists_example_2 : forall n,
  (exists m, n = 4 + m) ->
  (exists o, n = 2 + o).
Proof.
  intros n Hm.
  destruct Hm as [m H].
  exists (2 + m).
  apply H.  Qed.
```

Definição do tipo existencial

- O tipo existencial é uma proposição definida indutivamente.
- O tipo existencial é parametrizado por um tipo *A* (implicíto) e uma função *P : A -> Prop* que é proposição quantificada.
- O *forall* é um parâmetro do construtor, não do tipo.
- A tática *exists* é apenas um *apply* com o construtor.

```
Section MyEx.
   Inductive ex {A : Type} (P : A -> Prop) : Prop :=
    ex_intro : forall x : A, P x -> ex P.

   Theorem exemple_ex : ex (fun n : nat => n = n).
     apply ex_intro with (x:=1).
     reflexivity.
    Qed.
```

Notação do tipo existencial

```
Notation "'existss' x .. y , p" := (ex (fun x => .. (ex (fun
  (at level 200, x binder, right associativity,
   format "'[' 'existss' '/ ' x .. y , '/ ' p ']'").

   Theorem exemple_ex2 : existss n : nat, n = n.
     apply ex_intro with (x:=1).
     reflexivity.
   Qed.
End MyEx.
```

Coq vs Teoria dos Conjuntos

- A fundação da matemática tradocional, com papel e cante, é a Teoria dos Conjuntos (ZFC).
- Um objeto matemática pode ser membro de vários conjuntos em ZFC, porém em Coq um termo é membro de no máximo um tipo.
- Isso, em geral, não é um problema: em ZFC o número dois é membro do conjuntos do naturais em do números pares; em Coq o número dois é membro do tipo *nat* e tem-se que *ev 2* é verdade, onde *ev : nat -> Prop* é uma propriedade que descreve números pares.
- Mas há situações onde é interessante adiconar axiomas.

Princípio da Extensionalidade

- Duas funções são iguais se tem o mesmo mapeamento (forall x, f x = g x) -> f = g.
- Esse princípio não faz parte do Coq. Se desejamos, precisamos adicionar.

```
Example function_equality_ex2 :
  (fun x => plus x 1) = (fun x => plus 1 x).
Proof.
   (* Stuck *)
Abort.
```

Princípio da Extensionalidade

- Utilizar o comando *Axiom* tem o mesmo efeito de enunciar um teorema e pular a prova com *Admitted*.
- Obviamente deve-se ser cuidadoso ao adicionar axiomas ao Coq, pois deixar o sistema inconsistente.

```
Axiom functional_extensionality : forall {X Y: Type}
    {f g : X -> Y},
  (forall (x:X), f x = g x) -> f = g.

Example function_equality_ex2 :
  (fun x => plus x 1) = (fun x => plus 1 x).
Proof.
  apply functional_extensionality. intros x.
  apply plus_comm.
Qed.
```

Lógica Clássica vs Lógica Construtivista

- A Lógica Clássica tem como tautologia P \/ ~P, conhecido como Axioma do Terceiro Excluído.
- E se não for possível demonstrar P e nem ~P? Ex: e se P for verdade, mas sem prova (proposição indecídivel).
- A Lógica Construtivista não toma como sempre verdade o terceiro excluído.
- Não há como provar o terceiro excluído em Coq!

```
Definition excluded_middle := forall P : Prop,
  P \/ ~ P.
```

Terceiro excluído pode ser verdade

- Há casos onde o terceiro excluído é verdade em Coq:

```
Theorem restricted_excluded_middle : forall P b,
  (P <-> b = true) -> P \/ ~ P.
Proof.
  intros P [] H.
  - left. apply H. reflexivity.
  - right. unfold not. intros HP. apply H in HP as contra.
    inversion contra.
Qed.
```