Aula 4 - Prova por simplificação, reescrita e análise de caso

Rafael Castro - rafaelcgs10.github.io/coq

07/05/2018

Prova por Simplificação

- Já vimos alguns exemplos de simplificação utilizando a *tactic simpl* e *reflexivity*.
- Os exemplos abaixo são um pouco diferente dos anteriores: há o quantificador para todo (*forall*).
- A *tactic intros* remove do objetivo o quantificador e o termo de um tipo quantificado, e insere no contexto um termo arbitrário do mesmo tipo com um dado identificador.

```
Theorem plus_O_n : forall n : nat, 0 + n = n.
Proof.
  intros n. simpl. reflexivity.  Qed.

Theorem plus_1_l : forall n:nat, 1 + n = S n.
Proof.
  intros n. reflexivity.  Qed.
```

Prova por Reescrita

- *intros* também move hipoteses de implicações para o contexto.
- *n* e *m* são números arbitrários. Não podemos simplismente usar simplificação e reflexividade.
- Uma das hipoteses afirma que n é o mesmo número que m.
- A tática *rewrite* com a seta -> substitui as ocorrencias do termo do lado esquerdo pelo do lado direito. Com a seta o contrário.

```
Theorem plus_id_example : forall n m:nat,
  n = m ->
  n + n = m + m.
Proof.
  intros n m.
  intros H.
  rewrite -> H.
```

Exercícios de sala sobre reescrita

Exercícios realizados em sala

```
Theorem mult_0_plus : forall n m : nat,
  (0 + n) * m = n * m.
Proof.
Admitted.

Theorem mult_S_1 : forall n m : nat,
  m = S n ->
  m * (1 + n) = m * m.
Proof.
Admitted.
```

Limites da Simplificação e Reescrita

- Simplificação e reescrita são muito poderosos, mas não são suficientes para o caso abaixo.
- A definição de ambas as funções faz casamento de padrão no primeiro argumento, porém o primeiro argumento de ambos é um número desconhecido.

```
Theorem plus_1_neq_0_firsttry : forall n : nat,
  beq_nat (n + 1) 0 = false.
Proof.
  intros n.
  simpl.  (* does nothing! *)
Abort.
```

Prova por Análise de Caso - Parte 1

- Para prosseguir na prova é necessário considerar os possíveis valores de n:
  1. Se $n = 0$ então pode-se simplificar para $beq\_nat\ 1\ 0 = false$
  2. Se $n = S\ n'$ então pode-se simplificar para $beq\_nat\ (S\ (n' + 1))\ 0 = false$.
- A tática *destruct* faz exatamente isso, ao analisar os possíveis valores com base na definição de *nat*.

Prova por Análise de Caso - Parte 2

- *destruct* cria dois sub-objetivos novos.
- *as* possibilitar definir os nomes que serão dados aos termos da análise de caso.
- O padrão é dado por uma lista de listas denotado por [... | ... | ...]. Na definição de *nat*:
  1. O primeiro termo somente pode ser *0*. Então usa-se uma lista vazia.
  2. O segundo termo é um sucessor que tem um argumento que é um número natural. Nomea-se esse argumento de *n'*. Se houvesse mais um argumento para esse termo ele seria dado separado por um espaço: *[|n' m']*.

Prova por Análise de Caso - Parte 3

- O sinal -, conhecido como *bullet*, serve para focar ocultar os demais objetivos e focar no atual. O uso é opcional, serve apenas para organizar a prova.

```
Theorem plus_1_neq_0 : forall n : nat,
  beq_nat (n + 1) 0 = false.
Proof.
  intros n. destruct n as [| n'].
  - reflexivity.
  - reflexivity.   Qed.
```

negb é Involutivo

- Pode-se utilizar a tática *destruct* em qualquer tipo definido indutivamente, por exemplo o tipo *bool*.
- Note que não há cláusula *as*, pois o tipo *bool* não tem argumentos. Poderia-se escrever *as []* ou *as [|]*.

```
Theorem negb_involutive : forall b : bool,
  negb (negb b) = b.
Proof.
  intros b. destruct b.
  - reflexivity.
  - reflexivity.  Qed.
```

andb é Comutativo

- De fato, é sempre possível omitir a cláusula *as*. Porém, se necessário o assistente escolherá os nomes para os termos.
- O sinal + funciona como -, porém para um sub-objetivo de um -.
- O sinal * é um terceiro *bullet*.

```
Theorem andb_commutative : forall b c, andb b c = andb c b.
Proof.
  intros b c. destruct b.
  - destruct c.
    + reflexivity.
    + reflexivity.
  - destruct c.
    + reflexivity.
    + reflexivity.
Qed.
```

andb Permuta

- Também é possível cercar com chaves para focar numa nova prova.
- Chaves são úteis quando há três niveis ou mais numa prova.

```
Theorem andb3_exchange :
  forall b c d, andb (andb b c) d = andb (andb b d) c.
Proof.
  intros b c d. destruct b.
  - destruct c.
    { destruct d. - reflexivity. - reflexivity. }
    { destruct d. - reflexivity. - reflexivity. }
  - destruct c.
    { destruct d. - reflexivity. - reflexivity. }
    { destruct d. - reflexivity. - reflexivity. }
Qed.
```