Lógica em Coq

Introdução ao assistente de provas Coq

Rodrigo Ribeiro

Lógica Proposicional em Coq

- Implicação

```
Variables A B C : Prop.

Theorem first_theorem : (A -> B) -> A -> B.
Proof....
```

Lógica Proposicional em Coq

- Estado inicial da prova

```
1 subgoal, subgoal 1 (ID 1)

A, B, C : Prop
============================
(A -> B) -> A -> B
```

Lógica Proposicional em Coq

- Após a execução da tática `intro Hab`.

```
1 subgoal, subgoal 1 (ID 2)

A, B, C : Prop
Hab : A -> B
============================
A -> B
```

Lógica Proposicional em Coq

- Após a execução da tática `intro Hab`

```
1 subgoal, subgoal 1 (ID 3)

A, B, C : Prop
Hab : A -> B
Ha : A
==============================
B
```

Lógica Proposicional em Coq

- Após a execução da tática `apply Hab`

```
1 subgoal, subgoal 1 (ID 3)

A, B, C : Prop
Hab : A -> B
Ha : A
============================
A
```

Lógica Proposicional em Coq

- Ao executarmos a tática `assumption`...

```
No more subgoals.
(dependent evars: (printing disabled) )
```

- Demonstração encerrada! Comando `Qed` encerra uma prova em Coq.

Lógica Proposicional em Coq

- Conjunção
  - Tática `split` divide a conclusão `A /\ B` em duas conclusões.
  - Tática `destruct H as [Ha Hb]` divide a hipótese `H : A /\ B` nas hipóteses `Ha : A` e `Hb : B`.

Lógica Proposicional em Coq

```
Lemma and_comm : A /\ B -> B /\ A.
Proof.
  intro Hab.
  destruct Hab as [Ha Hb].
  split.
  +
    assumption.
  +
    assumption.
Qed.
```

Lógica Proposicional em Coq

- Bicondicional
  - Tática `unfold` pode ser utilizada para substituir um nome por sua definição em uma hipótese ou conclusão.

```
Definition iff (A B : Prop) : Prop := (A -> B) /\ (B -> A).
```

Lógica Proposicional em Coq

```
Lemma and_comm_iff : (A /\ B) <-> (B /\ A).
Proof.
  unfold iff.
  split.
  +
    apply and_comm.
  +
    intro Hba.
    destruct Hba as [Hb Ha].
    split.
    *
      assumption.
    *
      assumption.
Qed.
```

Lógica Proposicional em Coq

- Negação
  - `False`: proposição para a qual não há demonstração.

```
Definition not (A  : Prop) : Prop := A -> False.
```

Lógica Proposicional em Coq

```
Lemma modus_tollens : ((A -> B) /\ ~ B) -> ~ A.
Proof.
    intro H.
    destruct H as [Hb Hnb].
    unfold not.
    unfold not in Hnb.
    intro Ha.
    apply Hnb.
    apply Hb.
    assumption.
Qed.
```

Lógica Proposicional em Coq

- Contradição
  - Tática `contradiction` resolve qualquer conclusão a partir das hipóteses `A` e `~ A`, para qualquer proposição `A`

```
Lemma contra : A -> ~ A -> B.
Proof.
  intro Ha.
  intro Hna.
  contradiction.
Qed.
```

Lógica Proposicional em Coq

- Disjunção
  - Tática `left` modifica a conclusão de `A \/ B` para `A`.
  - Tática `right` modifica a conclusão de `A \/ B` para `B`.
  - Se `H : A \/ B` é uma hipótese, a tática `destruct H as [Ha | Hb]`, divide o estado de prova atual em dois: um contendo a hipótese `Ha : A` e outro contendo a hipótese `Hb : B`.

Lógica Proposicional em Coq

```
Lemma or_comm : (A \/ B) -> (B \/ A).
Proof.
  intro Hab.
  destruct Hab as [Ha | Hb].
  +
    right.
    assumption.
  +
    left.
    assumption.
Qed.
```

Lógica de Predicados em Coq

- Definições de predicados, universo de discurso.

```
Hypothesis U : Set.
Hypothesis u : U.
Hypothesis P : U -> Prop.
Hypothesis Q : U -> Prop.
Hypothesis R : U -> Prop.
```

Lógica de Predicados em Coq

- Sobre universos: `Set` e `Prop`
  - Teoria de tipos como resposta às inconsistências descobertas por Russell.
  - Paradoxo de Russell: "Um certo barbeiro só faz a barba de quem não faz a própria barba. O barbeiro faz a própria barba?"
- Hierarquia de universos resolve essa inconsistência.

Lógica de Predicados em Coq

- Quantificador universal.

```
Lemma forall_and
: (forall x : U, P x /\ Q x) ->
 ((forall x : U, P x) /\ (forall x : U, Q x)).
  Proof.
    intro H.
    split.
    +
      intro y.
      destruct (H y).
      assumption.
    +
      intro y.
      destruct (H y).
      assumption.
  Qed.
```

Lógica de Predicados em Coq

```
Lemma forall_modus_ponens
    : ((forall x : U, P x -> Q x) /\
        (forall y : U, Q y -> R y)) ->
            (forall z : U, P z -> R z).
  Proof.
    intro Hpqr.
    destruct Hpqr as [Hpq Hqr].
    intro z.
    intro Hpz.
    apply Hqr.
    apply Hpq.
    assumption.
  Qed.
```

Lógica de Predicados em Coq

- Quantificador existencial

```
Lemma ex_or :
(exists x : U, P x \/ Q x) ->
(exists x : U, P x) \/ (exists y : U, Q y).
  Proof.
    intro Hpq.
    destruct Hpq as [x [Hpx | Hqx]].
    +
      left.
      exists x.
      assumption.
    +
      right.
      exists x.
      assumption.
  Qed.
```

Lógica em Coq

- Táticas são idênticas a regras da dedução natural.
- Demonstrações que não dependem do terceiro excluído podem ser demonstrados pelas táticas apresentadas.