Elementos Lógicos da Programação I
2003-2004

José Manuel Valença

10 de Outubro de 2003

Conteúdo

1 AULA 1: uso da Lógica de Primeira Ordem (FOL) **1**

2 Primeiro exemplo: a reflexividade da implicação **1**

3 Segundo exemplo: a transitividade da implicação **2**

4 Provas estruturadas **2**

5 A negação clássica **4**

1 AULA 1: uso da Lógica de Primeira Ordem (FOL)

theory *Aula1* = *FOL*:

2 Primeiro exemplo: a reflexividade da implicação

Note-se:

- A definição de um lema a provar atribuindo-lhe um nome para que este resultado possa ser usado, eventualmente, em provas futuras
- A prova recorre ao uso directo da regra de introdução da implicação *impI*
    1. O construtor **apply** aplicado a um método de prova - neste exemplo *(rule impI)* - produz um passo de prova, i.e. uma transformação de um estado de prova noutro estado de prova.
    2. **done** é o último passo que completa a prova

lemma *reflexividade* : $A \longrightarrow A$
  apply (*rule impI*)
  apply *assumption*
done

3 Segundo exemplo: a transitividade da implicação

A primeira prova usa várias vezes as duas regras da implicação: a introdução *impI* e a eliminação *mp*

No interpretador, verifique a definição dessas regras com o comando `thm` *nome da regra*

lemma *transitividade1* : $(A \longrightarrow B) \longrightarrow (B \longrightarrow C) \longrightarrow A \longrightarrow C$
apply (*rule impI*)
apply (*rule impI*)
apply (*rule impI*)
apply (*rule mp*)
apply *assumption*
apply (*rule mp*)
apply *assumption*
apply *assumption*
done

A segunda prova do mesmo lema usa a aplicação repetida da regra de introdução *impI* seguida da aplicação repetida da regra de eliminação *mp*.

Necessita de uma assunção final.

lemma *transitividade2* : $(A \longrightarrow B) \longrightarrow (B \longrightarrow C) \longrightarrow A \longrightarrow C$
apply (*intro impI*)
apply (*elim mp*)
apply *assumption*
done

Os construtores **proof** e **qed** aplicam as assumções finais que forem necessárias; **qed** só é aplicável se terminar a prova.

lemma *transitividade3* : $(A \longrightarrow B) \longrightarrow (B \longrightarrow C) \longrightarrow A \longrightarrow C$
proof (*intro impI*)
qed (*elim mp*)

by m_1 m_2 é uma abreviatura de **proof** m_1 **qed** m_2 por isso também só é aplicável se terminar a prova

lemma *transitividade4* : $(A \longrightarrow B) \longrightarrow (B \longrightarrow C) \longrightarrow A \longrightarrow C$
by (*intro impI*) (*elim mp*)

O método *auto* tenta automatizar a prova realizando-a num só passo

lemma *transitividade5* : $(A \longrightarrow B) \longrightarrow (B \longrightarrow C) \longrightarrow A \longrightarrow C$
by *auto*

4 Provas estruturadas

Uma prova "criptica"não estruturada usando as formas heuristicas de procura automática da forma de aplicação das regras

lemma *comutatividade1* : $A \wedge B \longrightarrow B \wedge A$
apply (*rules intro*: *impI conjI elim*: *conjE*)
done

A prova estruturada procura imitar a forma "natural" de apresentação de provas escondendo as transformações explícitas do estado e convertendo certos passos em *propagação de factos*

A primeira versão ainda refere explicitamente a regra de introdução usada:

- **proof** estabelece o início de uma cadeia de passos de prova terminada em **qed**
- **assume** introduz um teorema local que deve unificar com uma assumção
- **from** *this* **show** propaga o facto assumido anteriormente e completa-o com a definição de um novo estado de prova

lemma *comutatividade2* : $A \wedge B \longrightarrow B \wedge A$
proof
  assume $A \wedge B$
  from *this*
    show $B \wedge A$
  proof
    assume A B
    show $B \wedge A$ **by** (*rule conjI*)
  qed
qed

A mesma prova usando algumas abreviaturas

- **thus** é equivalente a **from** *this* **show**
- **by** *rule* automatiza a procura do último objectivo

lemma *comutatividade3* : $A \wedge B \longrightarrow B \wedge A$
proof
  assume $A \wedge B$
  thus $B \wedge A$
  proof
    assume A B
    show *?thesis* **by** *rule*
  qed
qed

O construtor **hence** introduz um estado de prova local (um lema) que pode depois ser usado na prova.

A ordem pelo qual os novos lemas são gerados é importante e por isso é conveniente associar nomes a esses lemas assim como às hipótese assumidas.

lemma *comutatividade4* : $A \wedge B \longrightarrow B \wedge A$
proof
  assume h: $A \wedge B$
  from h **have** b: B **by** *rule*
  from h **have** a: A **by** *rule*
  from b a **show** $B \wedge A$ **by** *rule*
qed

Porque *this* contém a última hipótese ou lema, usando as abreviaturas

- **with** *factos* ≡ **from** *factos this*

- .. ≡ **by** *rule*

a última prova pode ser escrita

lemma *comutatividade5* : $A \wedge B \longrightarrow B \wedge A$
proof
  assume *h*: $A \wedge B$
  from *h* **have** *b*: B **..**
  from *h* **have** A **..**
  with *b* **show** $B \wedge A$ **..**
qed

Pela definição da negação

lemma *disj-not*: $\neg P \Longrightarrow \neg Q \Longrightarrow \neg\ (P \vee Q)$
proof
 assume *a*: $P \vee Q$
 assume $\neg P$ **then have** *b*: $P \Longrightarrow$ *False* **..**
 assume $\neg Q$ **then have** *c*: $Q \Longrightarrow$ *False* **..**
 from *a* **and** *b* **and** *c* **show** *False* **..**
qed

5 A negação clássica

Provas envolvendo a negação clássica envolvem, normalmente, a lei de redução ao absurdo. Em FOL essa lei é representada em duas regras equivalentes:

- *classical*: $(\neg\ ?P \Longrightarrow ?P) \Longrightarrow ?P$
- *ccontr*: $(\neg\ ?P \Longrightarrow False) \Longrightarrow ?P$

Uma proposição, que caracteriza bem a diferença entre as versões clássica e intuicionista da Lógica Proposicional, é o chamado Lema de Pierce. Esta é uma proposição que é uma tautologia na versão clássica mas não o é na versão intuicionista.

Em seguida apresentam-se duas versões de prova clássica deste resultado (não estruturada e estruturada).

lemma *pierce1* : $((A \longrightarrow B) \longrightarrow A) \longrightarrow A$
  apply (*intro impI*)
  apply (*rule classical*)
  apply (*elim mp* [*of* $A \longrightarrow B$])
  apply (*intro impI*)
  apply (*elim notE*)
  apply *assumption*
done

lemma *pierce2* : $((A \longrightarrow B) \longrightarrow A) \longrightarrow A$
 proof
   assume *a*: $(A \longrightarrow B) \longrightarrow A$

```
    show A
    proof (rule classical)
      assume b: ¬ A
      have A ⟶ B
        proof
          assume A with b show B ..
        qed
      with a show A ..
    qed
qed
```

A dupla negação é também característica da LP clássica.

```
lemma dupla-negacao: ¬¬A ⟶ A
proof
  assume ¬¬A
  show A by (rule ccontr) (rule notE)
qed
```

Outra característica da LP clássica é a relação entre a disjunção e a implicação

$$\neg P \longrightarrow Q \quad \equiv \quad P \vee Q$$

A prova desse resultado passa por vários lemas intermédios.

Este resultado está directamente ligado ao lema mais característico da LP clássica e que é designado por *Tertium non Datur*

$$\neg P \vee P$$

```
lemma imp-disj : (¬P ⟶ Q) ⟶ P ∨ Q
proof
  assume a: ¬ P ⟶ Q
  show P ∨ Q
  proof (rule ccontr)
    assume b : ¬ (P ∨ Q)
    have c: ¬P
     proof
       assume P then have P∨Q ..
       with b show False ..
     qed
    have P
    proof (rule ccontr)
      assume ¬P with a have Q ..
      then have P ∨ Q ..
      with b show False ..
    qed
    with c show False ..
  qed
qed

lemma disj-imp: (P ∨ Q) ⟶ (¬P ⟶ Q)
proof
    assume P ∨ Q
```

```
    then show ¬P ⟶Q
    proof
      assume P show ¬P ⟶ Q
      proof assume ¬P then show Q .. qed
      next
      assume Q then show ¬P ⟶ Q ..
    qed
qed

theorem equiv-disj-imp: (P ∨ Q) ⟷ (¬P ⟶ Q)
proof
  assume P∨Q with disj-imp show ¬P ⟶ Q ..
  next
  assume ¬P ⟶ Q with imp-disj show P ∨ Q ..
qed

theorem tertium-non-datur: ¬P ∨ P
proof −
   have a: (¬¬P ⟶P) ⟶ (¬P∨P) by (rule imp-disj)
   have ¬¬P ⟶P by (rule dupla-negacao)
   with a show ?thesis ..
qed
```

Outro resultado clássico: a prova por casos

```
theorem casos: (¬P ⟹ Q) ⟹ (P ⟹ Q) ⟹ Q
proof −
  have a: ¬P ∨ P by (rule tertium-non-datur)
  assume b: ¬P ⟹ Q
  assume c: P ⟹ Q
  from  a and b and c show Q ..
qed

end
```