Elementos Lógicos da Programação I– Módulo I

Lógica Proposicional

1. Sintaxe.
2. Validade Clássica e Validade Intuicionista.
3. Teorias.
4. Consequência Semântica.
5. Relações de Dedução; correcção e completude.
6. Sistemas de Dedução.
7. Sistemas Hilbertianos.
8. Equivalência de Sistemas Hilbertianos.
9. Correcção e Completude.

José Manuel E. Valença @ DI Universidade do Minho Outubro, 1999

Sintaxe

Genericamente a **Lógica Posicional** associa a **frases** de uma linguagem o atributo de **validade**.

Exemplos de frases eventualmente válidas

(1) *O programa P termina*

(2) *A variável x alcança o valor* 0.

(3) *Se a variável x alcança o valor* 0 *então o programa P termina.*

(4) *O programa P termina e a variável x alcança o valor* 0

Frases que, num determinado contexto, são **logicamente atómicas** no sentido em que a sua validade é independente da validade de qualquer outra frase são representadas por **símbolos proposicionais**.

Todas as frases da lógica que não são representadas por esses símbolos são construídas por combinação de outras frases usando **conectivos proposicionais**.

As frases (1) e (2) podem ser representados por símbolos proposicionais; p.ex. **p** e **q**.

As frases (3) e (4) são construídas à custa de **p** e **q** usando, respectivamente, o conectivo **implicação** e o conectivo **conjunção**

$$\mathbf{q} \supset \mathbf{p} \qquad \text{e} \qquad \mathbf{p} \wedge \mathbf{q}$$

1 Definição *Dado um conjunto de símbolos proposicionais $\mathcal{P}$ a linguagem proposicional $\mathcal{L}_{\mathcal{P}}$ é o menor conjunto de frases que contém $\mathcal{P}$ e ainda*

$$P \wedge Q \quad , \quad P \vee Q \quad , \quad P \supset Q \quad , \quad \neg P \quad , \quad \bot$$

para todo $P, Q \in \mathcal{L}_{\mathcal{P}}$.

Validade

Existem duas visões alternativas do conceito de validade:

Validade Clássica: a cada proposição está associado um e só um de dois **valores de verdade** (denotados pelos símbolos verdadeiro e falso); a proposição é válida se lhe é associado o valor verdadeiro.

Validade Intuicionista: uma proposição é válida se e só se é possível construir uma **prova** dessa validade.

EXEMPLO

chove amanhã ou não chove amanhã

é representada por uma proposição $\mathbf{p} \vee (\neg \mathbf{p})$ que é sempre válida na abordagem clássica mas não é garantidamente válida na abordagem intuicionista.

De facto intuicionisticamente a frase só seria válida se fosse possível provar que amanhã chove ou então se fosse possível provar que amanhã não chove.

Validade Clássica

2 DEFINIÇÃO *Um* ***modelo*** *de* $\mathcal{L}_{\mathcal{P}}$ *é um subconjunto* $\mathcal{M} \subseteq \mathcal{P}$ *. Um símbolo proposicional* $p \in \mathcal{P}$ *é* ***válido no modelo*** $\mathcal{M}$ *, e escreve-se*

$$\mathcal{M} \models p$$

se e só se for um elemento de $\mathcal{M}$ *.*

EXEMPLO

Se $\mathcal{P} = \{p_0, p_1, \cdots, p_n, \cdots\}$ for um conjunto de símbolos proposicionais, um possível modelo dessa lógica será

$$\mathcal{M} = \{p_i \mid i \text{ é ímpar}\}$$

Neste caso verifica-se

$$\mathcal{M} \models p_3 \qquad \text{e} \qquad \mathcal{M} \not\models p_2$$

A noção de *fórmula válida num modelo* estende-se a proposições genéricas

3 DEFINIÇÃO *Dado um modelo $\mathcal{M}$, uma proposição P é* **válida** *no modelo, representada por $\mathcal{M} \models P$, quando*

(1) $\mathcal{M} \models p$ *sse* $p \in \mathcal{M}$ *quando p for atómico.*

(2) $\mathcal{M} \models \neg P$ *sse* $\mathcal{M} \not\models P$.

(3) $\mathcal{M} \models P \wedge Q$ *sse* $\mathcal{M} \models P$ ***e*** $\mathcal{M} \models Q$.

(4) $\mathcal{M} \models P \vee Q$ *sse* $\mathcal{M} \models P$ ***ou*** $\mathcal{M} \models Q$.

(5) $\mathcal{M} \models P \supset Q$ *sse* $\mathcal{M} \not\models P$ ***ou*** $\mathcal{M} \models Q$.

(6) $\mathcal{M} \not\models \bot$

Esta definição traduz-se directamente no seguinte código PROLOG

```
:- use_module(library(lists),[member/2]).
:- op(400,fy,-), op(500,xfy,&), op(600,xfy,v), op(650,xfy,=>).

val(M, P)       :- atom(P) , member(P,M).
val(M, -(P))    :- \+ val(M,P).
val(M, P & Q)   :- val(M,P) , val(M,Q).
val(M, P v Q)   :- val(M,P) ; val(M,Q).
val(M, P => Q)  :- \+ val(M,P) ; val(M,Q).
val(M, abs)     :- fail.
```

O exemplo seguinte usa este código para determinar a validade da proposição $p \supset (q \wedge p)$ nos modelos $\{p, q\}$ e $\{p\}$.

```
| ?- val([p,q] , p => (q & p)).

yes
| ?- val([p] , p => (q & p)).

no
```

Validade Intuicionista

Em Lógica Proposicional Intuicionista uma **prova** para uma proposição P é uma qualquer estrutura matemática que verifique as seguintes condições (conhecidas por *condições de Kolmogorov-Heyting*)

(KH-1) Não existe qualquer prova associada à constante $\bot$.

(KH-2) Uma prova de $P \wedge Q$ obtém-se apresentando uma prova de P e uma prova de Q.

(KH-3) Uma prova de $P \vee Q$ obtém-se apresentando uma prova de P ou uma prova de Q e identificando a alternativa escolhida.

(KH-4) Uma prova de $P \supset Q$ é uma transformação que, a cada eventual prova de P, associa uma prova de Q.

(KH-5) $\neg P$ é uma abreviatura de $P \supset \bot$.

Um **modelo intuicionista** da Lógica Proposicional $\mathcal{M}_i$ selecciona as proposições válidas associando, a cada uma, pelo menos uma prova.

1 Proposição *$\mathcal{M}_i$ determina um modelo da $\mathcal{M}$ tal que toda proposição P válida intuicionisticamente em $\mathcal{M}_i$ verifica $\mathcal{M} \models P$.*

Prova Definimos $\mathcal{M}$ como o conjunto de todos os símbolos proposicionais válidos em $\mathcal{M}_i$. Podemos agora percorrer, por indução, os casos previstos na definição 3 e verificar $\mathcal{M} \models P$ em cada caso.

Tomando, por exemplo, o caso (5) temos de considerar duas possibilidades:

(i) Não existe prova para P; então P não é válido, verifica-se $\mathcal{M} \not\models P$ e, pela definição (5), isto garante $\mathcal{M} \models P \supset Q$,

(ii) Existe prova para P; então, usando a prova intuicionista de $P \supset Q$ constrói-se uma prova para Q. Isto garante $\mathcal{M} \models Q$ que, pela definição (5), é suficiente para assegurar $\mathcal{M} \models P \supset Q$.

Notas

A proposição 1 estabelece, essencialmente, que toda a proposição válida intuicioistcamente é válida clássicamente. O inverso não é verdade como se pode ver do exemplo

chove amanhã ou não *chove amanhã*

A Lógica Proposicional Clássica está mais perto da noção usual de validade. A Lógica Intuicionista representa melhor as situações onde os aspectos computacionais são importantes. Ambas devem ser estudadas neste curso.

Teorias

Uma **teoria** Γ é um conjunto de proposições. A teoria é válida no modelo $\mathcal{M}$, e escreve-se

$$\mathcal{M} \models \Gamma$$

quando todas as proposições em Γ são $\mathcal{M}$-válidas.

O seguinte código PROLOG usa a anterior definição de `val(M, P)` para verificar a validade de uma teoria.

```
valt(M, []).
valt(M, [P|G]) :- val(M,P) , valt(M,G).
```

Por exemplo, a validade da teoria $\{p \wedge q\, ,\, p \supset q\}$ nos modelos $\{p, q\}$ e $\{q\}$, pode ser verificada com

```
| ?- valt([p,q] , [p & q , p => q]).

yes

| ?- valt([q] , [ p & q , p => q]).

no
```

Consequência Semântica

A noção de **consequência** lida com validade que seja independente de modelos.

4 DEFINIÇÃO *Se, para todo o modelo $\mathcal{M}$, sempre que se verificar $\mathcal{M} \models \Gamma$ verifica-se $\mathcal{M} \models P$ diz-se que P é uma* ***consequência semântica*** *da teoria Γ e escreve-se*

$$\Gamma \models P$$

Uma ***tautologia*** *é qualquer consequência da teoria vazia – $\{\} \models P$ ou simplesmente $\models P$.*

Quando o conjunto de símbolos proposicionais é finito é possível implementar directamente este definição em PROLOG usando dois procedimentos auxiliares: `modelos` calcula todos os modelos possíveis formados com símbolos `U`, enquanto que `entails` implementa própriamente a definição de consequência dado um conjunto de modelos `S`.

```
conseq(U,G,P) :- modelos(U,S) , entails(S,G,P).
entails([],G,P).
entails([M|S], G, P):-
       (valt(M,G) -> val(M,P) ; true) , entails(S,G,P).
modelos(U,S) :- setof(M, sublist(M,U), S).
```

Usamos este programa para verificar que $((P \supset Q) \supset P) \supset P$ é uma tautologia

```
| ?- conseq([p,q], [], ((p => q) => p) => p).

yes
```

Relações de Dedução

Motivação

As noções semânticas (como *tautologia* ou *consequência*) não são fáceis de utilizar quando se pretende verificar a validade de uma frase. O número de modelos que é necessário analisar pode ser infinito ou, mesmo que seja finito, pode ser de tal modo elevado que torna a abordagem directa impraticável.

Alternativamente pode-se tentar identificar relações genéricas entre frases que permitam determinar aquelas que são válidas. Se for, depois, possível encontrar um processo algorítmico de calcular estas relações temos uma forma alternativa de determinar a validade dessas frases.

Surge assim a noção de *relação de dedução*

5 Definição *Uma* ***relação de dedução*** $\Gamma \vdash P$ *é uma relação binária entre teorias* Γ *e frases* P*, que verifica as seguintes condições,*

(i) Se $P \in \Gamma$ *então* $\Gamma \vdash P$ *(inclusão)*

(ii) Se $\Delta \vdash P$ *e* $\Gamma \supseteq \Delta$ *então* $\Gamma \vdash P$ *(monotonia)*

(iii) Se $\Gamma \cup \{H\} \vdash P$ *e* $\Delta \vdash H$ *, então* $\Gamma \cup \Delta \vdash P$ *(corte)*

Numa relação $\Gamma \vdash P$ *a proposição* P *chama-se* ***conclusão*** *e as proposições* $H \in \Gamma$ *chamam-se* ***hipóteses****.*
Se se verificar $\{\} \vdash P$ *(também escrito* $\vdash P$ *) então* P *é um* ***teorema*** *para a relação de dedução* $\vdash$ *.*

2 Proposição *A relação de consequência semântica* $\models$ *é uma relação de dedução.*

Prova É trivial verificar a satisfação de cada uma das três condições da definição 5 a partir da noção de consequência apresentada na definição 4.

Comentários

(i) diz-nos que qualquer hipótese $P \in \Gamma$ é sempre uma conclusão de Γ.

(ii) diz-nos que, se de um conjunto de hipóteses Δ é possível deduzir a conclusão P, então aumentando as hipóteses (escolhendo $\Gamma \supseteq \Delta$) consegue-se deduzir pelo menos a mesma conclusão P.

(iii) justifica a utilização de lemas nas deduções; em $\Gamma \cup \{H\} \vdash P$ assume-se que para concluir P são necessárias as hipóteses em Γ e ainda uma hipótese extra H; assume-se também que H é conclusão de um outro conjunto de hipóteses ($\Delta \vdash H$ – o lema); a condição do corte diz-nos que é possível *cortar* a hipótese H substituindo-a pelas hipóteses Δ do lema.

A relação de consequência $\models$ satisfaz as condições de uma relação de dedução genérica. No entanto esta é uma relação que, como dissemos, é de difícil utilização.

Assim o que se pretende é uma outra relação de dedução $\vdash$ que seja facilmente computável e que seja, de alguma forma, equivalente à relação de consequência. Pretende-se também definir termos de comparação entre a relação de consequência $\models$ e uma outra relação de dedução $\vdash$ qualquer.

6 Definição *Uma relação de dedução* $\vdash$ *diz-se* ***correcta*** *quando, sempre que se verifica* $\Gamma \vdash P$ *, verifica-se* $\Gamma \models P$.
A relação diz-se ***completa*** *quando, não se verificando* $\Gamma \vdash P$ *, podemos garantir que também não se verifica* $\Gamma \models P$.

Comentários

Quando uma relação de dedução é correcta isto significa que os deduções $\Gamma \vdash P$ que for possível construir correspondem realmente a consequências semânticas $\Gamma \models P$ legítimas: todos os teoremas são tautologias.

Se a relação for completa isto significa que não existe nenhuma consequência semântica que não possa ser deduzida na relação $\vdash$: não existem tautologias que não sejam também teoremas.

Exemplo

O seguinte programa PROLOG implementa uma relação de dedução (descrita no predicado `prove(G,P)`) que é correcta mas que não é completa.

```
prove(G, P)             :- atom(P) , member(P,G).
prove(G , P & Q)        :- prove(G , P) , prove(G , Q).
prove([A & B | G] , P)  :- prove([A,B|G] , P).
prove(G , P => Q)       :- prove([P|G] ,  Q).
prove([A => B| G] , P)  :- prove(G , A) , prove([B|G] , P).
prove([H|G] , P)        :- prove(G , P).
```

Com esta relação de dedução pode-se, por exemplo, deduzir $\{p \supset q, p\} \vdash q$

```
| ?- prove([p => q, p], q).

yes
```

A relação é correcta porque todas as deduções que constrói correspondem a consequências válias.

A relação de dedução está longe de ser completa porque está limitada a proposições que não têm outros conectivos além de $\supset$ e $\wedge$ e adicionalmente, mesmo nesta lógica restrita, não consegue deduzir todas as consequências válidas.

Por exemplo, a consequência $\{p, p \supset q\} \models q$ é válida mas o programa responde

```
| ?- prove([p , p => q], q).

no
```

O problema está na última cláusula que "perde" a hipótese `H` depois de a tentar usar nas cláusulas anteriores.

Pode-se modificar este programa acrescentando uma variável `Acc` de forma a acumular as hipóteses testadas; estas hipóteses não são "deitadas fora" mas ficam ainda disponíveis para futuros testes.

```
:-      use_module(library(lists),[member/2]).
:-      op(500,xfy,&), op(650,xfy,=>).

prove(G , P)                    :- prove(G , [] , P).
prove(G, Acc , P)               :-
      atom(P) , (member(P,G) ;  member(P,Acc)).
prove(G , Acc,  P & Q)          :-
      prove(G , Acc , P) , prove(G , Acc , Q).
prove([A & B | G] , Acc,  P)  :- prove([A,B|G] , Acc , P).
prove(G , Acc , P => Q)         :- prove([P|G] , Acc,  Q).
prove([A => B| G] , Acc , P)  :-
      prove(G , Acc , A) , prove([B|G] , Acc , P).
prove([H|G] , Acc , P)          :- prove(G , [H|Acc] , P).
```

Note-se que o predicado `prove` ocorre com 2 argumentos na primeira cláusula (onde não se usa nenhuma acumulação de hipóteses) e ocorre com 3 argumentos nas restantes cláusulas (introduzindo a acumulação de hipóteses usadas).

Agora a dedução $\{p, p \supset q\} \vdash q$ já dá o resultado esperado.

```
| ?- prove([p,p=>q],q).

yes
```

Note-se que esta técnica de acumulação de hipóteses é meramente algorítmica; não se reflecte na definição de dedução mas apenas na forma como ela é implementada.

Sistemas de Dedução

Como qualquer linguagem proposicional (não trivial) tem um número infinito de proposições P e um número infinito de teorias Γ não é possível construir um algoritmo que compare todas as possibilidades de consequência $\Gamma \models P$ com todas as possibilidades de dedução $\Gamma \vdash P$ e conclua se a relação de dedução é correcta ou completa.

A prova de que uma relação de dedução é eventualmente correcta e/ou completa tem de outros métodos que não a comparação exaustiva das duas relações.

Um **sistema de dedução** é uma linguagem (cujas frases se designam por **deduções**) que permite definir uma relação de dedução de tal modo que é possível verificar se tais relações são correctas e/ou completas.

Cada frase desta linguagem (dedução) identifica sintácticamente uma proposição particular como sendo a *conclusão* da dedução e um conjunto de outras proposições como sendo *hipóteses* da dedução.

Cada dedução bem-formada identifica de forma meramente sintáctica, um elemento $\Gamma \vdash P$ de uma relação entre hipóteses e conclusões. O sistema de dedução é **bem-formado** se a relação assim definida satisfizer as condições da definição 5.

Exemplo

Um possível sistema de dedução é o programa PROLOG que implementa o predicado `prove(G,P)`.

Como argumentos deste predicado temos o conjunto de hipóteses `G` e a conclusão `P`. Cada execução deste programa origina uma dedução que se escreve registando todas as invocações intermédias de `prove` com os respectivos argumentos (o "*trace*" da execução do programa).

A invocação da dedução $\{p, p \supset q\} \vdash q$, com o programa PROLOG descrito no anteriormente em modo *trace*, origina um texto do seguinte tipo.

```
| ?- prove([p, p => q], q).
      Call: prove([p,p=>q],q)
      Call:   prove([p,p=>q],[],q)
      Call:     prove([p=>q],[p],q)
      Call:       prove([],[p],p)
      Exit:       prove([],[p],p)
      Call:       prove([q],[p],q)
      Exit:       prove([q],[p],q)
      Exit:     prove([p=>q],[p],q)
      Exit:   prove([p,p=>q],[],q)
      Exit: prove([p,p=>q],q)
yes
{trace}
```

As linhas `Call` indicam a invocação do predicado enquanto as linhas `Exit` indicam o fim de uma invocação com sucesso.

Este texto identifica claramente as hipóteses e a conclusão (na invocação inicial) e representa uma descrição de como é construída a prova da consequência pretendida. Como tal o texto pode ser visto como uma *dedução* no *sistema de dedução* que o programa PROLOG representa.

Sistemas de Dedução Hilbertianos

Neste curso estudaremos em detalhe alguns sistemas de dedução com interesse computacional (como a **dedução natural**, o **cálculo de sequentes** , o **sistema de «tableaux»**) mas, no entanto, para ilustrar os conceitos genéricos é preferível começar-mos por um outro sistema conceptualmente mais simples.

7 DEFINIÇÃO *Um sistema de dedução hilbertiano é determinado por um conjunto de fórmulas* $\mathcal{A}$ *(chamadas* ***axiomas****) e por um conjunto de* ***regras de inferência*** *com a estrutura seguinte*

$$\frac{P_1 \quad P_2 \quad \cdots \quad P_n}{C}$$

C *é uma fórmula chamada* ***conclusão da rega*** *e cada uma das entidades* P_i *é uma também uma fórmula designada por* ***premissa da regra****.*

Uma ***dedução hilbertiana*** $\Gamma \vdash P$ *é uma sequência de fórmulas*

$$\mathcal{D} \equiv \langle Q_1 , \cdots , Q_i , \cdots , Q_j , \cdots , Q_m \rangle$$

satisfazendo as seguintes condições:

1. *A primeira fórmula,* Q_1 *, coincide com* P*.*
2. *Cada uma das fórmulas* Q_i *é gerada de um dos seguintes modos:*
   - *(i)* Q_i *coincide com um dos axiomas* $A \in \mathcal{A}$ *,* ou
   - *(ii)* Q_i *coincide com uma das hipóteses de dedução* $H \in \Gamma$ *,* ou
   - *(iii)* Q_i *coincide com a conclusão* C *de uma regra de inferência em que* <u>todas</u> *as premissas são fórmulas* Q_j *que ocorrem em* $\mathcal{D}$ <u>depois</u> *de* Q_i *(i.e.,* $j > i$*).*

Exemplo

É possível apresentar muitos sistemas hilbertianos para a Lógica Proposicional. Um dos mais vulgares é definido para apenas duas conectivas ($\neg$ e $\supset$) e é constituído por três axiomas e uma única regra de inferência.

(A_1) $P \supset Q \supset P$

(A_2) $(P \supset Q \supset R) \supset (P \supset Q) \supset P \supset R$

(A_3) $((\neg P \supset Q) \supset (\neg P \supset \neg Q) \supset P$

(MP) $\dfrac{P \quad P \supset Q}{Q}$ *Modus Ponens*

É importante referir que (A_1), (A_2) e (A_3) são realmente *esquemas de axiomas* denotando um número infinito de axiomas diferentes: para cada substituição das variáveis P, Q e R por fórmulas concretas, obtemos um novo axioma.

O mesmo se passa com (MP): cada substituição de P e Q por fórmulas concretas dá origem a uma regra diferente.

O seguinte quadro representa uma dedução neste sistema para $\vdash P \supset P$. Cada linha do quadro contém uma fórmula da sequência que constitui a dedução. À esquerda indicamos o número de ordem da fórmula e à direita indicamos a justificação para a sua inclusão na sequência.

refl	$P \supset P$	
1.	$P \supset P$	MP 2,3
2.	$P \supset (P \supset P)$	A_1
3.	$(P \supset (P \supset P)) \supset (P \supset P)$	MP 4,5
4.	$P \supset (P \supset P) \supset P$	A_1
5.	$(P \supset (P \supset P) \supset P) \supset (P \supset (P \supset P)) \supset (P \supset P)$	A_2

A dedução $\{P \supset Q\,,\, Q \supset R\} \vdash P \supset R$ ilustra a construção de uma dedução hilbertiana envolvendo hipóteses.

trans	$P \supset R$	$[\, P \supset Q\,,\, Q \supset R\,]$
1.	$P \supset R$	MP 2,3
2.	$(P \supset Q) \supset (P \supset R)$	MP 4,5
3.	$P \supset Q$	hipótese
4.	$(P \supset Q \supset R) \supset (P \supset Q) \supset (P \supset R)$	A_2
5.	$(P \supset Q \supset R)$	MP 6,7
6.	$(Q \supset R) \supset (P \supset (Q \supset R))$	A_1
7.	$Q \supset R$	hipótese

Os quadros com que apresentamos estas deduções têm um cabeçalho com um identificador (para que esta dedução possa ser referida em deduções futuras), a conclusão e as hipóteses da dedução.

Como se facilmente demonstra toda relação de dedução hilbertiana verifica as três condições (*inclusão*, *monotonia* e *corte*) que caracterizam uma relação de dedução genérica.

Nomeadamente a condição de corte pode ser usada para disponibilizar novas regras de inferência: se previamente se tiver construída uma dedução $\{H_1, \cdots, H_n\} \vdash H$, a condição de corte garante que é possível usar

$$\frac{H_1 \quad \cdots \quad H_n}{H}$$

como regra de inferência em deduções futuras.

É o que se ilustra no exemplo seguinte que invoca as duas deduções anteriores identificadas pelos nomes **refl** e **trans**.

dneg	$\neg\neg P \supset P$

1.	$\neg\neg P \supset P$	**trans** 2,3
2.	$\neg\neg P \supset (\neg P \supset \neg\neg P)$	A_1
3.	$(\neg P \supset \neg\neg P) \supset P$	MP 4,5
4.	$(\neg P \supset \neg P) \supset (\neg P \supset \neg\neg P) \supset P$	A_3
5.	$(\neg P \supset \neg P)$	**refl**

O seguinte programa Prolog implementa directamente a definição de dedução de Hilbert. O predicado `axioma` instancia os vários axiomas e o predicado `regra` instancia nas várias regras do sistema hilbertiano usado.

```
deduz(_,P,[P|_])  :- axioma(P).
deduz(G,P,[P|_])  :- member(P,G).
deduz(G,P,[P|D])  :- regra(Prem,P), deduz_prem(G,Prem,D).
deduz(G,P,[_|D])  :- deduz(G,P,D).
deduz_prem(G,[],D).
deduz_prem(G,[Q|Qs],D) :- deduz(G,Q,D), deduz_prem(G,Qs,D).
```

O sistema de dedução hilbertiano usado nos exemplos anteriores pode ser implementado da forma seguinte.

```
axioma(P  => Q => P).
axioma((P => Q => R) => (P => Q) => P => R).
axioma((-(P) => Q) => (-(P) => -(Q)) => P).
regra([P , P => Q] , Q).
```

Uma construção da dedução de $\vdash P \supset P$ é feita por

```
| ?- deduz([], p => p, D).
D = [p=>p,p=>_A=>p,(p=>_A=>p)=>p=>p,p=>(_A=>p)=>p,
                 (p=>(_A=>p)=>p)=>(p=>_A=>p)=>p=>p|_B] ?
yes
```

Comentários

Não há garantia de que um tal programa consiga construir uma dedução `D` para todo o par de argumentos `G` e `P`; essencialmente isto acontece porque cada invocação de `deduz` pode gerar argumentos `P` cada vez mais complexos.

A construção de deduções hilbertianas é sempre um processo assente numa abordagem de "tentativa e experimentação". O uso do corte e de lemas previamente deduzidos como regras de inferência permite simplificar um pouco esse processo.

O seguinte resultado permite simplificar o processo de construção de deduções hilbertianas mas também dá uma interpretação própria à implicação $\supset$ ligando-a ao conceito de dedução.

1 Teorema (da dedução) *A relação de dedução determinada pelos axiomas* (A_1), (A_2) *e* (A_3) *e a regra (MP) verifica a seguinte condição*

$$\Gamma \vdash P \supset Q \quad \textit{se e só se} \quad \Gamma \cup \{P\} \vdash Q$$

(*A demonstração detes resultado aparece em qualquer texto de Lógica Proposiconal e é feita por indução no comprimento da dedução.*)

Exemplo

Para construir uma dedução hilbertiana para a tautologia

$$\neg P \supset (P \supset Q)$$

pode-se usar o teorema da dedução e construir, ao invés, uma dedução para

$$\{\neg P\,,\, P\} \vdash Q$$

O quadro seguinte ilustra essa prova.

Q	$[\neg P, P]$
1. Q	MP 2,4
2. $(\neg Q \supset \neg P) \supset Q$	MP 3,5
3. $(\neg Q \supset P) \supset (\neg Q \supset \neg P) \supset Q$	A_3
4. $(\neg Q \supset \neg P)$	MP 6,9
5. $(\neg Q \supset P)$	MP 7,8
6. $\neg P \supset (\neg Q \supset \neg P)$	A_1
7. $P \supset (\neg Q \supset P)$	A_1
8. P	hipótese
9. $\neg P$	hipótese

A dedução directa para $\neg P \supset P \supset Q$ seria muito mais complexa.

O sistema de dedução de Hilbert aqui apresentado (descrito nos axiomas (A_1), (A_2) e (A_3) e na regra *Modus Ponens*) contempla apenas uma lógica descrita por dois conectivos primitivos: a negação $\neg$ e a implicação $\supset$. Os restantes conectivos podem ser expressos em termos destes dois.

$$P \vee Q \equiv (\neg P \supset Q) \qquad P \wedge Q \equiv \neg(P \supset \neg Q)$$

O seguinte programa PROLOG implementa esta conversão

```
conv(P,P)                 :- atom(P).
conv(-(P),-(P1))          :- conv(P,P1).
conv(P => Q, P1 => Q1)    :- conv(P1,Q1).
conv(P & Q, -(P1 => -(Q1))) :- conv(P,P1) , conv(Q,Q1).
conv(P v Q, -(P1) => Q1)    :- conv(P,P1) , conv(Q,Q1).
```

Exemplo de conversão de $P \wedge (P \vee Q)$.

```
| ?- conv( p & (p v q) , U).
U = -(p=> -(-(p)=>q)) ?
yes
```

É possível definir outros sistemas de dedução hilbertianos usando outros conectivos primitivos. Por exemplo, usando $\neg$ e $\vee$ como conectivos primitivos e $A \supset B$ como abreviatura de $\neg A \vee B$, pode-se definir o seguinte sistema de dedução

(B_1) $P \vee P \supset P$ $\quad$ (B_2) $P \supset P \vee Q$

(B_3) $P \vee Q \supset Q \vee P$ $\quad$ (B_4) $(P \supset Q) \supset (R \vee P \supset R \vee Q)$

(MP) $\dfrac{P \quad P \supset Q}{Q}$

Uma dedução para a tautologia $\vdash \neg P \vee P$ (equivalente a $P \supset P$) neste sistema

refl	$\neg P \vee P$	
1.	$\neg P \vee P$	MP 2,4
2.	$(\neg P \vee (P \vee P)) \supset (\neg P \vee P)$	MP 3,5
3.	$((P \vee P) \supset P) \supset ((\neg P \vee (P \vee P)) \supset (\neg P \vee P))$	(B_4)
4.	$P \supset (P \vee P) \quad \equiv \quad \neg P \vee (P \vee P)$	(B_2)
5.	$(P \vee P) \supset P$	(B_1)

Existe uma equivalência entre o sistema de dedução definido pelos axiomas $(A_1) \cdots (A_3)$ e o sistema de dedução determinado pelos axiomas $(B_1) \cdots (B_4)$ expressa no seguinte resultado.

3 Proposição *Os dois sistemas de dedução hilbertianos determinam exactamente a mesma relação de dedução.*

Para provar este resultado, atendendo que ambos os sistemas têm a mesma regra de inferência, basta mostrar, em cada sistema, que cada um dos seus axiomas é um teorema no outro sistema. Por exemplo a dedução seguinte mostra que (A_1) é um teorema no sistema $\{(\mathrm{B}_1), \ldots, (\mathrm{B}_4), (\mathrm{MP})\}$

A_2	$P \supset Q \supset P \quad \equiv \neg P \vee (\neg Q \vee P)$

1.	$\neg P \vee (\neg Q \vee P)$	MP 2,5
2.	$(\neg P \vee (P \vee \neg Q)) \supset (\neg P \vee (\neg Q \vee P))$	MP 3,4
3.	$((P \vee \neg Q) \supset (\neg Q \vee P)) \supset$ $((P \vee P \vee \neg Q) \supset (P \vee (\neg Q \vee P)))$	(B_4)
4.	$(P \vee \neg Q) \supset (\neg Q \vee P)$	(B_3)
5.	$P \supset P \vee \neg Q \quad \equiv \quad \neg P \vee (P \vee \neg Q)$	(B_2)

Vários outros conjuntos de axiomas são possíveis que dão origem precisamente à mesma relação de dedução. Por isso faz sentido representar todos estes sistemas de dedução equivalentes por uma única relação $\vdash_H$ (a **relação de dedução de Hilbert**).

2 TEOREMA *A relação de dedução de Hilbert, na Lógica Proposicional Clássica, é correcta e completa.*

Tomando o primeiro sistema de dedução ($\{(A_1), (A_2), (A_3), (MP)\}$) como descritivo da relação de dedução $\vdash_H$, a prova da correcção pode ser feita mostrando que cada um dos axiomas é uma tautologia e que $\{P, P \supset Q\} \models P$ é uma consequência correcta.

Usando o programa PROLOG `conseq` podemos facilmente verificar esses factos.

```
| ?- conseq([p,q], [], p => q => p).
yes
| ?- conseq([p,q,r],[], (p => q => r) => (p => q) => p => r).
yes
| ?- conseq([p,q], [], (-(p) => q) => (-(p) => -(q)) => p).
yes
| ?- conseq([p,q], [p,p => q], q).
yes
```

A prova da completude usa dois resultados intermédios.

4 PROPOSIÇÃO *São deduzíveis no sistema de Hilbert as seguintes relações*

(a) $\{P, \neg Q\} \vdash \neg(P \supset Q)$ *(b)* $\{P\} \vdash \neg\neg P$

(c) $\{\neg P\} \vdash P \supset Q$ *(d)* $\{Q\} \vdash P \supset Q$

(e) $\{\neg P \supset Q\, ,\, P \supset Q\} \vdash Q$

A construção destas deduções é simples e é colocada como exercício.

O segundo resultado intermédio usa a *teoria de um modelo* $\mathcal{M}$, representada por $\overline{\mathcal{M}}$, que é formada pelas fórmulas p, se $p \in \mathcal{M}$, e $\neg p$ se $p \notin \mathcal{M}$.

Exemplo: em $\mathcal{U} = \{p, q, r\}$, se $\mathcal{M} = \{p\}$ então $\overline{\mathcal{M}} = \{p, \neg q, \neg r\}$.

A teoria de $\mathcal{M}$ é calculada pelo seguinte programa [a].

```
teoria(U,M,T) :-
    (setof(-(P), (member(P,U) , \+ member(P,M)), T1) ; T1= []),
     append(M,T1,T).

| ?- teoria([p,q,r],[p],T).
T = [p,-(q),-(r)] ?
yes
```

5 PROPOSIÇÃO *Se* $\mathcal{M} \models A$ *então verifica-se a dedução* $\overline{\mathcal{M}} \vdash_H A$.

A prova deste resultado faz-se definindo um predicado `completude` que constrói a dedução usando como regras de inferência as deduções na proposição 4.

```
completude(T,P,[P])  :-
    member(P,T), (atom(P) ; (P = -(Q) , atom(Q))).
completude(T,P => Q, [P => Q | D]) :-
    completude(T,-(P),D) ; completude(T,Q,D).
completude(T, -(-(P)), [-(-(P)) | D]) :-
    completude(T,P,D).
completude(T,-(P => Q), [-(P => Q) | D]) :-
    completude(T,P,Dp) , completude(T,-(Q), Dq), append(Dp,Dq,D).
```

[a] **append** é um predicado pre-definido que faz a junção de duas listas.

Notas

A primeira cláusula analisa o caso em que A é um símbolo proposicional ou a negação de um símbolo proposicional ($A \equiv \mathbf{p}$ ou $A \equiv \neg\mathbf{p}$).

A segunda cláusula considera $A \equiv P \supset Q$. Neste A é válido de for válido $\neg P$ **ou** de for válido Q.

Na primeira hipótese, a regra (c) e a dedução de conclusão $\neg P$ geram a dedução de conclusão $P \supset Q$. No segundo caso é a regra (d) e a dedução de conclusão Q que originam a dedução pretendida.

As últimas duas cláusulas analisam os casos em que A é da forma $\neg A'$, sendo A' não proposicional. Existem duas possibilidades: $A' \equiv \neg P$ ou $A' \equiv P \supset Q$.

Quando $A \equiv \neg A' \equiv \neg\neg P$ a regra (b) da proposição 4 permite construir uma dedução de conclusão $\neg\neg P$ a partir de uma dedução de conclusão P.

Quando $A \equiv \neg A' \equiv \neg(P \supset Q)$ a regra (a) permite construir uma dedução de conclusão $\neg(P \supset Q)$ juntando uma dedução de conclusão P com uma dedução de conclusão $\neg Q$. [a]

O predicado `comp` é uma interface que lê o universo de símbolos `U`, o modelo `M` e a fórmula `A`, verifica se esta é uma consequência do modelo, calcula a teoria e a dedução e imprime estes resultados.

```
comp(U,M,A) :-  conseq(U,M,A), teoria(U,M,T), completude(T,A,D),
    print('Teoria '), print(T), nl,  print('Dedução '), print(D), nl.

| ?- comp([p,q], [] , ((p => q) => p) => p).

Teoria [-(p),-(q)]
Dedução [((p=>q)=>p)=>p,-((p=>q)=>p),p=>q,-(p)]

yes
```

[a] Para simplificar a dedução é possível eliminar as fórmulas repetidas que resultarem desta junção.

Prova do teorema da completude

Se A é uma tautologia então, para todo o modelo $\mathcal{M}$, verifica-se $\mathcal{M} \models A$; pela proposição 5, verifica-se também $\overline{\mathcal{M}} \vdash A$.

Suponhamos que existem n símbolos proposicionais $\{p_i\}_{i=1}^{n}$; então tem-se

$$\{p_1, p_2, \cdots, p_n\} \vdash A \qquad \text{e} \qquad \{\neg p_1, p_2, \cdots, p_n\} \vdash A$$

Pelo teorema da dedução verifica-se

$$\{p_2, \cdots, p_n\} \vdash p_1 \supset A \qquad \text{e} \qquad \{p_2, \cdots, p_n\} \vdash \neg p_1 \supset A$$

Usando a regra (e) da proposição 4 conclui-se

$$\{p_2, \cdots, p_n\} \vdash A$$

Eliminámos a hipótese p_1 da dedução de A. Repetindo este processo podemos eliminar sucessivamente todas as hipóteses $p_2, \cdots p_n$ e concluir que se verifica

$$\vdash A$$

Conclui-se assim que, sendo A uma tautologia, é também um teorema.

Esta mesma construção é generalizável à situação em que se verifica $\Gamma \models A$. Usando os mesmos argumentos pode-se construir a dedução $\Gamma \vdash A$.

1 Corolário O sistema de dedução hilbertiano é **consistente**; isto é, não existe nenhum teorema A em que $\neg A$ seja também teorema.

(*Como todo o teorema é uma tautologia – correcção – não é possível que A e $\neg A$ sejam ambos tautologias.*)

2 Corolário O sistema de dedução hilbertiano tem **negação consistente**; isto é, verifica-se $\Gamma \vdash \neg A$ se e só se $\Gamma \cup \{A\} \vdash \bot$.

(*Pela correcção e completude, verifica-se $\Gamma \vdash \neg A$ se e só se todo o modelo que valida Γ não valida A. Portanto não existe qualquer modelo que valide $\Gamma \cup \{A\}$. Concluímos então $\Gamma \cup \{A\} \models \bot$ e, pela completude, $\Gamma \cup \{A\} \vdash \bot$.*)