UNIVERSIDADE BEIRA INTERIOR

Lógica Computacional

Aula 2- Sintaxe da Lógica Proposicional

António Ravara Simão Melo de Sousa

NOVALINCS
LABORATORY FOR COMPUTER SCIENCE AND INFORMATICS

(re)lease
REliable And SEcure Computation Group

LISP
LABORATORY OF COMPUTER SCIENCE, SYSTEMS AND PARALLELISM

Sintaxe da lógica Propocisional

Introdução

Descrição informal

Um sistema formal de raciocínio, constituido por:

- Um **alfabeto** (conjunto de símbolos).
- Uma **linguagem** (conjunto de fórmulas).
- Uma **semântica** (para valoração de símbolos e fórmulas).
- Um **cálculo** (sistema sintático de prova, para raciocinar).

Objecto

- Ocupa-se do estudo do comportamento dos **conectivos** lógicos (negação, disjunção, conjunção, implicação e equivalência) e das regras que os manipulam.
- Linguagem das **asserções** ou **proposições**: afirmações que são ou verdadeiras ou falsas.
- Linguagem construida a partir de símbolos proposicionais (asserções básicas) e conectivos lógicos (ligam asserções).

Exemplos

- Básicas:
  - hoje chove;
  - todo o natural par maior que 2 é a soma de dois primos.
- Compostas:
  - estudo hoje **ou** amanha;
  - jogo hoje **e** amanha;
  - **se** tenho aulas **então** vou à Faculdade;
  - n é par se e só se $\mathrm{mod}(n, 2) = 0$.
- **Não são asserções**:
  - passe-me o sal, se faz favor;
  - quanto mais depressa, mais devagar.

Sintaxe

Alfabeto e linguagem

Definição da sintaxe da lógica proposicional

Objectivo

- Obter a linguagem formal das fórmulas proposicionais.
- A partir de um alfabeto (conjunto de símbolos, representando asserções) define-se como construir palavras (sequências finitas de símbolos, ditas **fórmulas**).

Definição 2.1: alfabeto proposicional sobre um conjunto P

Seja P um conjunto numerável (de símbolos proposicionais). O **alfabeto proposicional sobre P**, denotado Alf_P, é o conjunto constituido:

- por cada um dos elementos de P (as **asserções básicas**);
- pelo símbolo $\bot$ (chamado **falso**, ou **absurdo**);
- pelos conectivos **disjunção**, $\vee$, **conjunção**, $\wedge$ e **implicação**, $\rightarrow$;
- pelos parênteses esquerdo e direito, (e).

Definição da sintaxe da lógica proposicional

Nem toda a sequência de símbolos do alfabeto é uma palavra da linguagem.
Vamos defini-la com (axiomas e) regras.

Definição 2.2: A linguagem proposicional induzida por Alf_P

denotada F_P, é o conjunto definido indutivamente pelas seguintes regras:

- *BOT*: $\bot \in F_P$
- *PROP*: se $p \in P$ então $p \in F_P$
- *DIS*: se $\varphi, \psi \in F_P$ então $(\varphi \vee \psi) \in F_P$
- *CON*: se $\varphi, \psi \in F_P$ então $(\varphi \wedge \psi) \in F_P$
- *IMP*: se $\varphi, \psi \in F_P$ então $(\varphi \rightarrow \psi) \in F_P$

Terminologia

- Os elementos de F_P dizem-se **fórmulas**.
- Os elementos de P e o símbolo $\bot$ dizem-se fórmulas **atómicas**.

Definição 2.3: abreviaturas

São úteis novos conectivos para abreviar alguns tipos de fórmulas.

- Negação: $\neg\varphi \overset{\text{abv}}{=} \varphi \to \bot$;
- Verdade: $\top \overset{\text{abv}}{=} \neg\bot$;
- Equivalência: $\varphi \leftrightarrow \psi \overset{\text{abv}}{=} (\varphi \to \psi) \wedge (\psi \to \varphi)$.

Convenções

- Para simplificar a notação omitem-se por vezes os parênteses mais exteriores das fórmulas.
- Consideramos que o conectivo $\neg$ tem precedência.

Exemplos

- $\neg(\neg\varphi \wedge \psi) \overset{\text{abv}}{=} ((\varphi \to \bot) \wedge \psi) \to \bot$
- $\neg\varphi \leftrightarrow (\psi \vee \delta) \overset{\text{abv}}{=} ((\varphi \to \bot) \to (\psi \vee \delta)) \wedge ((\psi \vee \delta) \to (\varphi \to \bot))$

```
(* tipo das variáveis *)
type variavel = string

(* tipo das fórmulas lógicas proposicionais com todas *)
(* as conectivas clássicas                            *)
type formula =
  (* casos de base, na definição indutiva *)
  | Var      of variavel              (* variável     *)
  | Verdade                           (* Top          *)
  | Falso                             (* Bottom       *)
  (* constructores, na definição indutiva            *)
  | Implica  of formula * formula (* implicação   *)
  | Equivale of formula * formula (* equivalência *)
  | Ou       of formula * formula (* disjunção    *)
  | E        of formula * formula (* conjunção    *)
  | Nao      of formula           (* negação      *)
```

```
type fp = (* Negação, Top e Equivale são macros*)
  | Vf  of variavel  (* variável   *)
  | Bf               (* bottom     *)
  | Ouf of fp * fp   (* disjunção  *)
  | Ef  of fp * fp   (* conjunção  *)
  | IF  of fp * fp   (* implicação *)

(* conversão de formula para fp *)
let rec f2fp = function
  | Var x           -> Vf x
  | Verdade         -> f2fp (Nao Falso)
  | Falso           -> Bf
  | Implica  (a,b) -> If  (f2fp a , f2fp b)
  | Ou       (a,b) -> Ouf (f2fp a , f2fp b)
  | E        (a,b) -> Ef  (f2fp a , f2fp b)
  | Nao       a    -> If  (f2fp a , Bf)
  | Equivale (a,b) -> Ef  (If (f2fp a , f2fp b), If (f2fp b , f2fp a))
```

Sintaxe

Exemplos

Não são necessariamente fórmulas todas as sequências de símbolos do alfabeto.

As sequências seguintes não são fórmulas

- $pq \notin F_P$, porque não se podem fazer sequências de símbolos proposicionais;
- $(p\vee) \notin F_P$, porque a disjunção é um operador binário e a expressão só tem um operando;
- $(p \rightarrow (\vee q)) \notin F_P$, porque a implicação é um operador binário que deve ter como argumentos/operandos duas fórmulas; no entanto, apesar de p ser uma fórmula, $(\vee q)$ não o é.

Facilmente se vê que não foram seguidas as regras para definir fórmulas.

Como mostrar que $(p \wedge ((p \vee q) \rightarrow r)) \in F_P$?

Prova de fórmulas

Sejam $p, q, r \in P$.

1. Por *PROP*, tem-se que $p \in F_P$.
2. Por *PROP*, tem-se que $q \in F_P$.
3. Por *PROP*, tem-se que $r \in F_P$.
4. Por *DIS*, com 1 e 2, tem-se que $(p \vee q) \in F_P$.
5. Por *IMP*, com 4 e 3, tem-se que $((p \vee q) \rightarrow r) \in F_P$.
6. Por *CON*, com 1 e 5, tem-se que $(p \wedge ((p \vee q) \rightarrow r)) \in F_P$.

Em Ocaml: das string as fórmulas e vice versa

assuma a existência da função

```
val to_fp : string -> fp
```

(que não é de definição imediata, basea-se na escrita de um parser) podemos definir facilmente a função inversa

```
# let f =   to_fp ‘‘(A & B -> (A | B))’’ ;;
val f : fp = If (Ef (Vf ‘‘A’’, Vf ‘‘B’’), Ouf (Vf ‘‘A’’, Vf ‘‘B’’))
```

```
  let rec of_fp = function
  | Vf v       ->  v
  | Bf         -> "FALSE"
  | If  (f, g) -> ("( "^ of_fp f ^ " -> " ^ of_fp g ^" )")
  | Ef  (f, g) -> ("( "^ of_fp f ^ " & "  ^ of_fp g ^" )")
  | Ouf (f, g) -> ("( "^ of_fp f ^ " | "  ^ of_fp g ^" )")
```

```
#  print_endline (of_fp f);;
( ( A & B ) -> ( A | B ) )
- : unit = ()
```

Exemplos de fórmulas da lógica proposicional

Asserções básicas e compostas

Considere as seguintes asserções básicas.

- p: 'estudo hoje'
- q: 'estudo amanhã'
- r: 'tenho exame amanhã'.

Constroiem-se as seguintes asserções compostas:

- estudo hoje **ou** estudo amanhã: $p \vee q$
- estudo hoje **e** estudo amanhã: $p \wedge q$
- **se** tenho exame amanhã **então** estudo hoje **e** estudo amanhã: $r \rightarrow (p \wedge q)$

Representação da linguagem natural

Modelação/Representação de conhecimento

Tradução da linguagem natural para lógica proposicional

Intuição

- Os símbolos proposicionais representam asserções básicas (afirmações verdadeiras ou falsas): 'estudo hoje'.
- O conectivo disjunção representa alternativa.
  " hoje ou amanhã" $(p \vee q)$.
- O conectivo conjunção indica que a frase só é verdade se cada uma das partes o for.
  A fórmula $p \wedge q$ traduz as frases "p e q", "tanto p como q", "p tal como q", etc,...
- O conectivo implicação captura consequência. À esquerda está o **antecedente** (hipótese) e à direita o **consequente** (tese).
  A fórmula $p \rightarrow q$ traduz as frases "se p então q", "se p, q", "q se p", "p só se q", "caso p então q", "caso p, q", "como p, q", etc,...

Representação da linguagem natural

Exemplos

Asserções básicas e compostas

- "gosto de lógica" escreve-se p;
  "gosto de álgebra" escreve-se q;
  "gosto de análise" escreve-se r;
- "gosto de lógica e de álgebra" escreve-se $p \wedge q$;
- "gosto de lógica ou gosto de álgebra" escreve-se $p \vee q$;
- "gosto de lógica ou de álgebra e de análise" é ambígua,mas "gosto de lógica ou gosto de álgebra e de análise" já não;
  nas fórmulas, os parênteses desambiguam: $p \vee q \wedge r$ é ambígua, mas $p \vee (q \wedge r)$ já não.
- Há ambiguidades difíceis de resolver: "o Pedro foi ao médico e ficou doente" tanto pode querer dizer $m \wedge d$ como $m \rightarrow d$.

Outras ambiguidades da linguagem natural

- "o Pedro ficou doente e foi ao médico " à primeira vista deve querer dizer $d \rightarrow m$ mas também d (ou seja $(d \rightarrow m) \wedge d$), e não nessariamente $d \wedge m$. Mas pensando bem...
- "gosto de programação ou gosto de futebol". Posso gostar dos dois? de um só deles?
- "João, se não comeres a sopa tens castigo". O João come a sopa e tem castigo. O pai mentiu?
- "eu só digo mentiras". Será isso verdade?
- "ou gosto do Benfica ou gosto do Sporting". é uma disjunção?

  se traduzirmos por $GostarBenfica \vee GostarSporting$, quais são as condições para acharmos esta frase verdade? falsa?

  se traduzirmos por
  $(GostarBenfica \vee GostarSporting) \wedge \neg(GostarBenfica \wedge GostarSporting)$,
  quais as são condições para acharmos que esta frase é verdade? falsa?

Definições indutivas e provas

Definição indutiva de conjuntos

Menor conjunto gerado por regras

É uma forma "construtiva" (ou incremental) de definir conjuntos **infinitos**:
Como funciona?

- Diz-se primeiro ("axiomatiza-se") quais são os elementos "básicos" do conjunto (em número **finito**).
- Dão-se depois "regras" (em número **finito**) para obter novos elementos a partir dos que já estão no conjunto.

O conjunto resultante contém **todos** os elementos que se podem gerar com as regras, e **apenas** esses.

Definição indutiva: menor conjunto gerado por regras

Notar bem

- Aplicando infinitas vezes as regras, obtém-se um número infinito de elementos (a partir de um número finito de axiomas e regras).
- Cada elemento do conjunto tem no entanto uma justificação, prova ou **derivação finita**: a sequencia das regras aplicadas para o obter.

Exemplos

A linguagem F_P da Lógica Proposicional foi definida desta forma.
Outro exemplo essencial é o dos números naturais.

Definições indutivas e provas

Exemplos

Regras

- Fixa-se primeiro que zero é natural (axioma *ZERO*): $0 \in \mathbb{N}_0$
- Diz-se depois como obter novos naturais a partir dos já gerados (regra *SUCC*): se $n \in \mathbb{N}_0$ então $n + 1 \in \mathbb{N}_0$

Justificações de pertença a conjunto

São cadeias (finitas) de aplicações de regras, terminando num axioma, que mostram como se gerou o elemento.

- $3 \in \mathbb{N}_0$, pois $3 = 2 + 1$ (regra *SUCC*),
  e $2 \in \mathbb{N}_0$, pois $2 = 1 + 1$ (regra *SUCC*),
  e também $1 \in \mathbb{N}_0$, pois $1 = 0 + 1$ (regra *SUCC*),
  uma vez que $0 \in \mathbb{N}_0$ (axioma *ZERO*).
- $3,5 \notin \mathbb{N}_0$, pois $3,5 = 2,5 + 1$ (regra *SUCC*),
  e $2,5 = 1,5 + 1$ (regra *SUCC*),
  e $1,5 = 0,5 + 1$ (regra *SUCC*),
  e $0,5 = -0,5 + 1$ (regra *SUCC*),
  ... a sequência não termina porque não "encontra" o axioma.

Definição indutiva de pilhas de inteiros

Regras

- Fixa-se primeiro que *vazia* é pilha (axioma *VAZIA*): $vazia \in PilhaInt$
- Diz-se depois como obter uma nova pilha a partir de um inteiro e de uma pilha já gerada (regra *PUSH*):

  se $i \in \mathbb{Z}$ e $p \in PilhaInt$ então $push(i, p) \in PilhaInt$

Justificações de pertença a conjunto

$push(4, push(-7, push(-1, vazia))) \in PilhaInt$,

pois $4 \in \mathbb{Z}$

e $push(-7, push(-1, vazia)) \in PilhaInt$ (regra *PUSH*),

uma vez que $-7 \in \mathbb{Z}$ e $push(-1, vazia) \in PilhaInt$ (regra *PUSH*),

já que $-1 \in \mathbb{Z}$ e $vazia \in PilhaInt$ (axioma *VAZIA*).

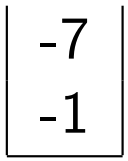
Pilhas de inteiros

A definição indutiva introduziu uma (nova) linguagem (formal) para falar de valores abstractos.

Denotação

- Sintaxe: regras para escrever palavras (**termos**) da linguagem.
  Definição indutiva gera linguagem infinita de palavras finitas.
- Semântica: associa sentido (*i.e.*, **valores**) às palavras.
  Função definida (também) indutivamente nas regras da linguagem:
  a cada termo corresponde um valor - a sua **denotação**.
  Definição indutiva de funções?
  Sim, uma função é um conjunto de pares ordenados.

Exemplo: o termo $push(-7, push(-1, vazia))$ denota o valor



OCaml: definições indutivas = tipos soma

```
(* tipo soma para as pilhas de inteiros, tal como definido inditivamente*)
type stack = Vazia | Push of int * stack

(* funcções utilitárias*)
let is_empty p = p=Vazia
let pop p = match p with Vazia -> failwith "Vazia" | Push (e,pp)-> pp
let top p = match p with Vazia -> failwith "Vazia" | Push (e,pp)-> e
let push e p = Push (e,p)
let print_stack p = (* omitido, deixado ao leitor *)
```

```
(* é igual a     top (push 8 (push 7 (push 1 Vazia)))*)
#let v = Vazia |> push 1 |> push 7 |> push 8 |> top;;
val v : int = 8
#let p = Vazia |> push 1 |> push 7 |> push 8 |> pop;;
val p : stack = Push (7, Push (1, Vazia))
# top p;;
- : int = 7
# print_stack p;;
- : string = «-|7|1||"
```

Conceitos auxiliares

Subfórmulas

Definição 3.1: subfórmulas de uma fórmula

O conjunto SBF(φ) das subfórmulas de $\varphi \in F_P$ é definido indutivamente pelas seguintes regras:

- SBF(φ) $= \{\varphi\}$, se φ é atómica;
- seja $\delta \stackrel{\text{def}}{=} \varphi \vee \psi$, $\delta \stackrel{\text{def}}{=} \varphi \wedge \psi$ ou $\delta \stackrel{\text{def}}{=} \varphi \rightarrow \psi$;

  SBF(δ) $= \{\delta\} \cup$ SBF(φ) $\cup$ SBF(ψ).

Definição 3.2: símbolos proposicionais de uma fórmula

O conjunto SMB(φ) $\stackrel{\text{def}}{=}$ SBF(φ) $\cap P$ contém os símbolos proposicionais de uma fórmula $\varphi \in F_P$.

Símbolos proposicionais e subfórmulas de uma fórmula

Sejam $p, q, r \in P$ e $\neg p, (p \vee q) \rightarrow r \in F_P$.

- $\text{SMB}(\neg p) = \{p\}$

  $\text{SBF}(\neg p) = \text{SBF}(p \rightarrow \bot) = \{\neg p\} \cup \text{SBF}(p) \cup \text{SBF}(\bot) = \{\neg p, p, \bot\}$;

- $\text{SMB}((p \vee q) \rightarrow r) = \{p, q, r\}$

  $\text{SBF}((p \vee q) \rightarrow r) = \{(p \vee q) \rightarrow r\} \cup \text{SBF}(p \vee q) \cup \text{SBF}(r) = \{(p \vee q) \rightarrow r\} \cup \{p \vee q\} \cup \text{SBF}(p) \cup \text{SBF}(q) \cup \{r\} = \{(p \vee q) \rightarrow r, p \vee q, p, q, r\}$.

```
let smb f =  (*calcula os simbolos de uma fórmula fp*)
  let rec smb_aux fo l =  match fo with
    (* podemos melhorar bastante o teste de redundância*)
    | Vf   v      -> if List.mem v l then l else v::l
    | Bf          -> l
    | If  (f, g) -> let l' = smb_aux f l in smb_aux g l'
    | Ef  (f, g) -> let l' = smb_aux f l in smb_aux g l'
    | Ouf (f, g) -> let l' = smb_aux f l in smb_aux g l'
  in smb_aux f []

let smb_list lf = fold_left (fun l f -> smb_aux f l) [] lf
```

```
# let f1 = f2fp (to_formula ``!p'');;
val f1 : fp = If (Vf ``p'', Bf)
# smb f1;;
- : variavel list = [``p'']
# let f2 = to_fp ``((p | q) -> r)'';;
val f2 : fp = If (Ouf (Vf ``p'', Vf ``q''), Vf ``r'')
# smb f2;;
- : variavel list = [``r''; ``q''; ``p'']
```

```
let smf form = (* lista as subfórmulas de form*)
(*esta tabela de hash auxiliar permite testar a redundância eficientemente*)
 let ht = Hashtbl.create 7 in
 let rec smf_aux form l=
   match form with
   | Vf _
   | Bf          ->  if Hashtbl.mem ht form then l
                     else let () = Hashtbl.add ht form () in
                          form::l
   | If  (f, g)
   | Ef  (f, g)
   | Ouf (f, g) ->  if Hashtbl.mem ht form then l
                    else let () = Hashtbl.add ht form () in
                         let l' = smf_aux f (form::l) in
                            smf_aux g l'
 in smf_aux form []
```

```
# smf f1;;
- : fp list = [If (Vf ‘‘p’’, Bf); Vf "p"; Bf]
# smf f2;;
- : fp list = [If (Ouf (Vf ‘‘p’’, Vf ‘‘q’’), Vf ‘‘r’’) ; Ouf (Vf ‘‘p’’, Vf ‘‘q’’); Vf ‘‘p’’; Vf ‘‘q’’; Vf ‘‘r’’]
```

Prova de fórmula

Sequências da linguagem

Como mostrar que dada sequência de símbolos do alfabeto é fórmula da linguagem?

Uma prova é uma sequência de fórmulas, sendo

- cada uma obtida por aplicação de uma regra, eventualmente usando fórmulas anteriores (na sequência) como hipóteses dessa regra;
- a última fórmula da sequência é a conclusão desejada.

Definição 3.3: prova de fórmula

Uma **prova** para $\varphi \in F_P$ é uma sequência $\varphi_1, \ldots, \varphi_n$ de fórmulas de F_P, com $n \geq 1$, tal que:

- $\varphi = \varphi_n$;
- para qualquer $1 \leq i \leq n$ tem-se que φ_i resulta de aplicar uma das regras da definição de F_P a fórmulas da sequência $\varphi_1, \ldots, \varphi_{n-1}$.

Prova de fórmula

Árvores de derivação

A ideia

- Uma prova ou inferência é apresentada em árvore, dita de dedução ou derivação.
- Cada árvore é construida a partir de árvores singulares (ou folhas) utilizando-se as regras da definição indutiva.
- Obtém-se um novo nível da árvore por aplicação de uma regra.
- A etiquetas dos nós são fórmulas.
  - As fórmulas nas folhas resultam de axiomas.
  - A fórmula na raíz é a conclusão da prova. Diz-se que a árvore é uma derivação dessa fórmula.

Exemplo de derivação

Sejam $p, q, r \in P$.
O termo $(p \to (q \wedge (r \vee p))) \in F_P$ (é fómula).

Prova:

$$\dfrac{\overline{p}\ (\text{PROP}) \qquad \dfrac{\overline{q}\ (\text{PROP}) \qquad \dfrac{\overline{r}\ (\text{PROP}) \qquad \overline{p}\ (\text{PROP})}{(r \vee p)}\ (\text{DIS})}{(q \wedge (r \vee p))}\ (\text{CON})}{(p \to (q \wedge (r \vee p)))}\ (\text{IMP})$$

Note que as provas não são (necessariamente) únicas.

Indução estrutural

Definição

Motivação

Uma regra de uma definição indutiva de um conjunto S, introduzindo termos construidos com dado operador n-ário op, tem a forma geral seguinte.

Se para qualquer i tal que $1 \leq i \leq n$, com $n \geq 0$, se tem que $e_i \in S_i$, então $\mathrm{op}(e_1, \ldots, e_n) \in S$,
sendo cada S_i ou o conjunto S ou outro conjunto já previamente definido.

Nota

- Tal como se faz indução sobre os naturais, pode-se fazer indução sobre (os construtores de) qualquer conjunto definido indutivamente.
- Na verdade. a indução natural é um caso particular desta indução **estrutural**.

Definição 3.4

Seja S um conjunto definido indutivamente e P um predicado sobre elementos de S.

Se para cada construtor (k-ário) op da definição indutiva de S,
se tem $P(\text{op}(e_1, \ldots, e_k))$, se $P(e_i)$, para cada $e_i \in S$ (com $1 \leq i \leq k$),
então tem-se $P(e)$ para qualquer $e \in S$.

Este princípio de indução, por sua vez, é um caso particular da chamada **indução generalisada** ou Noetheriana.

Definição 3.5

Seja $(S, \leq)$ um conjunto parcialmente ordenado bem fundado e P um predicado sobre elementos de S.

$$\forall r \in S\ (\forall s \in S\ (s < r \rightarrow P(s)) \rightarrow P(r)) \rightarrow \forall t \in s.P(t)$$

Provas por indução estrutural: um exemplo

Relembre a Definição 3.2 de conjunto de símbolos proposicionais de uma fórmula.

Considere o conjunto F_P das fórmulas da Lógica Proposicional e a propriedade $P(\varphi)$: SMB(φ) **é finito**.

Prova-se por indução estrutural que P se verifica para todos os elementos de F_P.

- Casos Base.
  - $\varphi = p$, para $p \in P$. Pela definição, SMB(p) $= \{p\}$, que é finito.
  - $\varphi = \bot$. Pela definição, SMB($\bot$) $= \emptyset$, que é finito.
- Passo (faz-se apenas para $\varphi = \psi_1 \wedge \psi_2$; os restantes casos são semelhantes).
  Pela definição, SMB(φ) $=$ SMB(ψ_1) $\cup$ SMB(ψ_2),
  que é finito pois por hipótese de indução SMB(ψ_1) e SMB(ψ_2) são finitos,
  e a união de conjuntos finitos é finita.

Proposição 3.1

O termo $\varphi \in F_P$, se e só se, existe uma prova para φ.

Prova do sentido "só se": por indução estrutural

Hipótese: termo $\varphi \in F_P$. **Tese**: existe uma prova para φ.

Base: $\varphi = \bot$ ou $\varphi = p$ para $p \in P$. Por *BOT* ou por *PROP*, a sequência φ é uma prova.

Passo: seja $\varphi = \psi_1 \vee \psi_2$ (os restantes casos têm prova semelhante).

Por hipótese de indução existem sequências $\psi_{11} \cdots \psi_{1n}$ e $\psi_{21} \cdots \psi_{2m}$ (com $n, m \geq 1$), que provam respectivamente ψ_1 e ψ_2.

Logo, como $\psi_1 = \psi_{1n}$ e $\psi_2 = \psi_{2m}$, por *DIS*

a sequência $\psi_{11} \cdots \psi_{1n}\psi_{21} \cdots \psi_{2m}\varphi$ prova φ.

Prova do sentido "se": por indução no comprimento da prova

Hipótese: existe uma prova para φ. **Tese**: termo $\varphi \in F_P$.

Base: a prova é a sequência σ com comprimento 1.
Então, a sequência tem apenas uma fórmula ($\sigma = \varphi$),
e logo, $\varphi = \bot$ ou $\varphi = p$ para $p \in P$.
Por *BOT* ou por *PROP* (respectivamente), a fórmula $\varphi \in F_P$.

Passo: A prova é uma sequência σ de comprimento $1 + n$, sendo $\varphi = \psi_1 \vee \psi_2$ a última fórmula da sequência (os restantes casos têm prova semelhante).
Então, a sequência tem duas subsequências σ_1 e σ_2, de comprimentos n_1 e n_2 com $n = n_1 + n_2$, sendo a primeira subsequência a prova de ψ_1 e a segunda a prova de ψ_2.

Por hipótese de indução $\psi_1 \in F_P$ e $\psi_2 \in F_P$.
Logo, por *DIS*, $\varphi \in F_P$.