

3. Introdução da Conjunção:

$n \alpha$
 $m \beta$
 $\vdash \alpha \wedge \beta$ é verdadeiro

Caminho sempre verdadeiro, a qualquer momento se α e β são verdadeiros, $\alpha \wedge \beta$ é verdadeiro

$\vdash \alpha \wedge \beta \text{ I}_\wedge(n, m)$

exemplo:

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

1 P Prem

2 Q Prem

3 $P \wedge Q \text{ I}_\wedge(1, 2)$

4. Regra da Eliminação da Conjunção:

$n \alpha \wedge \beta$
 \vdash ou
 $m \alpha$ E_\wedge, n $m \beta E_\wedge, n$

exemplo:

$$\{P \wedge Q, R\} \vdash P \wedge R$$

1 $P \wedge Q$ Prem

2 R Prem

3 P $E_\wedge, 1$

4 R Rep, 2

5 $P \wedge R \text{ I}_\wedge(3, 4)$

5. Próva Hipotética:

$n \alpha \text{ hip}$ Considera-se α verdade

6. Regra da Reiteração:

$c \quad \beta$
 $n \alpha \text{ hip}$
 \vdash
 $m \beta \text{ Reit } c$

Reit n
 Põe uma premissa para dentro da hipótese

7. Regra da Introdução da Implicação:

$n \alpha$
 \vdash
 $m \beta$

$I \rightarrow, (n, m)$

$m+1 \alpha \rightarrow \beta$

exemplo:

$$\{P\} \vdash Q \rightarrow (P \wedge Q)$$

1 P Prem

2 Q hip

3 P Reit, 1

4 $P \wedge Q \text{ I}_\wedge(3, 4)$

5 $Q \rightarrow (P \wedge Q) \text{ I} \rightarrow(2, 5)$

$(\{\}, P \rightarrow (Q \rightarrow P))$

$P \rightarrow (Q \rightarrow P)$ é um teorema

1 P hip
 2 Q hip
 3 P Reit, 1
 4 $Q \rightarrow P \text{ I} \rightarrow(2, 3)$
 5 $P \rightarrow (Q \rightarrow P) \text{ I} \rightarrow(1, 5)$

8. Regra da Eliminação da Implicação: (modus ponens)

$n \alpha$
 $m \alpha \rightarrow \beta$
 \vdash
 $k \beta$

$E \rightarrow, (n, m)$

exemplo:

$$\{P \rightarrow (Q \rightarrow R)\} \vdash (P \rightarrow Q) \rightarrow (P \rightarrow R)$$

<u>exemplo:</u>	$\{P \vee Q\} \vdash Q \vee P$
1	$P \vee Q$ Prem
2	P hip
3	$Q \vee P$ IV, 2
4	Q hip
5	$Q \vee P$ IV, 4
6	$Q \vee P$ EV (1, (2,3), (4,5))

13. Introdução da Equivalecia:

$$\begin{array}{l} n \quad \alpha \rightarrow B \\ m \quad B \rightarrow \alpha \\ k \quad \alpha \leftrightarrow B \end{array} \text{ I} \Leftrightarrow (n, m)$$

12. Eliminação da Equivalecia:

$$\begin{array}{l} n \quad \alpha \leftrightarrow B \\ \vdots \\ m \quad \alpha \rightarrow B \end{array} \text{ E} \leftrightarrow, n \qquad \text{ou} \qquad \begin{array}{l} n \quad \alpha \leftrightarrow B \\ \vdots \\ m \quad B \rightarrow \alpha \end{array} \text{ E} \leftrightarrow, n$$

• Reduzir as Linhas de uma Pirra

- Utilizando Regras de inferência derivadas: padrão de raciocínio correspondente à aplicação de várias regras de inferência
- Utilizando teoremas (π tem premissas)

• Regra de Resolução

- Abordagem ao sistema dedutivo baseado numa única Regra de inferência;
- Princípio da Resolução;
- Obliga a que as fbf estejam na forma cláusula;

• Cláusula

- Cláusula \equiv literal ou disjunção de literais;

• Literal

- Símbolo proposicional ou a sua negação;

• Conjuncional Normal Form

- fbf na CNF \equiv conjunção de cláusulas;

exemplos:

$$\text{CNF: } \neg A \quad A \vee \neg B \quad \neg A \wedge (B \vee C)$$

$$\text{Não estão na CNF: } (A \wedge B) \vee C \quad A \wedge (B \vee (D \wedge E))$$

Notação:

$$(P \vee \neg Q \vee \neg R) \wedge (\neg P \vee S) \wedge (Q \vee R \vee S) \equiv \{\{P, \neg Q, \neg R\}, \{\neg P, S\}, \{Q, R, S\}\}$$

• Conversão para CNF

1. Eliminação da implicação:

$$\alpha \rightarrow B \equiv \neg \alpha \vee B$$

exemplo:

$$P \rightarrow \neg (Q \vee ((R \wedge S) \rightarrow P)) \equiv \neg P \vee \neg (Q \vee ((R \wedge S) \rightarrow P)) \equiv$$

$$\equiv \neg P \vee \neg (Q \vee (\neg (R \wedge S) \vee P))$$

2. Redução do domínio da negação:

$$\neg \neg \alpha \equiv \alpha$$

$$\neg (\alpha \vee B) \equiv \neg \alpha \wedge \neg B$$

$$\neg (\alpha \wedge B) \equiv \neg \alpha \vee \neg B$$

3. Obtenção da forma conjuntiva normal:

$$\alpha \vee (B \wedge \gamma) \equiv (\alpha \vee B) \wedge (\alpha \vee \gamma)$$

$$(\neg P \vee \neg Q) \wedge (\neg P \vee R) \wedge (\neg P \vee S) \wedge (\neg P \vee \neg R) \rightarrow \{\{\neg P, \neg Q\}, \{\neg P, R\}, \{\neg P, S\}, \{\neg P\}\}$$

Lógica de 1ª Ordem

- Seja f_i^m um símbolo de função de aridade m e seja x uma variável, então:
 $\hookrightarrow n$ de argumentos

- x é um termo (x, y, z, \dots)
 - f_i^0 (i constantes) são termos ($a, b, c, \text{ constantes}$)
 - Se t_1, \dots, t_m são termos, então $f_i^m(t_1, \dots, t_m)$ é um termo (f, g, h, \dots)
- Nada mais é termo !!!**
- exemplos (termos):
 $x / \text{Portugal} / \text{capital}(\text{Portugal}) / \text{pai}(\text{Maria}) / \text{capital}(x) \dots$

\hookrightarrow símbolo de função

- Seja:

A_1, \dots, A_n termos

P_i^m um símbolo de aridade m

Então:

$P_i^m(t_1, \dots, t_m)$ é uma fbf (atómica)

- Se α e β são fbf:

- $\neg\alpha, (\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), \exists x[\alpha], \forall x[\alpha]$ são fbf.
- **Nada mais são fbf**

exemplo:

existe-fronteira (Portugal, Espanha)

símbolo de predicado

Definição dos Predicados

estudante(x) é verdade se x é estudante

estuda(x, y) é verdade se x estuda y

gosta(x, y) é verdade se x gosta de y

diferente(x, y) é verdade se $x \neq y$

igual(x, y) é verdade se $x = y$

irrata-de(x, y) é verdade se y é irrrata de x

constantes: Pedro, IAED, LP

1. O Pedro é estudante \rightarrow estudante(Pedro)

2. O Pedro estuda LP e IAED \rightarrow estuda(Pedro, LP) \wedge estuda(Pedro, IAED)

3. O Pedro estuda LP ou IAED, mas não as duas
 \hookrightarrow (estuda(Pedro, LP) \vee estuda(Pedro, IAED)) \wedge \neg (estuda(Pedro, LP) \wedge estuda(Pedro, IAED))

4. O Pedro não estuda LP \rightarrow \neg estuda(Pedro, LP)

5. Todos os estudantes são esperitos $\rightarrow \forall x [\text{estudante}(x) \rightarrow \text{esperito}(x)]$

6. Existe um estudante esperito $\rightarrow \exists x [\text{estudante}(x) \wedge \text{esperito}(x)]$

$\exists x [\text{estudante}(x) \rightarrow \text{esperito}(x)]$
 \hookrightarrow se não existe estudante é falso

7. Todo o estudante gosta de um outro estudante
 $\hookrightarrow \forall x [\text{estudante}(x) \rightarrow \exists y [\text{estudante}(y) \wedge \text{gosta}(x, y)]]$

8. Todo o estudante gosta de um estudante diferente de si
 $\hookrightarrow \forall x [\text{estudante}(x) \rightarrow \exists y [\text{estudante}(y) \wedge \text{gosta}(x, y) \wedge \text{diferente}(x, y)]]$

Dedução Natural em Lógica de 1ª Ordem

n $\forall x [\alpha(x)]$

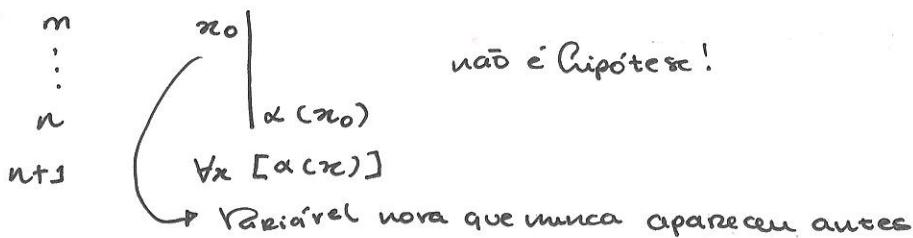
⋮

n $\alpha(t) \in V_m$ } Regra de eliminação de 1ª ordem

exemplo:

{ $p(t)$, $\forall x [\alpha(x) \rightarrow \beta(x)]$ } $\vdash \beta(t)$

- 1 $P(x)$ prem
- 2 $\forall x [P(x) \rightarrow Q(x)]$ prem
- 3 $P(t) \rightarrow Q(t)$ $E\forall, 2$
- 4 $\neg Q(t)$ $E \rightarrow (1, 3)$



$$\{\forall x [P(x) \rightarrow Q(x)], \forall x [P(x)]\} \vdash \forall x [$$

- 1 $\forall x [P(x) \rightarrow Q(x)]$ prem
- 2 $\forall x [P(x)]$ prem
- 3 $x_0 \quad \forall x [P(x) \rightarrow Q(x)]$ reit 1
- 4 $\forall x [P(x)]$ reit 2
- 5 $P(x_0) \rightarrow Q(x_0)$ $E\forall, 3$
- 6 $P(x_0)$ $E\forall, 4$
- 7 $Q(x_0)$ $E \rightarrow 5, 6$
- 8 $\forall x [Q(x)]$ $I\forall (3, 7)$

Exemplos

9. Nenhum estudante gosta do Pedro → $\neg \exists x [\text{estudante}(x) \wedge \text{gosta}(x, \text{Pedro})]$
↳ $\forall x [\text{estudante}(x) \rightarrow \neg \text{gosta}(x, \text{Pedro})]$
10. O Pedro tem pelo menos uma irmã → $\exists x [\text{irmã-de}(\text{Pedro}, x)]$
11. O Pedro não tem nenhuma irmã → $\neg \exists x [\text{irmã-de}(\text{Pedro}, x)]$
12. O Pedro tem no máximo 3 irmãs → $\forall x, y [\text{irmã-de}(\text{Pedro}, x) \wedge \text{irmã-de}(\text{Pedro}, y) \rightarrow \text{igual}(x, y)]$
13. O Pedro tem exatamente 3 irmãs
↳ $\exists x [\text{irmã-de}(\text{Pedro}, x) \wedge (\forall y [\text{irmã-de}(\text{Pedro}, y) \rightarrow \text{igual}(x, y)])]$
14. O Pedro tem, pelo menos, duas irmãs
↳ $\exists x, y [\text{irmã-de}(\text{Pedro}, x) \wedge \text{irmã-de}(\text{Pedro}, y) \wedge \neg (\text{igual}(x, y))]$

Dedução Natural em Lógica da 3ª Ordem

1. Eliminação do \forall

$$n \quad \forall x [\alpha(x)]$$

⋮

$$m \quad \alpha(t) \quad E\forall, n$$

exemplo:

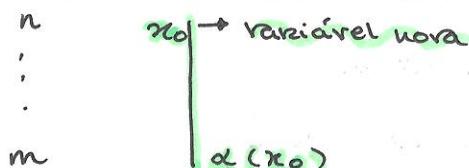
$$\{\forall x [P(x)]\} \vdash \exists x [P(x)]$$

$$1 \quad \forall x [P(x)] \text{ prem}$$

$$2 \quad P(t) \quad E\forall, 1$$

$$3 \quad \exists x [P(x)] \quad I\exists, 2$$

2. Introdução do \forall



$$m+1 \quad \forall x [\alpha(x)] \quad I\forall (m, n)$$

3. Introdução do \exists

$$n \quad \alpha(t)$$

⋮

$$m \quad \exists x [\alpha(x)] \quad I\exists, n$$

4. Eliminação do \exists

$$\begin{array}{ll} n & \exists x [\alpha(x)] \\ m & x_0 \mid \alpha(x_0) \text{ hip} \\ \vdots & \vdots \\ k & B \rightarrow \text{fórmula que não depende de } x_0 \\ k+1 & B \models \exists_{n, (m, k)} \end{array}$$

- exemplo:

$$\{\forall x [P(x) \rightarrow Q(x)], \exists x [P(x)]\} \vdash \exists x [Q(x)]$$

$$\begin{array}{l} 1 \quad \forall x [P(x) \rightarrow Q(x)] \text{ prem} \\ 2 \quad \exists x [P(x)] \text{ prem} \\ 3 \quad x_0 \mid P(x_0) \\ 4 \quad \forall x [P(x) \rightarrow Q(x)] \text{ ret 1} \\ 5 \quad P(x_0) \rightarrow Q(x_0) \in \forall, 4 \\ 6 \quad Q(x_0) \in \rightarrow 3, 5 \\ 7 \quad \exists x [Q(x)] \models \exists, 6 \\ 8 \quad \exists x [Q(x)] \in \exists 2, (3, 7) \end{array}$$

o Conversão para a forma clausal

1. Eliminação da implicação \rightarrow

$$2. Redução do domínio da \exists \rightarrow \forall x [\alpha(x)] = \exists x [\exists \alpha(x)] // \forall x [\exists \alpha(x)] = \forall x [\exists \alpha(x)]$$

3. Normalização de variáveis

↳ Garantir que uma variável só tem um quantificador associado;

4. Eliminação do \exists

$$\exists x \alpha(x) = \alpha(a) \xrightarrow{\text{constante de skolem}}$$

$$\forall x \forall y \exists z \alpha(x, y, z) = \forall x \forall y \alpha(x, y, f(x, y))$$

↳ função de skolem

5. Conversão para forma Prenex

Passar os \forall para a esquerda

6. Eliminação dos \forall

7. Distributividade

8. Obtenção de Conjuntos

- exemplo: $\forall x [P(x) \rightarrow (\forall y [P(y) \rightarrow P(f(x, y))] \wedge \forall y [Q(x, y) \rightarrow P(y)])]$

$$(1) \forall x [P(x) \rightarrow (\forall y [\neg P(y) \vee P(f(x, y))] \wedge \forall y [\neg Q(x, y) \vee P(y)])]$$

$$\forall x [\neg P(x) \vee (\forall y [\neg P(y) \vee P(f(x, y))] \wedge \forall y [\neg Q(x, y) \vee P(y)])]$$

$$(2) \forall x [\neg P(x) \vee (\forall y [\neg P(y) \vee P(f(x, y))] \wedge \exists y [\neg Q(x, y) \wedge \neg P(y)])]$$

$$\forall x [\neg P(x) \vee (\forall y [\neg P(y) \vee P(f(x, y))] \wedge \exists y [Q(x, y) \wedge \neg P(y)])]$$

$$(3) \forall x [\neg P(x) \vee (\forall y [\neg P(y) \vee P(f(x, y))] \wedge \exists z [Q(x, z) \wedge \neg P(z)])]$$

$$(4) \forall x [\neg P(x) \vee (\forall y [\neg P(y) \vee P(f(x, y))] \wedge [Q(x, g(x)) \wedge \neg P(g(x))])]$$

$$(5.6) \neg P(x) \vee (\neg P(y) \vee P(f(x, y))) \wedge [Q(x, g(x)) \wedge \neg P(g(x))]$$

$$(7) (\neg P(x) \vee \neg P(y) \vee P(f(x, y))) \wedge (\neg P(x) \vee Q(x, g(x))) \wedge (\neg P(x) \vee \neg P(g(x)))$$

o Resolução

$$\begin{array}{ll} - \{P, A\}, \{ \neg P, B\} & - \{P(x), R(x)\} \quad \{ \neg P(c), Q(a)\} \\ \checkmark \{A, B\} & \checkmark \{c/x\} \rightarrow \text{substitui } x \text{ por } c \\ & \{R(c), Q(a)\} \end{array}$$

↳ Infraestrutura geral

Substituição

- $S_1 = \{t_1/x_1, \dots, t_n/x_n\}$
- $S_2 = \{u_1/y_1, \dots, u_n/y_n\}$

Composição de Substituição ($S_1 \circ S_2$)

- Tal que para uma fbf α : $\alpha \cdot (S_1 \circ S_2)$ resulta em:
 1. Aplica-se S_2 aos termos de S_1 ;
 2. Adicionam-se os elementos de S_2 que dizem respeito a variáveis que não ocorrem em S_1 ;
 3. Removem-se $(t_i \circ S_2)/x_i$, tais que $t_i \circ S_2 = x_i$ (não podemos ter x/x)
- exemplo:

$$S_1 = \{f(y)/x, z/y\} \quad S_2 = \{a/x, b/y, y/z\}$$

$\begin{matrix} t_1 & t_2 \\ x_1 & x_2 \end{matrix}$

$S_1 \circ S_2$

1. $f(y) \cdot \{a/x, b/y, y/z\}/x, (z : \{a/x, b/y, y/z\}/y, a/x, b/y, y/z)$
2. $\{f(b)/x, y/y, a/x, b/y, y/z\}$
x e y já ocorriam em S_1
3. $S_1 \circ S_2 = \{f(b)/x, y/z\}$

• Um conjunto de fbf atómicas $\{\alpha_1, \dots, \alpha_n\}$ diz-se unificável se existir uma substituição que torna idênticas todas as suas fbf $\alpha_1 \cdot s = \dots = \alpha_n \cdot s$

exemplo: $\{P(a, y, z), P(x, b, z)\}$

↳ unificador do conjunto

unificador:

$$\left. \begin{array}{l} S_1 = \{a/x, b/y, c/z\} \\ S_2 = \{a/x, b/y, d/z\} \\ \vdots \end{array} \right\} \text{possibilidades}$$

Unificador Mais Geral

- Dado um conjunto de fbf atómicas $\{\alpha_1, \dots, \alpha_n\}$. O unificador mais geral (UMG) é o unificador s tal que se S_1 é um unificador de $\{\alpha_1, \dots, \alpha_n\}$ então existe S_2 tal que $S_1 = S_1 \circ S_2$, e UMG é o único

- exemplo: $\Delta = \{P(a, x, f(y)), P(u, v, w), P(a, r, f(c))\}$
 x, y, u, v, w, r são variáveis
 a, c são constantes

Conjunto de Discrepância	Substituições
$\{P(a, x, f(y)), P(u, v, w), P(a, r, f(c))\}$	$\{u, a\}$
	$\{a/u\} \rightarrow$ única hipótese
$\{P(a, x, f(y)), P(a, v, w), P(a, r, f(c))\}$	$\{u, v\}$
	$\{v/x, v/r\} \rightarrow$ escolha
$\{P(a, v, f(y)), P(u, v, w), P(a, r, f(c))\}$	$\{f(c), w, f(y)\}$
	$\{c/y, f(c)/w\}$
$\{P(a, v, f(c))\}$	
	UMG
	$\{a/u\} \circ \{v/x, v/r\} \circ \{c/y, f(c)/w\} = \{a/u, v/x, u/r, c/y, f(c)/w\}$
	↳ como o cardinal é 1, acabou.

- exemplos:

1. Os antepassados dos nossos ascendentes diretos são nossos antepassados;
2. Os nossos ascendentes diretos são nossos antepassados;
3. O Pedro é ascendente direto da Ana;
4. O Rui é um ascendente direto do Pedro;

a) O Rui é antepassado da Ana?

$AD(x, y)$: x é ascendente direto de y

$ant(x, y)$: x é antepassado de y

1. $\forall x, y, z [ant(x, y) \wedge AD(y, z) \rightarrow ant(x, z)]$
2. $\forall x, y [AD(x, y) \rightarrow ant(x, y)]$
3. $AD(Pedro, Ana)$
4. $AD(Rui, Pedro)$

forma clausal

- 1 $\{\neg AD(x, y), \neg AD(y, z), \neg ant(x, z)\}$
- 2 $\{\neg AD(x, y), \neg ant(x, y)\}$
- 3 $\{AD(Pedro, Ana)\}$
- 4 $\{AD(Rui, Pedro)\}$

a)

- 1 $\{\neg AD(x, y), \neg ant(x, y)\}$ prem → conclusão: $\{ant(Rui, Ana)\}$
- 2 $\{\neg ant(x, y), \neg AD(y, z), \neg ant(x, z)\}$ prem ↓ negar
- 3 $\{AD(Pedro, Ana)\}$ prem $\{\neg ant(Rui, Ana)\}$
- 4 $\{AD(Rui, Pedro)\}$ prem
- 5 $\{\neg ant(Rui, Ana)\}$ prem
- 6 $\{\neg ant(Rui, Pedro)\}$ Res(1,4) $\{Rui/x, Pedro/y\}$
- 7 $\{\neg AD(Pedro, z), \neg ant(Rui, z)\}$ Res(2,6) $\{Rui/x, Pedro/y\}$
- 8 $\{\neg ant(Rui, Ana)\}$ Res(3,7) $\{Ana/z\}$
- 9 $\{\}$ Res(5,8)

o Cláusula de Horn

- Cláusula que contém no máximo um literal positivo

- exemplo:

$$\left. \begin{array}{l} \{c, \neg P_1, \neg P_2\} \\ \{c\} \\ \{\neg P_1, \neg P_2\} \\ \{c, P_1\} \end{array} \right\} \text{cláusulas de Horn}$$

- Ao literal positivo, se existir, chamamos cabeça da cláusula;
- Aos literais negativos, se existirem, chamamos corpo da cláusula;
- Cláusula vazia \square ;
- Representação alternativa cabeça \rightarrow corpo;
- exemplo:

$$\{c, \neg P_1, \neg P_2\} \quad c \leftarrow P_1, P_2 \quad \text{regra} \\ \{c\} \quad c \leftarrow \text{afirmação}$$

$$\{\neg P_1, \neg P_2\} \quad \leftarrow P_1, P_2 \quad \text{objetivo}$$

o Tipos de Cláusulas de Horn

- Regras ou implicação (cabeça e corpo não vazios);
- Afirmação ou factos: corpo vazio;
- Objetivo: cabeça vazia;
- Cláusula vazia: corpo e cabeças vazios;

o Resolução Com Cláusulas de Horn

- $\alpha \leftarrow \gamma_1, \gamma_n$
- $\delta \leftarrow \beta_1, \dots, \beta_l \quad R_i : R_i + 1 \cdot R_n$

$(\delta \leftarrow B_1, \dots, B_{i-1}, \gamma_1, \dots, \gamma_n, B_{i+1}, \dots, B_n)$, $\xrightarrow{\Delta}$ UMG para α

* se δ não existir, tem um objetivo (prolog)*

Nota:

$$\begin{array}{ll} (\exists y \{ \exists p, b \}) & \{ P(x) \} \vdash \{ \exists p(c), B(\alpha) \} \\ \checkmark \{ B \} & \checkmark \{ c/x \} \\ & \{ B(c) \} \end{array}$$

$$\begin{array}{l} P(x) \leftarrow \\ \quad \checkmark \{ c/x \} \\ \quad B(\alpha) \leftarrow \end{array} \quad B(\alpha) \leftarrow P(c)$$

- exemplos:

Forma Cláusural

$$\{ \exists AD(x,y), \exists Aut(x,y) \} \dashv \dashv \exists Aut(x,y) \leftarrow \exists AD(x,y)$$

$$\{ \exists Aut(x,y), \exists AD(y,z), \exists Aut(x,z) \} \dashv \dashv Aut(x,z) \wedge \exists Aut(x,y), \exists AD(y,z)$$

$$\{ \exists AD(Pedro, Ana) \} \dashv \dashv \exists AD(Pedro, Ana) \leftarrow$$

$$\{ \exists AD(Rui, Pedro) \} \dashv \dashv \exists AD(Rui, Pedro) \leftarrow$$

$$\{ \exists Aut(Rui, Ana) \} \dashv \dashv \exists Aut(Rui, Ana) \leftarrow$$

$$1 \exists Aut(x,y) \leftarrow AD(x,y) \text{ prem}$$

$$2 \exists Aut(x,z) \leftarrow Aut(x,y), AD(y,z) \text{ prem}$$

$$3 \exists AD(Pedro, Ana) \leftarrow \text{prem}$$

$$4 \exists AD(Rui, Pedro) \leftarrow \text{prem}$$

$$5 \exists Aut(Rui, Ana) \leftarrow \text{prem}$$

$$6 \exists Aut(Rui, Pedro) \leftarrow \text{Res}(1,4), \{ Rui/x, Pedro/y \}$$

$$7 \exists Aut(Rui, z) \leftarrow \exists AD(Pedro, z) \text{ Res}(2,6) \{ Rui/x, Pedro/y \}$$

$$8 \exists Aut(Rui, Ana) \leftarrow \text{Res}(3,7) \{ Ana/z \}$$

$$9 \square \text{Res}(5,8) \{ \}$$

Cláusulas de Horn



o Programa

- Conjunto finito de cláusulas;

- Determinadas $\xrightarrow{\text{Regras}}$ factos

- Seja A um programa e α um objetivo. A resposta de A a α é uma substituição (pode ter razia) para as variáveis de α .

↳ A diz-se uma resposta correta se $A \cup \{ \alpha, \beta \}$ for contraditório.

? Resolução SLD

① Regra de seleção (dos literais do objetivo)

② Regra de procura (de cláusulas)

③ Estratégia de resolução linear (cláusula inicial + sucessores)

- exemplo:

Programa Anterior

Objetivo $\leftarrow \exists Aut(Rui, Ana)$

Função de seleção: $S(\leftarrow \alpha_1, \dots, \alpha_n) = \alpha_n$

Regra de Procura: a última que unifica

$\leftarrow \exists Aut(Rui, Ana)$

$\exists Aut(x_1, z_1) \leftarrow \exists Aut(x_1, y_1), \exists AD(y_1, z_1)$

$\{ Rui/x, Ana/z_1 \}$

$\leftarrow \exists Aut(Rui, y_1), \exists AD(y_1, Ana)$

$\exists AD(Pedro, Ana) \{ Pedro/y_1 \}$



o Prova SLD

- Sequência de objetivos $[Y_0, \dots, Y_n]$
- Se $Y_n = \square$ diz-se que a prova SLD é uma Refutação SLD, supondo que foram feitas as substituições S_0, \dots, S_n , a resposta de Δ a α é $(S_0 \circ S_1 \circ \dots \circ S_n)$

$v(\alpha)$

variáveis de α

> Árvore SLD

- | | |
|---|---|
| <ul style="list-style-type: none"> - Δ programa - α objetivo - função de seleção | <ul style="list-style-type: none"> Explorar todas as hipóteses Ramo cuja folha tem \square diz-se <u>bem-sucedido</u> Ramo que não tem \square diz-se <u>falhado</u> Os restantes são <u>infinitos</u>; |
|---|---|

exemplo da solução:

$$1. (\{u/x_1, b/z_1\} \circ \{a/x_1, b/x_1\} \circ \{b/x_3y\}) \Big|_{\{x_2\}} = \{a/x\}$$

$$2. (\{b/x, b/xuy\}) \Big|_{\{x,y\}} = \{b/x\}$$

o Prolog

- programa em prolog:



o Componentes Básicos

- Constantes: $a, 'a', 3$
- Variáveis: $X, Y, -X$ ↳ variável especial
- Termos compostos = função mais argumentos
- Literais \rightarrow predicados + argumentos
- Programa {
 - factos \rightarrow estudante (mae(a))
 - regras \rightarrow dia(x,y) :- oi(xy)
- Objetivo

comandos
1. /* */

o Semântica do Prolog

- Programa + objetivo + Refutação SLD

* função de seleção: escolhe o 1º literal do objetivo;

Regra de prova: escolhe a 1ª cláusula unificável com o literal do objeto em estudo;

o Programa

```

C1 :- ...
;
Cg :- corpo ...
;
CN :- ...

```

o Objetivo

? - O_1, O_2, \dots, O_n

assumindo que O_g é unificável com ci

? - corpo, O_2, \dots, O_n

- exemplo:

c1 amigo(harry, ron).

c2 amigo(lupin, harry).

c3 ligado(X, Y) :- amigo(X, Y).

c4 ligado(X, Z) :- ligado(X, Y), amigo(Y, Z).

? - ligado(lupin, ron)

✓ ligado(X, Y) :- amigo(X1, Y1)
{ lupin / X1, ron / Y1 }

? amigo(lupin, ron)

X

✓ ligado(X1, Z1) :- ligado(X1, Y1),
amigo(Y1, Z1)
{ lupin / X1, ron / Z1 }

? - ligado(lupin, Y1), amigo(Y1, ron)

✓ ligado(X2, Y2) :- amigo(X2, Y2)
{ lupin / X2, Y1 / Y2 }

? amigo(lupin, Y1)

✓ amigo(lupin, harry)
{ Y1 / harry }

? amigo(harry, ron)

✓ amigo(harry, ron)

□

True

o Unificação de Termos

- = (predicado da unificação)

- $\langle t_1 \rangle = \langle t_2 \rangle$ tem sucesso se $\langle t_1 \rangle$ e $\langle t_2 \rangle$ podem ser unificados.

- $\backslash=$ (negação do predicado da unificação) $\langle t_1 \rangle \backslash= \langle t_2 \rangle$ tem sucesso se $\langle t_1 \rangle$ e $\langle t_2 \rangle$ não podem ser unificados.

Nota: = faz a unificação, mas não a avaliação numérica. Is avalia numericamente a expressão do lado direita e unifica com o lado esquerdo.

o Comparação de Termos

- == (predicado identidade)

$\langle t_1 \rangle == \langle t_2 \rangle$ tem sucesso se $\langle t_1 \rangle$ e $\langle t_2 \rangle$ são identicos;

$\backslash==$ $\langle t_1 \rangle \backslash== \langle t_2 \rangle$ tem sucesso se $\langle t_1 \rangle$ e $\langle t_2 \rangle$ não são identicos.

$a == a$

$x == y$

$a == 'a'$

$x == a, x == a$

$a == b$

$x == a, x == a$

True

False

o Listas

[elem1, ..., elemn]

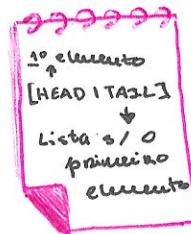
[] → lista vazia

[(1,2), (3,4)]

[[0,1,-,2], [1,-,-,-], [0,1,0,1]]

exemplo:

? - [HEAD | TAIL] = [dp, th, fd]



HEAD = dp,
TAIL = [th, fd]

- [dp, th, fd] = [dp | T]
T = [th, fd]

Soma

soma([H | T], N) :- soma(T, M), N is M + H.

soma([], N)

soma([4, 2], m), N is m + 1

Junta

junta(L1, L2, L3) é verdade se L3 for o resultado de juntar as listas L1 e L2.

junta([P1 | R], L1, [P1 | L2]) :- junta(R, L1, L2).

Recurse

comp2(L, O, C).
 ① Engordar → método iterativo

comp2([], C, C).
 ② Inicializado

comp2([], C, C).
 ③ Transferir

comp2([-1 cauda], A, C) :- A1 is A + 1, comp2(cauda, A1, C).

④ Implementação

$$\text{factorial } n! = \begin{cases} 1 & \text{se } n = 1 \\ n(n-1)! & \text{se } n > 1 \end{cases}$$

|| Recursiva ||

em prolog:

factorial(1, 1)	factorial(2, 2)
factorial(N, F) :- N > 1, N1 is N - 1, factorial(N1, F1), F is N * F1.	



Negação / conte

* falhando forçado: fail

Negação: ! + (p) :- p, !, fail

! + (p).

ou not.

Corte

R: - p1, ..., pm, !, q1, ..., qm.

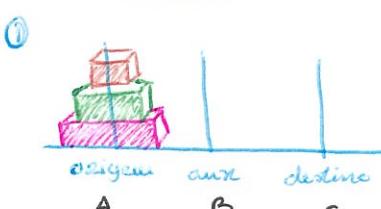
Quando o corte é atingido todas as alternativas possíveis e ainda não exploradas para R p1, ..., pm, não são considerados independentemente do que acontecer a q1, ..., qm.

Funções Importantes Prolog

findall(T, G, Bag) : Bag é a lista com as instâncias de T que verificam G.

member(X, [List]) : True se X é um elemento de List

Torres de Hanoi

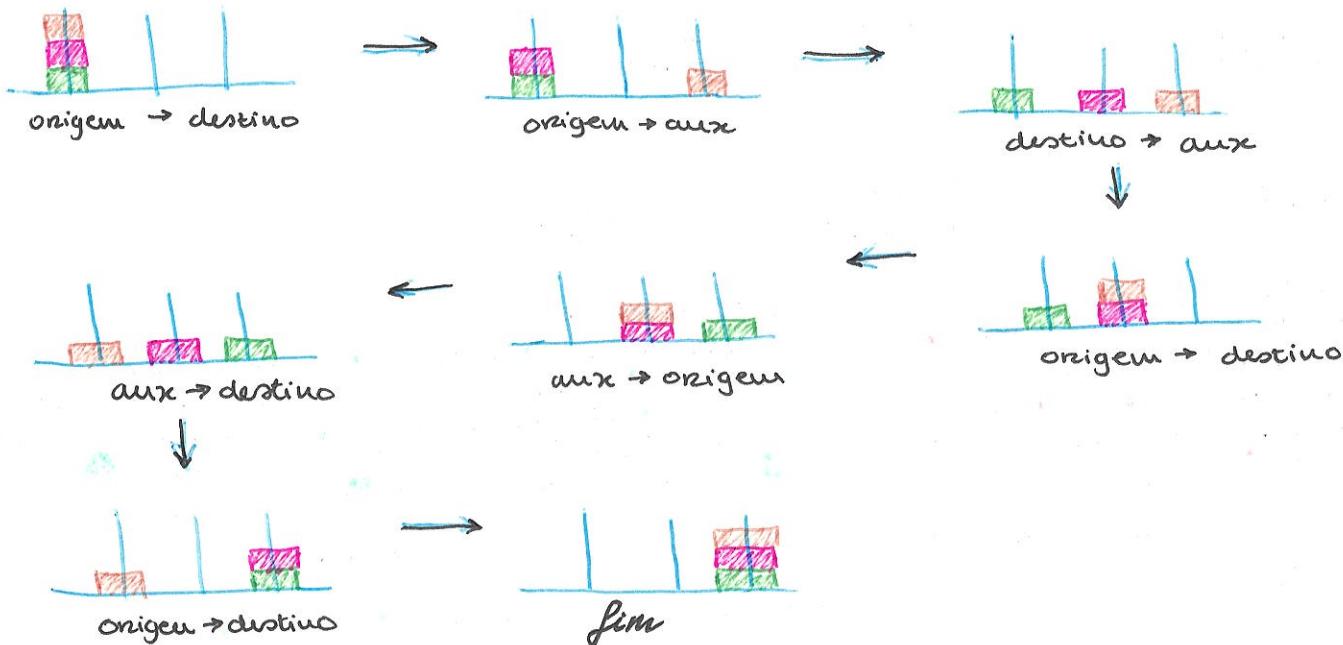


Intuição: mover N discos de A para C é o mesmo que:

- move os N-1 primeiros discos de A para B.
- o último mover para C.
- move os N-1 discos de B para C.

hanoi :- write('Número de discos:'), read(N), number(N), move(N, origem, destino, aux).
move(1, origem, destino, _) :- write(origem), write(' ->'), write(destino), nl.

`move (N, Origem, Destino, aux) :- N > 1, M is N-1,
 move (M, Origem, Aux, Destino),
 move (1, Origem, Destino, _),
 move (M, Aux, Destino, Origem).`



• Funções Progeto

`maplist(G, L)`: Verdade se G pode ser aplicado a todos os elementos L;
`include(G, L1, L2)`: filtra elementos que verificam G de L1 e coloca-os em L2.
`exclude(G, L1, L2)`: filtra elementos que não verificam G e coloca-os em L2.

• Estruturas

Exemplo:

```
data(ano, mes, dia).
ano-de(data(A, _, _), A).
mes-de(data(_, M, _), M).
dia-de(data(_, _, D), D).
muda_ano(A, data(_, M, D), data(A, M, D)).
```

Homologicidade: permite que um programa se modifique a si próprio;

Predicado estático: não podem ser alterados (default);

Predicados dinâmicos: podem ser alterados;

declarar:

`:- dynamic pred / aridade`

exemplo:

`:- dynamic mais alto / 2`

`:- dynamic mais baixo / 2`

?- asserta (maisAlto(h, t)).

?- asserta (maisAlto(h, ca)).

?- assertz (mais Baixo(x, y) :- mais Alto(y, x)).

Nota:

`asserta (c)` → adiciona c no inicio

`assertz (c)` → " " c no final

`retract(c)`: remove a 1ª cláusula que verifica c.

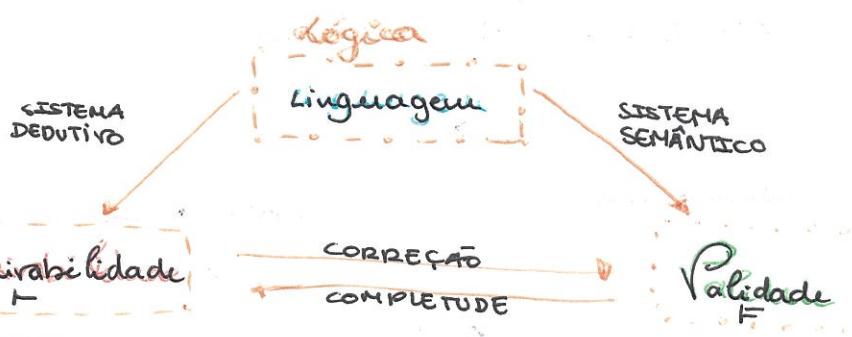
`retractall(pred)`: remove todas as cláusulas cuja cabeça é pred.

• Componentes de uma Lógica

Linguagem: A linguagem da sua lógica é definida com base num conjunto de regras de formação, que especificam frases legais da lógica (fbf).

Argumentos: (Δ, α), $\Delta \subseteq \delta$ (conjunto ou de fbf)

Um argumento é válido se e só se é impossível negar todas as premissas verdadeiras e a conclusão falsa.



SISTEMA DEDUTIVO:

→ manipulação de símbolos; provas (sequências de fbf); com base nas premissas tentarmos chegar à conclusão. Se $\Delta \vdash \alpha$, o argumento é demonstrável.

- Uma lógica diz-se completa, se todo o argumento válido é demonstrável.
- Uma lógica diz-se correta, se todo o argumento demonstrável é válido.
- O sistema semântico: Lógica Proposicional

[DEF] função de valoração

$$v: \mathcal{P} \rightarrow \{V, F\}$$

símbolos de proposição

[DEF] interpretação

$$I: \mathcal{L} \rightarrow \{V, F\}$$

I define-se recursivamente do seguinte modo:

- $I(\alpha) = V \text{ se } \alpha \in P$
- $I(\neg \alpha) = V \text{ se } I(\alpha) = F$
- $I(\alpha \wedge \beta) = V \text{ se } I(\alpha) = V \text{ e } I(\beta) = V$
- $I(\alpha \vee \beta) = V \text{ se } I(\alpha) = V \text{ ou } I(\beta) = V$
- $I(\alpha \rightarrow \beta) = V \text{ se } I(\alpha) = F \text{ ou } I(\beta) = V$

[DEF] satisfação

Diz-se que I satisfaz α se $I(\alpha) = V$

- α é satisfazível se e só se existe pelo menos uma interpretação I tal que $I(\alpha) = V$.
- α é falsificável se e só se existe pelo menos uma interpretação I tal que $I(\alpha) = F$.
- α é uma tautologia se e só se $I(\alpha) = V$ para todas as interpretações.
- α é uma contradição se e só se $I(\alpha) = F$ para todas as interpretações.

Conjunto satisfazível: existe uma interpretação que satisfaz todas as suas fórmulas;

Conjunto contraditório: se e só se não existe nenhuma interpretação que satisfaz todas as suas fórmulas; ex: $\{P, \neg P\}$

Modelo de um Conjunto de fórmulas: dado um conjunto de fbf Δ , uma interpretação que satisfaz todas as fbf de Δ diz-se um modelo de Δ .

$\models (\Delta, \alpha)$ é válido se e só se todos os modelos de Δ são modelos de α .

Teorema da Refutação

Δ conjunto de fbf

α fbf

$\Delta \vdash \alpha$ se e só se $\Delta \cup \{\neg \alpha\}$ não é satisfazível.

↓
Validade

SISTEMA SEMÂNTICO:

→ Atribuímos significado (valores lógicos) aos símbolos. Através destas atribuições determinamos a validade/invalidade de um argumento. Se $A \vdash \alpha$, α é consequência lógica de A , ou é implicação lógicamente α .

exemplo: $\alpha = (P \wedge Q) \rightarrow R$

P	Q	R	$P \wedge Q$	$(P \wedge Q) \rightarrow R$
V	V	V	V	V
V	V	F	V	F
V	F	V	F	V
V	F	F	F	V
F	V	V	F	V
F	V	F	F	V
F	F	V	F	V
F	F	F	F	F

$I(\alpha) = F$

Árvores de Decisão

Árvores binárias em que:

1. nós são símbolos proposicionais

2. folhas são **V** ou **F**

3. ramos — verdadeiro
--- falso

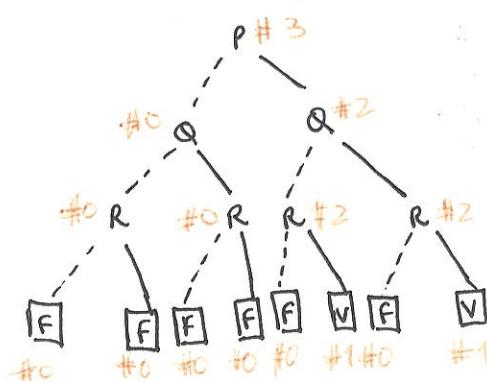
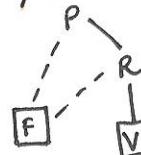


tabla associativa:

F #0	V #1	P #3	R #2
#0	#1	#0	#2

filhos iguais → mesma marca

compacta:



ideia: transformar árvore em grafo acíclico e dirigido.

OBDOS Grafo dirigido acíclico e rotulado que satisfaça a relação de ordem total parcial entre seus símbolos;

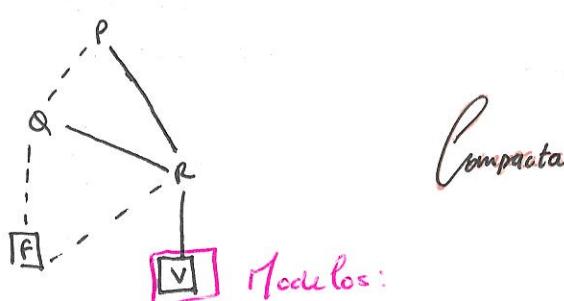
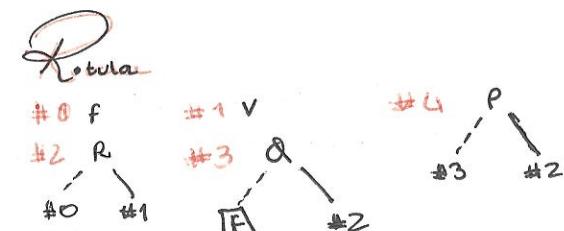
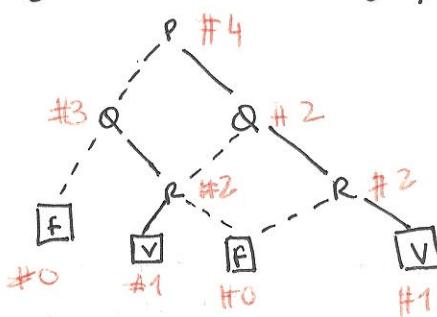
{ P_1, \dots, P_n } símbolos de proposições

≤ Relação de ordem

total → permite comparar todos os elementos do conjunto.

exemplo:

Algoritmos Rotula & Compacta



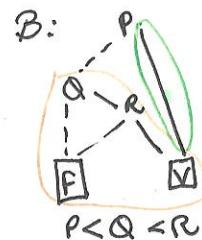
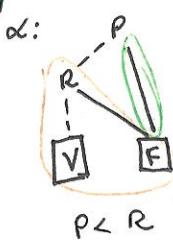
Modelos:

$$I(Q) = V \quad I(P) = V$$

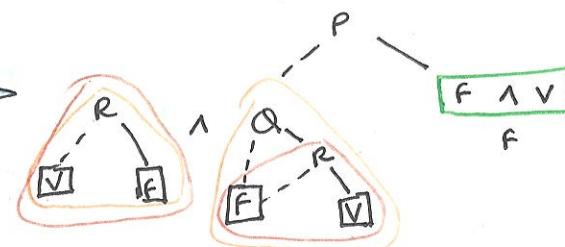
$$I(R) = V, I(P) = F$$

$$I(R) = V, I(Q) = V, I(P) = F$$

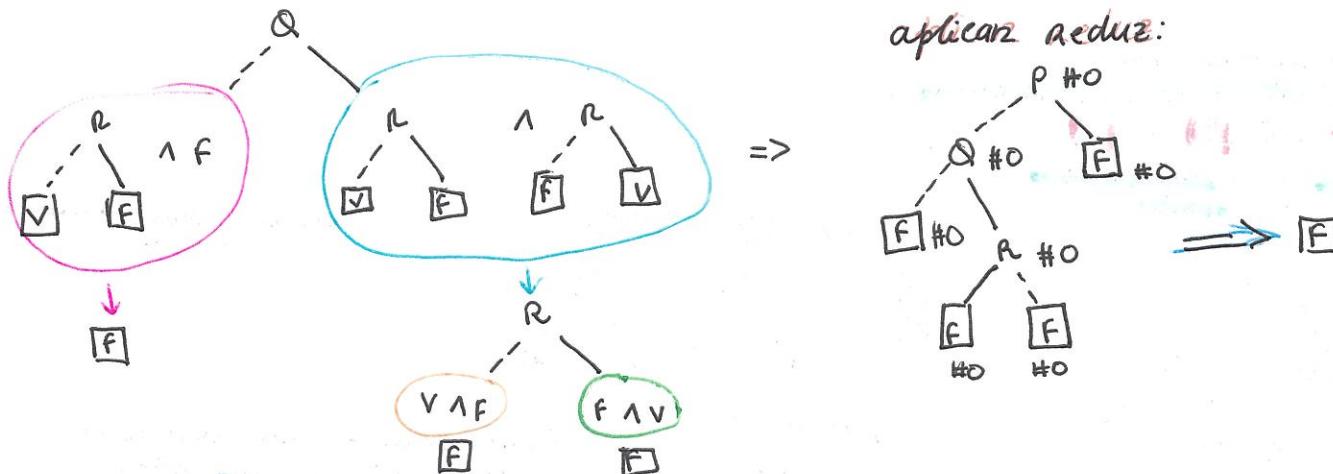
Aplica:



aplica (*A, B, 1*) → OPERAÇÃO



↓ enolhe-se o Q porque $Q < R$



Representação Lógica de OBDD's

Possibilita a realização de vários testes:

- ausência de símbolos redundantes;
- teste de validade;
- teste de satisfação;
- teste de implicação ($\alpha \rightarrow \beta$ se o OBDD resultante de $\alpha \wedge \neg \beta$ é \perp).

Algoritmo de SAT (Satisfiability)

Permite apenas saber se uma fbf é satisfazível ou não (se sim, identificar uma testemunha (módulo)).

OBDD's permitem identificar todos os módulos de uma fbf (interpretações que satisfazem a fbf);

Algoritmos:

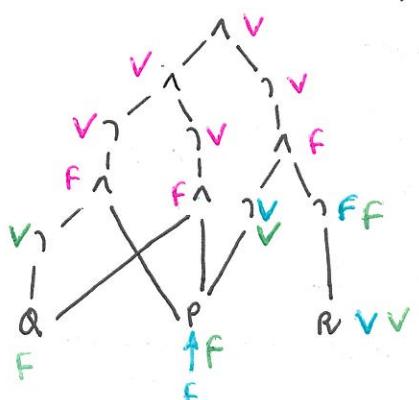
1. Propagação de marcas (eficiente, mas não completo).
2. DP (menos eficiente, mas completo).

Algoritmo de Propagação de Marcas

1. Obter uma fbf que só contém conjunções e negações.
 $\neg \vee B \equiv \neg(\neg \alpha \wedge \neg \beta)$
 $\alpha \rightarrow \beta \equiv \neg \alpha \vee \beta$
 $\neg \neg \alpha \equiv \alpha$
 2. Construir um grafo dirigido acíclico (DAG)
 3. Marcar o topo do DAG com um V
 4. Propagar marcas tendo em conta as regras para a \wedge e \neg .
- Exemplo:**
- $$\begin{aligned} & \neg(P \rightarrow \neg(\neg P \vee Q)) \\ \Rightarrow & \neg(\neg P \vee (\neg \neg P \wedge \neg Q)) \equiv \neg P \wedge (\neg P \wedge \neg Q) \\ \equiv & P \wedge (\neg P \wedge Q) \end{aligned}$$
-

Testemunha (módulo): $I(P) = V, I(Q) = V$. A fbf é satisfazível.

Exemplo: $(P \rightarrow Q) \wedge (P \rightarrow \neg Q) \wedge (P \vee R)$
 $(\neg(P \wedge Q)) \wedge (\neg(P \wedge \neg Q)) \wedge (\neg(\neg P \wedge R))$



Não teste - Paralelo

Indeterminado

Marcas permanentes
teste de nós:

Marcas temporárias que podem passar a permanentes
(se pode ser chamado novamente se tudo tiver carácter permanente)

teste de um nó:

Hip. 1: atribuição de marcas consistentes a todos os nós.
Hip. 2: propagação do valor atribuído (V/F) da origem a uma contradição, logo é atribuído o valor círculo.

(F/V) a esse no com carácter permanente;

Hip. 3: nenhuma das hipóteses anteriores... mas: marcas temporárias comuns aos dois valores possam ser permanentes.

exemplo: $I(P)=V$ $\xrightarrow{\text{Prop.}}$ $:V$ $\quad \quad \quad \text{N}:$
 $I(P)=F$ $\xrightarrow{\text{Prop.}}$ $:V$ } passa a permanente

Algoritmo DP

Idéia: seja Δ uma Fbf e P um símbolo de proposição, Δ satisfazível se o conjunto obtido por eliminação de P for satisfazível.

$\exists P(\Delta) \rightarrow$ eliminação de P em Δ - como se calcula?

Todos os Resolventes - P são adicionados

Todas as cláusulas que contém P são retiradas

exemplo:

$$\Delta = \{\{P, Q\}, \{\neg P, Q\}, \{\neg P, S\}, \{\neg Q, R\}\}$$

$$\exists P(\Delta) = \{\{Q\}, \{Q, S\}, \{\neg Q, R\}\}$$

Aux:

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



$$\{Q\}$$

$$\{P, Q\} \setminus \{\neg P, S\}$$



$$\{Q, S\}$$

teorema: Seja Δ uma Fbf na forma cláusal e P_i um símbolo de proposição. Então Δ é satisfazível se e só se $\exists P_i(\Delta)$ é satisfazível.

Algoritmo DP:

1. Eliminar sucessivamente os símbolos de proposição de Δ .
2. Condição de paragem:

2.1. Geração de cláusula vazia ($\{\}$ ∈ Δ): Fbf inicial não é satisfazível.

2.2. Obtenção de um conjunto vazio ($\{\} = \Delta$): Fbf inicial é satisfazível.

exemplo 1

$$\Delta = \{\{P, Q\}, \{\neg P, Q\}, \{\neg Q, R\}, \{\neg R\}\}$$

$$\exists P(\Delta) = \{\{Q\}, \{\neg Q, R\}, \{\neg R\}\}$$

$$\exists Q(\exists P(\Delta)) = \{\{\neg R\}, \{\neg R\}\}$$

$$\exists R(\exists Q(\exists P(\Delta))) = \{\} \rightarrow \Delta$$

Δ é satisfazível

exemplo 2

$$\Delta = \{\{P, Q\}, \{\neg P, Q\}, \{\neg Q, R\}\}$$

$$\exists P(\Delta) = \{\{Q\}, \{\neg Q, R\}\}$$

$$\exists Q(\exists P(\Delta)) = \{\{\neg R\}\}$$

$$\exists R(\exists Q(\exists P(\Delta))) = \{\} \rightarrow \text{satisfazível}$$

implementação usando balde:

- Ordem nos símbolos de proposição;
- um balde para cada símbolo;
- cada cláusula vai parar ao 1º balde que menciona 1 dos seus símbolos.
- Prossseguir cada balde:
 - o em cada balde bp: gerar os resolventes de P_i e colocar no balde respetivo.
 - o o algoritmo termina com a geração de $\{\}$ ou após não se conseguir fazer nada. (não satisfazível) (satisfazível)

exemplo: $\Delta = \{\{P, Q, \neg R\}, \{\neg P, S, T, R\}, \{\neg Q, \neg R\}, \{S\}, \{\neg P, Q, S\}\}$ orden: $P < Q < R < S < T$

$b_P: \{P, Q, \neg R\}, \{\neg P, S, T, R\}, \{\neg P, Q, S\} \rightarrow C.\text{aux}: \{P, Q, \neg R\} \setminus \{P, S, T, R\} \quad \{P, Q, \neg R\} \setminus \{P, Q, S\}$

$b_Q: \{\neg Q, \neg R\}, \{\neg Q, \neg R, S\} \rightarrow C.\text{aux}: \{\neg Q, \neg R\} \setminus \{\neg Q, \neg R, S\} \quad \{\neg Q, \neg R\} \setminus \{\neg Q, \neg R, S\}$

$b_R: \{\neg R, S\} \text{ FIM} - (\neg RVS)$

$b_S: \{S\}$

$b_T:$

- testemunha:
I(R)=V (opcional)
I(Q)=F (obrigat.)
I(S)=V (obrigatório)
I(P)=V (obrigatório)

sai pois é um teorema

