Lógica Computacional

Departamento de Ciência de Computadores
Faculdade de Ciências, Universidade do Porto
email: nelma.moreira@fc.up.pt

Versão: 2022

Estas notas baseam-se parcialmente nos *Apontamentos de Lógica Computacional* de Sabine Broda [Bro00].

Conteúdo

Capítulo 1

Lógica proposicional

A lógica proposicional remonta a Aristóteles, e teve como objectivo modelizar o raciocínio humano. Partindo de frases declarativas (*proposições*), que podem ser *verdadeiras* (**V**) ou *falsas* (**F**) estuda-se o processo de construção e a veracidade de outras proposições usando conectivas como *ou* ($\vee$), *e* ($\wedge$), *não* ($\neg$), *se... então...* ($\rightarrow$). Se p e q são proposições, $p \wedge q$ é uma proposição *verdadeira* se p e q o forem, e é uma proposição *falsa*, caso contrário; $p \vee q$ é uma proposição *verdadeira* se p ou q o forem, e *falsa*, caso contrário;$\neg p$ é uma proposição verdadeira se p for *falsa*, e *falsa* se p for *verdadeira*.
Considera as seguintes frases declarativas (com o respectivo valor de verdade):

- *Os gorilas são mamíferos* **V**
- *O Porto é uma cidade* **V**
- $2 + 3 = 6$ **F**
- $3 + 3 = 6$ **V**
- $3 \in \mathbb{N}$ **V**
- $3 \geq 7$ **F**
- *Um quadrado tem 6 lados* **F**

Então, podemos concluir que:

- *Os gorilas são mamíferos* e *O Porto é uma cidade* **V**

  porque é uma conjunção de proposições **V**
- $2 + 3 = 6$ ou $3 \in \mathbb{N}$ **V**

  porque é uma disjunção de proposições das quais uma é **V**

- não $3 \geq 7$ **V**

  porque é uma negação de uma proposição **F**

- Se $7 > 3$ então $3 + 3 = 6$ **V**

  porque uma implicação é **V** sse o consequente é **V** sempre que o antecedente é **V**

- Se $3 \geq 7$ então $2 + 3 = 6$ **V**

  porque é uma implicação cujo antecedente é **F**.

Cada proposição vai ser representada por uma *variável proposicional* (p, q, s, t, p_1, ...) e as conectivas lógicas por *símbolos n-ários*. Em particular temos:

Conectiva	Símbolos	Aridade	Outros símbolos equivalentes
Conjunção	$\wedge$	2	&, &&, $\cdot$
Disjunção	$\vee$	2	|, +
Negação	$\neg$	1	$\sim$, $^-$, !
Implicação	$\rightarrow$	2	$\Rightarrow$, $\supset$

1.1 Linguagens da lógica proposicional

Uma *linguagem da lógica proposicional* é formada a partir dos seguintes conjuntos de símbolos primitivos:

- um conjunto numerável de variáveis proposicionais

  $\mathcal{V}_{Prop} = \{p, q, r, \ldots, p_1, \ldots\}$

- conectivas lógicas $\wedge$, $\vee$, $\neg$ e $\rightarrow$

- os parêntesis (e)

Definição 1.1. *Uma fórmula* bem-formada *(φ,ψ,θ,...) é definida indutivamente pelas seguintes regras:*

i) uma variável proposicional p é uma fórmula

ii) se φ é uma fórmula então $(\neg\varphi)$ é uma fórmula

iii) se φ e ψ são fórmulas então $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ e $(\varphi \rightarrow \psi)$ são fórmulas

Exemplos de fórmulas são:

- $((p \wedge (\neg p)) \rightarrow \neg(p \wedge (q \vee r)))$

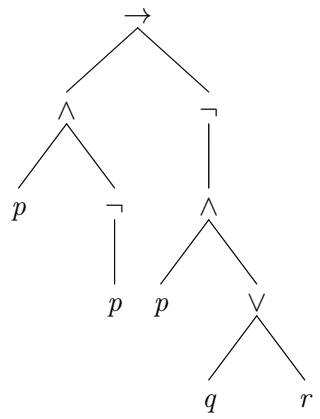
- $(((\neg p) \wedge q) \rightarrow (p \wedge (q \vee (\neg r))))$

O conjunto das fórmulas da lógica proposicional pode também ser descrito pela seguinte gramática independente de contexto, em notação BNF:

$$\varphi := p \mid (\neg\varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$$

uma gramática independente de contexto que gere a linguagem das fórmulas da lógica proporcional, onde $p \in \mathcal{V}_{Prop}$. Usando a gramática podemos associar a cada fórmula uma árvore sintáctica.

Exemplo 1.1. *Omitindo os parêntesis, temos a seguinte árvore pra a fórmula* $((p \wedge (\neg p)) \rightarrow \neg(p \wedge (q \vee r)))$*:*



Para legibilidade das fórmulas, os parêntesis podem-se omitir, considerando as seguintes regras:

- os parêntesis exteriores são omitidos
- $\neg$ tem precedência sobre $\wedge$
- $\wedge$ tem precedência sobre $\vee$
- $\vee$ tem precedência sobre $\rightarrow$
- $\wedge$ e $\vee$ são associativas à esquerda
- $\rightarrow$ é associativa à direita

Por exemplo, $\varphi \wedge \psi \vee \theta$ é uma abreviatura de $((\varphi \wedge \psi) \vee \theta)$ e $\psi \wedge \varphi \wedge \theta$ corresponde a $((\psi \wedge \varphi) \wedge \theta)$

Exemplo 1.2. *Fazendo a correspondência entre variáveis lógicas e as seguintes frases declarativas:*

p**:** Os gorilas são mamíferos

q**:** O Porto é uma cidade

r**:** $2 + 3 = 6$

s**:** $3 + 3 = 6$

t**:** $3 \in \mathbb{N}$

u**:** $3 \geq 7$

v**:** Um quadrado tem 6 lados

as frases consideradas acima correspondem às seguintes fórmulas:

1. $p \wedge q$
2. $r \vee t$
3. $\neg t$
4. $\neg u \rightarrow s$
5. $u \rightarrow r$

Definição 1.2. *Uma* sub-fórmula imediata *é definida indutivamente pelas seguintes regras:*

1. *uma variável proposicional não tem sub-fórmulas imediatas;*
2. $(\neg\varphi)$ *tem* φ *como sub-fórmula imediata;*
3. *as fórmulas* $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ *e* $(\varphi \rightarrow \psi)$ *têm* φ *e* ψ *como sub-fórmulas imediatas.*

Definição 1.3. *Uma fórmula* φ *é uma* sub-fórmula *duma fórmula* ψ *se e só se:*

1. φ *é uma sub-fórmula imediata de* ψ
2. *existe uma fórmula* θ *tal que* φ *é uma sub-fórmula de* θ *e* θ *é uma sub-fórmula de* ψ

Dada uma árvore sintáctica duma fórmula, cada nó define uma sua sub fórmula.

Exemplo 1.3. *Para o exemplo 1.1 as sub-fórmulas são:*$p, q, \neg p, r, p \wedge \neg p, q \vee r, p \wedge (q \vee r) e \neg(p \wedge (q \vee r))$.

Exercício 1.1. *Constrói a árvore sintáctica (omitindo parêntisis) e determina quais as sub-fórmulas e quais as sub-fórmulas imediatas de:*
$((\neg p \wedge q) \rightarrow (p \wedge (q \vee \neg r)))$
$\diamond$

1.2 Semântica da lógica proposicional

Os valores de verdade são **V** e **F**, onde **V** representa o valor lógico *verdadeiro* e **F**, *falso*.

Definição 1.4. *Uma* atribuição de valores de verdade *(ou valoração) é uma função* $v : \mathcal{V}_{Prop} \longrightarrow \{\boldsymbol{V}, \boldsymbol{F}\}$ *que atribuí um valor de verdade a cada variável proposicional. Uma valoração* v *pode ser estendida ao conjunto das fórmulas:*

i. *para* $p \in \mathcal{V}_{Prop}$ $v(p)$ *já está definido*

ii.

$$v(\neg\varphi) = \boldsymbol{V} \quad se \quad v(\varphi) = \boldsymbol{F}$$
$$v(\neg\varphi) = \boldsymbol{F} \quad se \quad v(\varphi) = \boldsymbol{V}$$

iii.

$$v(\varphi \wedge \psi) = \boldsymbol{V} \quad se \quad v(\varphi) = \boldsymbol{V} \text{ e } v(\psi) = \boldsymbol{V}$$
$$v(\varphi \wedge \psi) = \boldsymbol{F} \quad caso\ contrário$$

iv.

$$v(\varphi \vee \psi) = \boldsymbol{V} \quad se \quad v(\varphi) = \boldsymbol{V} \text{ ou } v(\psi) = \boldsymbol{V}$$
$$v(\varphi \vee \psi) = \boldsymbol{F} \quad caso\ contrário$$

v.

$$v(\varphi \to \psi) = \boldsymbol{F} \quad se \quad v(\varphi) = \boldsymbol{V} \text{ e } v(\psi) = \boldsymbol{F}$$
$$v(\varphi \to \psi) = \boldsymbol{V} \quad caso\ contrário$$

Podemos resumir usando as seguintes *tabelas de verdade*:

φ	$\neg\varphi$
F	**V**
V	**F**

φ	ψ	$\varphi \wedge \psi$
F	**F**	**F**
F	**V**	**F**
V	**F**	**F**
V	**V**	**V**

φ	ψ	$\varphi \vee \psi$
F	**F**	**F**
F	**V**	**V**
V	**F**	**V**
V	**V**	**V**

φ	ψ	$\varphi \to \psi$
F	**F**	**V**
F	**V**	**V**
V	**F**	**F**
V	**V**	**V**

Dada uma fórmula φ, o valor de $v(\varphi)$ pode assim ser recursivamente calculdado a partir dos valores atribuídos às variáveis da fórmula φ.
Por exemplo se $v(p) = \mathbf{V}$, $v(q) = \mathbf{F}$ e $v(r) = \mathbf{V}$, podemos calcular o valor de $v((p \wedge q) \vee \neg r)$:

p	q	r	(	p	∧	q	)	∨	¬	r
V	**F**	**V**			**F**			**F**	**F**	

Uma fórmula φ com n variáveis proposicionais tem 2^n valorizações. Porquê?
Podemos construir uma tabela em que cada linha corresponde a uma delas. Seja φ a fórmula $(p \wedge q) \vee \neg r$, temos as seguintes valorizações:

p	q	r	(	p	∧	q	)	∨	¬	r
V	**V**	**V**			**V**			**V**	**F**	
V	**V**	**F**			**V**			**V**	**V**	
V	**F**	**V**			**F**			**F**	**F**	
V	**F**	**F**			**F**			**V**	**V**	
F	**V**	**V**			**F**			**F**	**F**	
F	**V**	**F**			**F**			**V**	**V**	
F	**F**	**V**			**F**			**F**	**F**	
F	**F**	**F**			**F**			**V**	**V**	

1.2.1 Satisfazibilidade, validade e consequência

Definição 1.5. *Uma fórmula φ é*

satisfazível *se existe uma valoração v tal que $v(\varphi) =$ **V**. Escreve-se $\models_v \varphi$ (ou $v \models \varphi$) e diz-se que v satisfaz φ*

uma tautologia *se para todas as valorizações v, $v(\varphi) =$ **V** e escreve-se $\models \varphi$. Também se diz que φ é* válida. *Ex: $\models p \vee \neg p$ (Terceiro excluído)*

uma contradição *se para todas as valorizações v, $v(\varphi) =$ **F**. Escreve-se $\not\models \varphi$. Ex: $\not\models p \wedge \neg p$.*

Em alternativa, podemos definir a relação $\models_v$ de satisfazibilidade entre uma formula φ e uma valoração v directamente.

Definição 1.6. *Seja v uma valoração, a relação $\models_v$ é definida indutivamente na estrutura de φ por:*

1. $\models_v p$ *se* $v(p) =$ **V**;
2. $\models_v \neg\varphi$ *se* $\not\models_v \varphi$;
3. $\models_v \varphi \wedge \psi$ *se* $\models_v \varphi$ *e* $\models_v \psi$;
4. $\models_v \varphi \vee \psi$ *se* $\models_v \varphi$ *ou* $\models_v \psi$;

5. $\models_v \varphi \to \psi$ *se* $\not\models_v \varphi$ *ou* $\models_v \psi$;

Dizemos que v satisfaz φ *se* $\models_v \varphi$.

A definição de $\models$ e $\not\models$ mantêm-se assim como a noção de tautologia e contradição.

Exemplo 1.4. *Sendo* $v(p) =$ **V**, $v(q) = v(r) =$ **F** *determinar se*

$$\models_v (p \to (q \vee r)) \vee (r \to \neg p).$$

Para tal, $\models_v (p \to (q \vee r))$ *ou* $\models_v (r \to \neg p)$. *A segunda verifica-se porque* $\not\models_v r$. *Pelo que podemos também concluir que* $\models_v (p \to (q \vee r)) \vee (r \to \neg p)$.

A demonstração da proposição seguinte segue das definições e mostra que tanto podemos usar a notação $\models_v \varphi$ como $v(\varphi)$. A vantagem da primeira é que se pode generalizar para outras lógicas enquanto a segunda só é utilizável na lógica proposicional.

Proposição 1.1. *Dada uma fórmula* φ *e uma valoração* v, $\models_v \varphi$ *se* $v(\varphi) =$ **V** *e* $\not\models_v \varphi$ *se* $v(\varphi) =$ **F**.

Proposição 1.2. *Uma fórmula* φ *é uma tautologia se e só se* $\neg\varphi$ *é uma contradição. Uma fórmula* φ *é satisfazível se e só se* $\neg\varphi$ *não é uma tautologia. Uma fórmula é* não-satisfazível *se e só se é uma contradição.*

Demonstração. Seja v uma atribuição de valores de verdade. Por definição, $v(\varphi) =$ **V** ($\models_v \varphi$) se e só se $v(\neg\varphi) =$ **F** ($\not\models \neg\varphi$). Se φ é verdade para todas as valorizações ($\models \varphi$) então $\neg\varphi$ é falsa para todas elas ($\not\models \varphi$). Do mesmo modo se concluem a segunda e terceira afirmação. □

Definição 1.7. *Seja* Γ *um conjunto de fórmulas. Uma valoração* v satisfaz Γ *se e só se* v *satisfaz toda a fórmula* $\psi \in \Gamma$. *O conjunto* Γ *é* satisfazível *se existe uma valoração que o satisfaz. Uma fórmula* φ *é uma* consequência semântica *de* Γ, *se para toda a valoração* v *que satisfaz* Γ, *se tem* $v(\varphi) =$ **V**; *e escreve-se* $\Gamma \models \varphi$.

Se $\Gamma = \emptyset$, então $\emptyset \models \varphi$ é equivalente a $\models \varphi$. Nota que $\models \varphi$ se e só se φ é uma tautologia. Se $\Gamma = \{\psi\}$ e $\Gamma \models \varphi$ então diz-s que φ é *consequência semântica* de ψ (e escreve-se $\psi \models \varphi$). Em geral se $\Gamma \models \varphi$, podemos dizer que a veracidade de todas as fórmulas de Γ garante a veracidade de φ.

Exemplo 1.5. *Sendo* $\Gamma = \{p \vee q, (p \vee q) \to \neg r, q \to r\}$ *mostrar que* $\Gamma \models \neg q$. *Temos que analizar quais são as valorações que tornam verdadeiras todas as fórmulas de* Γ *e para todas elas garantir que* $\neg q$ *também é verdade. Uma valoração* v *que satisfaça as duas primeiras fórmulas tem de ter* $v(r) =$ **F**. *Isto porque,* $\models_v (p \vee q)$ *e* $\models_v (p \vee q) \to \neg r$ *obriga a que* $\models_v \neg r$ *(i.e.* $\not\models v$*). Então, para que* $\models_v q \to r$ *temos também que* $\not\models_v q$, *isto é, que* $v(\neg q) =$ **V**.

Todos estes conceitos podem ser avaliados usando tabelas de verdade (embora isso seja muito pouco eficiente). Uma fórmula φ é *satisfazível*, se houver uma linha da tabela para a qual o seu valor é **V**; uma *tautologia* se para todas as linhas o seu valor é **V**; uma *contradição* se para nenhuma linha o seu valor é **V**. A fórmula φ é *consequência semântica* de um conjunto finito de fórmulas $\{\psi_1, \ldots, \psi_n\}$ se para todas as linhas que todos os ψ_i são **V**, então φ também é **V**.

Se $\psi \models \varphi$ e $\varphi \models \psi$ então ψ e φ são *semânticamente equivalentes*(i.e as suas tabelas de verdade são iguais), e escreve-se $\psi \equiv \varphi$.

Alguns exemplos de fórmulas semânticamente equivalentes:

$\varphi \wedge \psi$	$\equiv$	$\psi \wedge \varphi$	(comutatividade do $\wedge$)
$\varphi \vee \psi$	$\equiv$	$\psi \vee \varphi$	(comutatividade do $\vee$)
$\neg(\varphi \wedge \psi)$	$\equiv$	$(\neg\varphi \vee \neg\psi)$	(Lei de DeMorgan)
$\neg(\varphi \vee \psi)$	$\equiv$	$(\neg\varphi \wedge \neg\psi)$	(Lei de DeMorgan)
$(\varphi \wedge \psi) \wedge \theta$	$\equiv$	$\varphi \wedge (\psi \wedge \theta)$	(associatividade)
$(\varphi \vee \psi) \vee \theta$	$\equiv$	$\varphi \vee (\psi \vee \theta)$	(associatividade)
$(\varphi \vee \varphi)$	$\equiv$	φ	(idempotência)
$(\varphi \wedge \varphi)$	$\equiv$	φ	(idempotência)
$(\varphi \wedge \psi) \vee \theta$	$\equiv$	$(\varphi \vee \theta) \wedge (\psi \vee \theta)$	(distributividade)
$(\varphi \vee \psi) \wedge \theta$	$\equiv$	$(\varphi \wedge \theta) \vee (\psi \wedge \theta)$	(distributividade)
$\neg\neg\varphi$	$\equiv$	φ	(Dupla negação)
$\varphi \to \psi$	$\equiv$	$\neg\varphi \vee \psi$	

Exercício 1.2. *Verifica as equivalências anteriores.* $\diamond$

Resolução 1.2

As duas primeiras são consequência imediata da definição. Para uma conjunção ser **V** têm de ambos os argumentos serem, independentemente da ordem. A disjunção só é **F** se ambos forem e portanto a ordem não interessa. Do mesmo modo se prova que são válidas as associatividades e as idempotências.

Considere-se a primeira Lei de DeMorgan e começamos por mostrar que $\neg(\varphi \wedge \psi) \models \neg\varphi \vee \neg\psi$. Suponhamos que para uma valoração v_1 se tem $v_1(\neg(\varphi \wedge \psi)) = \mathbf{V}$. Então $v_1(\varphi \wedge \psi) = \mathbf{F}$ o que significa que $v_1(\varphi) = \mathbf{F}$ ou $v_1(\psi) = \mathbf{F}$. Isto é exactamente dizer que $v_1(\neg\varphi) = \mathbf{V}$ ou $v_1(\neg\psi) = \mathbf{V}$, ou ainda $v_1(\neg\varphi \vee \neg\psi) = \mathbf{V}$. Então provámos que $\neg(\varphi \wedge \psi) \models \neg\varphi \vee \neg\psi$. A consequência oposta ($\neg\varphi \vee \neg\psi \models \neg(\varphi \wedge \psi)$) prova-se de maneira análoga.

O mesmo racíocinio se aplica para a segunda lei de DeMorgan e análogo para as distributividades.

Para a dupla negação basta notar que para uma valoração v_1, $v_1(\neg\neg\varphi) = \mathbf{V}$ se e só se $v_1(\neg\varphi) = \mathbf{F}$ se e só se $v_1(\varphi) = \mathbf{V}$.

Para a última equivalência seja $v_1(\varphi \to \psi) = \mathbf{V}$. Temos 2 casos a considerar. Se $v_1(\varphi) = \mathbf{V}$ então $v_1(\psi) = \mathbf{V}$. Portanto, $v_1(\neg\varphi \vee \psi) = \mathbf{V}$. Se $v_1(\varphi) = \mathbf{F}$, então $v_1(\neg\varphi) = \mathbf{V}$, e também $v_1(\neg\varphi \vee \psi) = \mathbf{V}$. Logo, $\varphi \to \psi \models \neg\varphi \vee \psi$. Para provar que $\neg\varphi \vee \psi \models \varphi \to \psi$, suponhamos que $v_1(\neg\varphi \vee \psi) = \mathbf{V}$. Se $v_1(\varphi) = \mathbf{V}$ então, necessariamente, $v_1(\psi) = \mathbf{V}$ e concluímos que $v_1(\varphi \to \psi) = \mathbf{V}$. Se $v_1(\varphi) = \mathbf{F}$ então, pela definição da semântica da implicação temos $v_1(\varphi \to \psi) = \mathbf{V}$. O que prova o que se queria.

Exercício 1.3. *Justifica a veracidade ou falsidade de cada uma das afirmações seguintes, onde Γ e Σ representam conjuntos de fórmulas e $\varphi, \psi, \theta, \gamma$ representam fórmulas da lógica proposicional:*

1. *$\varphi \models \psi \to \theta$ e $\gamma \models \psi$ se e só se $\varphi, \gamma \models \theta$;*
2. *Se Σ é satisfazível então existe uma fórmula φ tal que $\Sigma \not\models \varphi$.*
3. *Se $\Gamma \models \theta$ e $\Gamma \subseteq \Sigma$, então $\Sigma \models \theta$;*

$\diamond$

Resolução 1.3

1.

$\Rightarrow$: Queremos que $\varphi, \gamma \models \theta$, isto é qualquer valoração que satisfaz φ e γ também satisfaz θ. Suponhamos uma valoração v_1 tal que $v_1(\varphi) = v_1(\gamma) = \mathbf{V}$ ($\models_{v_1} \varphi$ e $\models_{v_1} \psi$) e temos que provar que $v_1(\theta) = \mathbf{V}$ ($\models_{v_1} \theta$). Mas pela primeira hipótese, $v_1(\psi \to \theta) = \mathbf{V}$ ($\models_{v_1} \psi \to \theta$) e pela segunda $v_1(\psi) = \mathbf{V}$ ($\models_{v_1} \psi$). Logo pela definição da valoração duma implicação, se o antecedente é $\mathbf{V}$ e a implicação também é $\mathbf{V}$ então o consequente tem de ser $\mathbf{V}$. Logo, concluímos que $v_1(\theta) = \mathbf{V}$ ($\models_{v_1} \theta$).

$\Leftarrow$: Esta afirmação é falsa. Considera as variáveis proposicionais p e q e γ a fórmula p, φ a fórmula $p \to q$, θ a fórmula q e ψ a fórmula $\neg p$. É verdade que $\varphi, \gamma \models \theta$ mas $\gamma \not\models \psi$.

2. Suponhamos, por contradição, que para toda a fórmula φ, se tem $\Sigma \models \varphi$. Como Σ é satisfazível, seja v_1 uma valoração tal que $v_1(\Sigma) = \mathbf{V}$. Então para uma qualquer fórmula φ temos que $v_1(\varphi) = \mathbf{V}$. Mas como por hipótese todas as fórmulas são consequência de Σ então também $v_1(\neg\varphi) = \mathbf{V}$. Mas isto é uma contradição. Logo a afirmação é verdadeira.

3.

A afirmação é verdadeira. Se $\Gamma \models \theta$ então toda a valoração que satisfaz Γ (i.e. todas as suas formulas) também satisfaz θ. Queremos mostrar que se uma valoração satisfaz Σ então também satisfaz θ. Seja v tal que $v(\Sigma) = \mathbf{V}$ isto é para toda a fórmula $\psi \in \Sigma$ se tem $v(\psi) = \mathbf{V}$. Mas como $\Gamma \subseteq \Sigma$ então para toda fórmula $\gamma \in \Gamma$ como $\gamma \in \Sigma$ vem que $v(\gamma) = \mathbf{V}$ e portanto $v(\Gamma) = \mathbf{V}$. Mas então $v(\theta) = \mathbf{V}$, porque $\Gamma \models \theta$. Logo concluímos que $\Sigma \models \theta$.

1.2.2 Funções de verdade

A tabela de verdade duma fórmula φ, com $n > 0$ variáveis proposicionais $p_1, \ldots p_n$, define uma função de verdade

$$F_\varphi : \{\mathbf{V}, \mathbf{F}\}^n \longrightarrow \{\mathbf{V}, \mathbf{F}\}$$

tal que $F_\varphi(x_1, \ldots, x_n) = v_X(\varphi)$, onde v_X é uma valoração tal que $v_X(p_i) = x_i$ para $i \in \{1 \ldots n\}$ e $x_i \in \{\mathbf{V}, \mathbf{F}\}$.
Qualquer função de $f : \{\mathbf{V}, \mathbf{F}\}^n \longrightarrow \{\mathbf{V}, \mathbf{F}\}$, com $n > 0$ diz-se uma *função de verdade* ou função booleana.

Exercício 1.4. *Existem 4 funções de verdade e aridade 1 e 16 funções de verdade de aridade 2. Constrói as tabelas de verdade correspondentes. E quantas funções existem de aridade n, para $n > 0$?* $\diamond$

Definição 1.8. *Um conjunto de conectivas C diz-se* completo *se para qualquer função de verdade f, existe uma fórmula φ com n variáveis proposicionais e contendo apenas conectivas de C, tal que $F_\varphi = f$.*

Proposição 1.3. *O conjunto de conectivas $\{\wedge, \vee, \neg\}$ é completo.*

Demonstração. Mostramos por indução sobre n.

Base. Para $n = 1$ existem 4 funções de verdade:

x_1	f_1
V	**F**
F	**F**

x_1	f_2
V	**F**
F	**V**

x_1	f_3
V	**V**
F	**F**

x_1	f_4
V	**V**
F	**V**

Sendo $\varphi_1 = p \wedge \neg p$, $\varphi_2 = \neg p$, $\varphi_3 = p$, $\varphi_4 = p \vee \neg p$, tem-se que $F_{\varphi_i} = f_i$ para $1 \leq i \leq 4$.

Indução. Supondo que a hipótese é válida para n, seja $f : \{\mathbf{V}, \mathbf{F}\}^{n+1} \longrightarrow \{\mathbf{V}, \mathbf{F}\}$. Construímos duas funções n-árias f_1 e f_2 tal que:

$f_1(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, \mathbf{V})$

$f_2(x_1, \ldots, x_n) = f(x_1, \ldots, x_n, \mathbf{F})$.

Por hipótese de indução existem φ_i, com variáveis $p_1, \ldots, p_n$, tal que $F_{\varphi_i} = f_i$ para $i = 1, 2$. Tome-se $\varphi = (p_{n+1} \wedge \varphi_1) \vee (\neg p_{n+1} \wedge \varphi_2)$, então $F_\varphi = f$.

□

Proposição 1.4. *O conjunto de conectivas $\{\neg, \rightarrow\}$ é completo.*

Demonstração. Basta ver que $\varphi \wedge \psi \equiv \neg(\varphi \rightarrow \neg\psi)$ e $\varphi \vee \psi \equiv (\neg\varphi \rightarrow \psi)$. □

Exercício 1.5. *Mostra que o conjunto de conectivas $\{\neg, \vee\}$ é completo.* $\diamond$

1.2.3 Uso de conjuntos completos de conectivas

Podemos restringuir-nos só a conjuntos completos de conectivas. Em particular, podiamos ter considerado na definição da linguagem da lógica proposicional, apenas ou:

- as conectivas $\wedge$, $\vee$ e $\neg$
- as conectivas $\rightarrow$ e $\neg$
- ...

E considerar as restantes abreviaturas.
Uma das vantagens de ter um número menor de tipos de fórmulas é o de facilitar as demonstrações... Mas também podemos definir outras conectivas. Por exemplo uma para cada uma das funções de verdade unárias ou binárias... As mais usuais são:

Designação	Conectiva	Fórmula semanticamente equivalente
Falso	**F**	$\varphi \wedge \neg\varphi$
Verdade	**V**	$\varphi \vee \neg\varphi$
Implicação	$\varphi \rightarrow \psi$	$\neg\varphi \vee \psi$
Equivalência	$\varphi \leftrightarrow \psi$	$(\neg\varphi \vee \psi) \wedge (\varphi \vee \neg\psi)$
Ou Exclusivo	$\varphi \dot{\vee} \psi$	$(\varphi \wedge \neg\psi) \vee (\neg\varphi \wedge \psi)$
Não-e	$\varphi \tilde{\wedge} \psi$	$\neg(\varphi \wedge \psi)$
Não-ou	$\varphi \tilde{\vee} \psi$	$\neg(\varphi \vee \psi)$

Exercício 1.6. *Constrói as tabelas de verdade associadas a cada uma das conectivas.* ◇

Exercício 1.7. *Mostra que*

a) $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

b) $\varphi \rightarrow \psi \equiv \neg\psi \rightarrow \neg\varphi$ *(contrapositivo)*

c) $\models \varphi \rightarrow \psi$ *se e só se* $\varphi \models \psi$

d) $\models \varphi \leftrightarrow \psi$ *se e só se* $\varphi \equiv \psi$

◇

Leituras suplementares [BE00] (Cap. 1, 3, 7) [HR00] (Cap. 1.3, 1.4.1)

1.3 Formas normais

Vamos ver que podemos transformar fórmulas em fórmulas semânticamente equivalentes, de modo a obter fórmulas de *formas especiais* e que nos permitam decidir mais fácilmente sobre a satisfabilidade ou validade das fórmulas originais. Algumas dessas *formas normais* existem para qualquer fórmula, outras apenas para certas classes de fórmulas.

1.3.1 Forma normal negativa

Um *literal* é uma variável proposicional (p) ou a sua negação ($\neg p$). Uma fórmula diz-se em forma normal negativa se a ocorrência de $\neg$ for só em literais.

Proposição 1.5. *Qualquer fórmula contendo apenas as conectivas $\wedge$, $\vee$ e $\neg$ é semanticamente equivalente a uma fórmula em forma normal negativa.*

Demonstração. Basta usar as Leis de DeMorgan e eliminar as duplas negações. □

Exemplo 1.6. *Determinar a forma normal negativa de $\neg((p \vee q) \wedge \neg p)$.*

$$\begin{aligned} \neg((p \vee q) \wedge \neg p) & \\ \neg(p \vee q) \vee \neg\neg p & \quad \textit{(DeMorgan)} \\ (\neg p \wedge \neg q) \vee \neg\neg p & \quad \textit{(DeMorgan)} \\ (\neg p \wedge \neg q) \vee p & \quad \textit{(Dupla Negação)} \end{aligned}$$

Exercício 1.8. *Determina a forma normal negativa de*

$$(p \leftrightarrow q) \leftrightarrow r) \wedge \neg(p \leftrightarrow (q \leftrightarrow r))$$

◇

1.3.2 Forma normal disjuntiva

Uma fórmula diz-se em *forma normal disjuntiva* se for da forma:

$$(\alpha_{11} \wedge \ldots \wedge \alpha_{1k_1}) \vee \ldots \vee (\alpha_{n1} \wedge \ldots \wedge \alpha_{nk_n})$$

onde cada α_{ij} é um literal.

Lema 1.1. *Para qualquer função de verdade $f : \{\boldsymbol{V}, \boldsymbol{F}\}^n \longrightarrow \{\boldsymbol{V}, \boldsymbol{F}\}$, existe uma fórmula φ com n variáveis proposicionais e em forma normal disjuntiva, tal que $F_\varphi = f$.*

Demonstração. Se $f = \mathbf{F}$, para todos os valores dos argumentos, então $\varphi = p_1 \wedge \neg p_1$. Senão, para cada valoração v (correspondente a uma linha da tabela de verdade de f) seja:

$$\varphi_v = l_1^v \wedge \ldots \wedge l_n^v$$

onde

$$l_i^v = \begin{cases} p_i & \text{se } v(p_i) = \mathbf{V} \\ \neg p_i & \text{se } v(p_i) = \mathbf{F} \end{cases}$$

Nota que $v(\varphi_v) = \mathbf{V}$. Então, basta considerar

$$\varphi = \bigvee_{f(v(p_1),\ldots,v(p_n))=\mathbf{V}} \varphi_v$$

□

Para a seguinte função de verdade:

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
V	**V**	**V**	**V**
V	**V**	**F**	**V**
V	**F**	**V**	**V**
V	**F**	**F**	**F**
F	**V**	**V**	**V**
F	**V**	**F**	**F**
F	**F**	**V**	**F**
F	**F**	**F**	**F**

uma fórmula em forma normal disjuntiva é:

$$(p_1 \wedge p_2 \wedge p_3) \vee (p_1 \wedge p_2 \wedge \neg p_3) \vee (p_1 \wedge \neg p_2 \wedge p_3) \vee (\neg p_1 \wedge p_2 \wedge p_3)$$

Corolário 1.1. *Qualquer fórmula é semanticamente equivalente a uma fórmula em forma normal disjuntiva.*

Exercício 1.9. *Demonstra o Corolário 1.1.* ◇

Resolução 1.6.1. *Dada uma fórmula, é possível transformá-la numa semanticamente equivalente em forma normal disjuntiva, considerando os seguintes passos:*

1. *obter uma fórmula apenas com as conectivas $\wedge$, $\vee$ e $\neg$*
2. *obter uma fórmula em forma normal negativa*

3. *aplicar a distributividade:* $(\varphi \vee \psi) \wedge \theta \equiv (\varphi \wedge \theta) \vee (\psi \wedge \theta)$

   $\theta \wedge (\varphi \vee \psi) \equiv (\theta \wedge \varphi) \vee (\theta \wedge \psi)$

Exercício 1.10. *Determina uma forma normal disjuntiva para*

$$(p \vee r) \leftrightarrow (q \wedge \neg p)$$

$\diamond$

Resolução 1.6.2 (1.10)**.**

$$\begin{aligned}
(p \vee r) \leftrightarrow (q \wedge \neg p) &\Leftrightarrow ((p \vee r) \rightarrow (q \wedge \neg p)) \wedge ((q \wedge \neg p) \rightarrow (p \vee r)) \\
&\Leftrightarrow (\neg(p \vee r) \vee (q \wedge \neg p)) \wedge (\neg(q \wedge \neg p) \vee (p \vee r)) \\
&\Leftrightarrow ((\neg p \wedge \neg r) \vee (q \wedge \neg p)) \wedge ((\neg q \vee p) \vee p \vee r) \\
&\Leftrightarrow (((\neg p \wedge \neg r) \vee (q \wedge \neg p)) \wedge (\neg q \vee p)) \vee (((\neg p \wedge \neg r) \\
&\vee (q \wedge \neg p)) \wedge (p \vee r)) \\
&\Leftrightarrow (\neg p \wedge \neg r \wedge (\neg q \vee p)) \vee (q \wedge \neg p \wedge (\neg q \vee p)) \\
&\vee (\neg p \wedge \neg r \wedge (p \vee r)) \vee (q \wedge \neg p \wedge (p \vee r)) \\
&\Leftrightarrow (\neg p \wedge \neg r \wedge \neg q) \vee (\neg p \wedge \neg r \wedge p) \vee (q \wedge \neg p \wedge \neg q) \vee (q \wedge \neg p \wedge p) \\
&\vee (\neg p \wedge \neg r \wedge p) \vee (\neg p \wedge \neg r \wedge r) \vee (q \wedge \neg p \wedge p) \vee (q \wedge \neg p \wedge r) \\
&\Leftrightarrow (\neg p \wedge \neg r \wedge \neg q) \vee (q \wedge \neg p \wedge r)
\end{aligned}$$

No último passo eliminamos conjunções que não eram satisfazíveis.

Lema 1.2. *Uma conjunção de literais $l_1 \wedge \ldots \wedge l_n$ é satisfazível se e só se para todo o $1 \leq i, j \leq n$, l_i não é $\neg l_j$.*

Exercício 1.11. *Indica se são ou não saisfazíveis as seguintes fórmulas:*

1. $p \wedge \neg q \wedge \neg r \wedge q$

2. $\neg p \wedge q \wedge \neg r \wedge \neg s$

$\diamond$

Corolário 1.2. *Uma fórmula φ em forma normal disjuntiva é satisfazível se e só se alguma das suas conjunções de literais o for.*

Obtemos assim um método de determinar se uma fórmula é satisfazível. Em particular, se a fórmula já estiver em forma normal disjuntiva o método é linear no seu tamanho.

Exercício 1.12. *Explicita este método.* $\diamond$

Exercício 1.13. *Determina se a fórmula $(p \vee r) \leftrightarrow (q \wedge \neg p)$ é satisfazível* $\diamond$

1.3.3 Forma normal conjuntiva

Uma fórmula diz-se em *forma normal conjuntiva* se for da forma:

$$(\alpha_{11} \vee \ldots \vee \alpha_{1k_1}) \wedge \ldots \wedge (\alpha_{n1} \vee \ldots \vee \alpha_{nk_n})$$

onde cada α_{ij} é um literal.
Por dualidade, temos

Lema 1.3. *Uma disjunção de literais $l_1 \vee \ldots \vee l_n$ é uma tautologia se e só se para algum $1 \leq i, j \leq n$, l_i é $\neg l_j$.*

Então, é fácil determinar se uma fórmula em forma normal conjuntiva é uma tautologia: basta verificar se todas as disjunções são tautologias, pelo método dado no Lema 1.3.
Mas como obter uma fórmula em forma normal conjuntiva?

1. se tivermos a tabela de verdade, por um método dual ao da forma normal disjuntiva: isto é, escolher as linhas que correspondem a **F**, considerar para cada uma a disjunção de literais tal que se $x_i = \mathbf{V}$ coloca-se $\neg p_i$ e se $x_i = \mathbf{F}$ coloca-se p_i; e finalmente tomar a conjunção dessas disjunções (Verifica a correção!).

2. se tivermos uma fórmula, adaptar o método dado para a forma normal disjuntiva, usando a distributividade para a conjunção...

Exercício 1.14. *Obtém uma fórmula em forma normal conjuntiva correspondente à tabela de verdade dada anteriormente.* ◇

Resolução 1.6.3. *Uma fórmula é:*

$$(\neg p_1 \vee p_2 \vee p_3) \wedge (p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_2 \vee \neg p_3) \wedge (p_1 \vee p_2 \vee p_3)$$

Exercício 1.15. *Determina uma forma normal conjuntiva equivalente à fórmula*

$$(p \vee r) \leftrightarrow (q \wedge \neg p)$$

e determina se é ou não uma tautologia. ◇

1.3.4 Fórmulas de Horn e Satisfazibilidade

Uma *fórmula de Horn* da lógica proposicional é uma fórmula em forma normal conjuntiva em que em cada disjunção existe no máximo um literal positivo. Exemplos de fórmulas de Horn são:

$$p \wedge \neg q \wedge (q \vee \neg p)$$
$$(\neg p \vee \neg q \vee \neg s \vee p) \wedge (\neg q \vee \neg r \vee p) \wedge (\neg p \vee \neg s \vee s)$$
$$(\neg p \vee \neg q \vee \neg s) \wedge (\neg q \vee \neg r \vee p) \wedge s$$

Numa fórmula de Horn, as disjunções $\neg p_1 \vee \ldots \vee \neg p_n \vee p$ também se podem escrever como

$$(p_1 \wedge \ldots \wedge p_n) \rightarrow p$$

ou se p não existe (ou é **F**):

$$(p_1 \wedge \ldots \wedge p_n) \rightarrow \mathbf{F}$$

ou se os p_i não existem)

$$\mathbf{V} \rightarrow p$$

Nota que nem todas as fórmulas têm uma fórmula de Horn equivalente...basta que as suas formas normais conjuntivas tenham mais que um literal positivo que não possa ser simplificado...

Para determinar se uma fórmula de Horn da lógica proposicional é **satisfazível** podemos usar um algoritmo (mais eficiente que a construção da tabela de verdade correspondente e a verificação se para alguma linha a fórmula tem o valor **V**).

Vamos ilustrar o algoritmo com a fórmula $p \wedge \neg q \wedge (q \vee \neg p)$:

- começar por colocar numa linha as variáveis proposicionais que ocorrem na fórmula e colocar a fórmula. Ex:

p	q	$p \wedge \neg q \wedge (q \vee \neg p)$

- se alguma das variáveis proposicionais é um dos elementos da conjunção atribuir o valor **V** a essa variável (porquê?). Ex:

p	q	$p \wedge \neg q \wedge (q \vee \neg p)$
V		

- Com a informação dessas variáveis preencher a tabela como se tivesse a construir a tabela de verdade (para essa linha), analisando cada disjunção para determinar se, para ela ser verdadeira, se pode determinar mais valores para as variáveis proposicionais:

p	q	p	$\wedge$	$\neg$	q	$\wedge$	$(q$	$\vee$	$\neg$	$p)$
V										**F**

  Neste caso, q tem de ser **V** e então isso pode ser acrescentado:

p	q	p	$\wedge$	$\neg$	q	$\wedge$	$(q$	$\vee$	$\neg$	$p)$
V	**V**									**F**

E voltando a repetir este passo, obtém-se:

p	q	p	$\wedge$	$\neg$	q	$\wedge$	$(q$	$\vee$	$\neg$	$p)$
V	**V**			**F**					**F**	

Continuar até mais nada poder ser acrescentado.

- se no passo anterior se atribuir **F** a um dos elementos da conjunção, a fórmula também fica com o valor **F** e não é satisfazível. Caso contrário podemos atribuir à fórmula o valor **V** se atribuirmos **F** às restantes variáveis proposicionais. Nota que assim no máximo um literal por disjunção é **F** (e as disjunções não são unitárias).

  No exemplo que estamos a considerar, a fórmula tem o valor **F** e portanto não é satisfazível.

Exercício 1.16. *1. Justifica a correção do algoritmo, isto é, que atribui o valor verdade se e só se a fórmula de Horn é satisfazível.*

2. Aplica o algoritmo às seguintes fórmulas:

- $(\neg p \vee \neg q) \wedge (\neg q \vee r) \wedge q$
- $p \wedge (\neg p \vee q) \wedge (\neg q \vee p)$
- $\neg p \wedge (\neg p \vee q) \wedge \neg q$
- $p \wedge (\neg p \vee q) \wedge \neg r$

◇

1.3.5 Satisfazibilidade

O problema de determinar se uma fórmula da lógica proposicional é satisfazível[1] é um problema com muitas aplicações em Ciência de Computadores uma vez que muitos problemas de outras áreas se podem exprimir em termos dele:

- optimização: planeamento, escalonamento,etc.
- combinatória: coloração de grafos,etc.
- teorias de lógica de primeira ordem: programação linear, aritmética de reais, strings de bits, apontadores, etc. (resolutores SMT).
- resolução de puzzles, como o sudoku,etc.

[1] Normalmente designado por SAT.

Já vimos que este problema se pode resolver através da construção de tabelas de verdade, mas no pior caso isso pode ser exponencial no número de variáveis. Na realidade, não se conhece nenhum algoritmo mais eficiente (i.e., polinomial) para resolver este problema para qualquer tipo de fórmulas. A existência de tal algoritmo responderia a uma das questões mais importantes da Complexidade Computacional é: P=NP? Isto é, se a classe de problemas de decisão[2] que se podem resolver com algoritmos polinomiais (classe P) coincide com a classe de problemas para os quais se pode verificar em tempo polinomial se um candidato a solução é realmente uma solução (classe NP). No caso da satisfazilibidade de fórmulas proposicionais é fácil testar se uma dada valoração torna uma fórmula verdeira ou não.
Contudo, existem classes de fórmulas para as quais o problema é polinomial (linear): por exemplo se a fórmula estiver em forma normal disjuntiva ou for uma fórmula de Horn. Por outro lado, no caso geral existem diversos algoritmos que na prática se comportam muito melhor que o da construção de tabelas de verdade. Estes algoritmos podem ser aplicados a fórmulas genéricas (com qualquer tipo de conectiva) mas são especialmente simples se as fórmulas estiverem em forma normal conjuntiva (FNC). Este tipo de fórmulas pode ser representado de forma compacta usando a noção de *cláusula*.

1.3.5.1 Cláusulas

Definição 1.9. *Uma* cláusula *é uma disjunção de literais* $l_1 \vee l_2 \vee \ldots \vee l_n$, $n \geq 0$. *Uma cláusula pode-se representar por um conjunto de literais. Se* $n = 0$ *dizemos que a cláusula é* vazia *e corresponde a* $\boldsymbol{F}$. *Se* $n = 1$ *dizemos que a cláusula é* unitária.

Por exemplo $p \vee \neg q \vee \neg p \vee s$ é uma cláusula e pode representar-se por um conjunto $\{p, \neg q, \neg p, s\}$. Qualquer fórmula da lógica proposicional em FNC pode-se representar por um conjunto de cláusulas. Por exemplo,

$$\neg p \wedge (q \vee r \vee q) \wedge (\neg r \vee \neg s) \wedge (p \vee s) \wedge (\neg q \vee \neg s)$$

pode ser vista como um conjunto de cláusulas:

$$\{\{\neg p\}, \{q, r\}, \{\neg r, \neg s\}, \{p, s\}, \{\neg q, \neg s\}\}$$

E como qualquer fórmula é equivalente a uma em FNC, se tivermos um método para determinar a satisfazibilidade de cláusulas, temos um método que se pode aplicar a qualquer fórmula.

[2] Cuja solução é *sim* ou *não*.

Definição 1.10. *Dado um literal l designamos por* literal complementar *o literal $\tilde{l}$ definido por:*

$$\tilde{l} = \begin{cases} \neg l, & \text{se } l \text{ é uma variável (positivo)} \\ p, & \text{se } l \text{ é da forma } \neg p \text{ (negativo)} \end{cases}$$

Como vimos no Lema 1.3, uma cláusula é uma tautologia se contém um par de literais complementares p e $\neg p$. Estas cláusulas podem ser retiradas do conjunto sem alterar a satisfazibilidade.

1.3.5.2 O algoritmo de Davis-Putnam

O algoritmo que vamos apresentar é uma variante do original de 1960 [DP60, DLL62] e que continua a ser a base de muitos dos algoritmos que actualmente são mais competitivos (estando as diferenças nas estruturas de dados e nas diversas heurísticas que usam).
A ideia base do algoritmo é considerar para cada variável os possíveis valores de verdade e simplificar a fórmula de acordo com essas atribuições até se poder concluir que ela é ou não satisfazível. As simplificações incluem um caso especial para as cláusulas unitárias. Por uma questão de legibilidade vamos continuar a representação de cada cláusula usando disjunções (em vez de um conjunto de literais).

Definição 1.11. *Seja S um conjunto de cláusulas. Um conjunto S' é obtido de S por* propagação unitária *se S' se obtém de S por repetição da seguinte transformação: se S contém uma cláusula unitária (i.e. com um único literal l), então:*

1. *remover de S todas as cláusulas da forma $l \vee C'$*
2. *substituir em S cada cláusula da forma $\tilde{l} \vee C'$ pela cláusula C'.*

Exemplo 1.7. *Considera o seguinte conjunto de cláusulas:*

$$\{p_1, \neg p_1 \vee \neg p_2, p_3 \vee p_2, \neg p_7 \vee p_2, \neg p_3 \vee p_4, \neg p_3 \vee p_5, \neg p_4 \vee \neg p \vee q, \\ \neg p_5 \vee \neg p_6 \vee r, \neg p \vee \neg q \vee p_6, p \vee p_7, \neg r \vee p_7\} \tag{1.1}$$

Aplicando a propagação a p_1 resulta em:

$$\{\neg p_2, p_3 \vee p_2, \neg p_7 \vee p_2, \neg p_3 \vee p_4, \neg p_3 \vee p_5, \neg p_4 \vee \neg p \vee q, \\ \neg p_5 \vee \neg p_6 \vee r, \neg p \vee \neg q \vee p_6, p \vee p_7, \neg r \vee p_7\} \tag{1.2}$$

Aplicando a propagação a $\neg p_2$ resulta em:

$$\{p_3, \neg p_7, \neg p_3 \vee p_4, \neg p_3 \vee p_5, \neg p_4 \vee \neg p \vee q, \\ \neg p_5 \vee \neg p_6 \vee r, \neg p \vee \neg q \vee p_6, p \vee p_7, \neg r \vee p_7\} \tag{1.3}$$

Depois de aplicar a propagação a p_3 e $\neg p_7$, temos:

$$\{p_4, p_5, \neg p_4 \vee \neg p \vee q, \neg p_5 \vee \neg p_6 \vee r, \neg p \vee \neg q \vee p_6, p, \neg r\} \tag{1.4}$$

Propagando para p_4, p_5,$\neg r$ e p vem:

$$\{q, \neg p_6, \neg q \vee p_6\} \tag{1.5}$$

E finalmente propagando para q e $\neg p_6$:

$$\{\boldsymbol{F}\} \tag{1.6}$$

o que determina que o conjunto inicial de cláusulas é não satisfazível (e neste caso usando só propagação unitária).

O algoritmo básico de Davis-Putnam é o seguinte:

```
1  DLL(S) {
2    input: conjunto de clausulas S
3    output: satisfazivel ou nao satisfazivel
4    S := propaga(S)
5    if S vazio then return satisfazivel
6    if S contem F then return nao satisfazivel
7    l := selecciona_literal(S)
8    if DLL(S ∪ {l}) = satisfazivel
9       then return satisfazivel
10      else return DLL(S ∪ {l̃})
11 }
```

A função `propaga` implementa a propagação unitária. A função `seleciona_literal` retorna um literal da cláusula. Esta função poderá ser considerada um parâmetro do algoritmo, pois uma escolha adequada do literal pode tornar o algoritmo mais eficiente. Possíveis critérios são: escolher uma variável que ocorre mais vezes; que o produto das ocorrências de l e $\tilde{l}$ é máximo; que ocorre mais vezes em cláusulas de tamanho minimal, etc.
Para verificar a satisfazibilidade de um conjunto de fórmulas será necessário primeiro convertê-lo para um conjunto de cláusulas.

Exemplo 1.8. *Vamos aplicar o algoritmo `DLL` ao seguinte conjunto de cláusulas:*

$$S = \{\neg p \vee \neg q, \neg p \vee q, p \vee \neg q, p \vee q\} \tag{1.7}$$

Como não tem cláusulas unitárias não podemos aplicar a propagação. Temos que seleccionar um literal, por exemplo, $\neg p$. *Consideremos, primeiro, o conjunto* S *aumentado com* $\neg p$. *Podemos nesta caso aplicar a propagação* ($\Rightarrow$) *sucessivamente:*

$$\begin{aligned}\{\neg p \vee \neg q, \neg p \vee q, p \vee \neg q, p \vee q, \neg p\} &\Rightarrow \\ \{\neg q, q\} &\Rightarrow \{\boldsymbol{F}\}\end{aligned} \tag{1.8}$$

Como $S \cup \{\neg p\}$ *é não satisfazível, pelo algoritmo temos que considerar ainda* $S \cup \{p\}$. *Neste caso a propagação é:*

$$\begin{aligned}\{\neg p \vee \neg q, \neg p \vee q, p \vee \neg q, p \vee q, p\} &\Rightarrow \\ \{\neg q, q\} &\Rightarrow \{\boldsymbol{F}\}\end{aligned} \tag{1.9}$$

E o algoritmo retorna `nao satisfazivel`.

Pode-se demonstrar que:

Proposição 1.6. *O algoritmo* `DLL` *é correcto e completo para a satisfazibilidade de um conjunto de cláusulas* S, *i.e. o algoritmo retorna* `satisfazivel` *se* S *é satisfazível e retorna* `nao satisfazivel` *se* S *é não satisfazível.*

Podemos considerar ainda algumas optimizações a este algoritmo. A primeira é a *eliminação de tautologias* que pode ser feita apenas uma vez quando se converte uma fórmula (ou conjunto de fórmulas) para um conjunto de cláusulas (e usando o Lema 1.3), dado que o algoritmo não introduz tautologias. Outra optimização pode ser feita quando uma variável p só aparece positivamente ou só negativamente ($\neg p$) num conjunto de cláusulas.

Definição 1.12. *Um literal* l *é* puro *num conjunto de cláusulas* S, *se* S *não contém cláusulas da forma* $\tilde{l} \vee C$. *A* eliminação de literais puros *remove do conjunto de cláusulas todas as cláusulas que contêm um literal puro.*

Exemplo 1.9. *Considera a fórmula* $\neg((p \to q) \wedge (p \wedge q \to r) \to (\neg p \to r))$. *Uma FNC equivalente em forma clausal é:*

$$\{\neg p \vee q, \neg p \vee \neg q \vee r, \neg p, \neg r\} \tag{1.10}$$

O literal $\neg p$ *é puro neste conjunto, portanto todas as cláusulas em que ele ocorre podem ser eliminadas. Ficámos apenas com* $\{\neg r\}$. *Como também é puro, podemos eliminar e concluir que o conjunto é* satisfazível.

Para implementar esta eliminação, basta termos um contador com o número de ocorrências de cada literal l. Quando esse contador for 0, o literal $\tilde{l}$ é puro.

Exercício 1.17. *Aplica o algoritmo DLL ao seguinte conjunto de cláusulas:*

$$\{p \vee q \vee r, \neg p \vee \neg q \vee \neg r, p \vee \neg q \vee \neg rp \vee q \vee \neg r, \\ \neg p \vee q, \neg p \vee r, p \vee \neg q \vee r\} \tag{1.11}$$

◇

Exercício 1.18. *O seguinte conjunto de cláusulas formaliza o princípio de Dirichlet (ou* pigeonhole*) para 3 pombos e 2 poleiros: é impossível colocar 3 pombos em 2 poleiros de modo que cada poleiro só comtém um pombo! Considera o seguinte conjunto de variávies proposicionais* $\{p_{ij} \mid i = 1..3, j = 1, 2\}$ *em que* p_{ij} *denota que pombo* i *foi colocado no poleiro* j*. As cláusulas seguintes indicam que cada pombo é colocado nalgum poleiro:*

$$p_{11} \vee p_{12}, p_{21} \vee p_{22}, p_{31} \vee p_{32}$$

Agora é necessário formalizar que cada poleiro tem no máximo um pombo: para cada par de pombos i_1 *e* i_2 *e qualquer poleiro* j *tem-se que ambos não podem estar em* j*, i.e.* $\neg(p_{i_1 j} \wedge p_{i_2 j})$ *ou equivalentemente* $\neg p_{i_1 j} \vee \neg p_{i_2 j}$*. Temos então 6 cláusulas:*

$$\neg p_{11} \vee \neg p_{21}, \neg p_{11} \vee \neg p_{31}, \neg p_{21} \vee \neg p_{31}, \\ \neg p_{12} \vee \neg p_{22}, \neg p_{12} \vee \neg p_{32}, \neg p_{22} \vee \neg p_{32}.$$

Mostra a não satisfazibilidade o conjunto de 9 cláusulas usando o algoritmo `DLL`*.* ◇

Leituras suplementares [BE00] (Cap. 3, 4, 7) [HR00] (Cap 1.5) [GLM97] (Cap 2.2, 4.3)

1.4 Sistemas dedutivos

Considera os raciocínios seguintes:

1	Todos os homens são mortais
2	Sócrates é um homem
3	Sócrates é mortal

e

1	Todos os actores ricos são bons actores
2	Brad Pitt é um actor rico
3	Brad Pitt é bom actor

Informalmente, um *raciocínio* é uma sequência de afirmações das quais uma – a conclusão – deve ser consequência das restantes – as premissas. A conclusão é uma consequência lógica

das premissas, se for verdadeira sempre que as premissas forem verdadeiras. Neste caso temos uma *raciocínio válido*. Num raciocínio válido se as premissas forem verdadeiras, o raciocínio é *integro*. Nos exemplos anteriores, o primeiro raciocínio é íntegro, mas o segundo não: a primeira premissa é falsa!

1.4.1 Métodos de dedução

Como podemos mostrar que uma conclusão *é uma consequência lógica* das premissas?
Construindo uma *sucessão de passos* em que em cada um a conclusão é inequivocamente consequência das conclusões e premissas anteriores. Formalmente iremos considerar *sistemas de dedução*.
Por outro lado, para mostrar que uma conclusão *não é consequência lógica* das premissas, temos que mostrar que existe uma situação em que as premissas podem ser verdadeiras e a conclusão falsa. Essa situação é designada de *contra-exemplo*.

1.4.2 Sistemas de dedução axiomáticos

Um sistema de dedução axiomático $\mathcal{D}$ é um método sintáctico, constituído por:

axiomas (lógicos): fórmulas base, que caracterizam as propriedades das conectivas

regras de inferência: modos de obter fórmulas a partir de outras

Definição 1.13. *Uma sucessão finita de fórmulas $\varphi_1, \ldots \varphi_n$ é uma* dedução *de φ_n em $\mathcal{D}$ a partir de um conjunto Σ de fórmulas se para cada $1 \leq i \leq n$ se verifica:*

- $\varphi_i \in \Sigma$
- φ_i *é um axioma*
- φ_i *resulta de $\varphi_1 \ldots \varphi_{i-1}$ por aplicação duma regra de inferência*

Neste caso diz-se também que φ_n pode ser deduzido a partir de Σ e escreve-se $\Sigma \vdash_{\mathcal{D}} \varphi_n$. A fórmula φ_n é um teorema *(de $\mathcal{D}$) se $\Sigma = \emptyset$ e escreve-se $\vdash_{\mathcal{D}} \varphi_n$. Neste caso, a dedução $\varphi_1, \ldots \varphi_n$ diz-se uma* demonstração *de φ_n. Se $\Sigma = \{\theta_1, \ldots, \theta_n\}$ é finito, em vez de $\Sigma \vdash_{\mathcal{D}} \varphi$ escreve-se*

$$\theta_1, \ldots, \theta_n \vdash_{\mathcal{D}} \varphi$$

e $\mathcal{D}$ será omitido se for explícito no contexto.

1.4.3 Sistema de dedução natural, DN

Sistema inventado por G. Gentzen (1935) (e também por S. Jaskowski), e cujas regras pretendem reflectir as formas de raciocínio usadas nas demonstrações matemáticas. Não tem axiomas, só regras de inferência. Uma das suas originalidades, em relação a outros sistemas axiomáticos, é possibilidade de introduzir hipóteses no meio da dedução, mas que terão se ser eliminadas antes da dedução perminar. Outra característica é a dualidade das regras. Para cada conectiva lógica existem dois tipos de regras: de *introdução* (da conectiva) e de *eliminação* (da conectiva).
Consideremos um exemplo. Suponhamos que queremos concluir que

$$(p \vee (q \wedge r)) \rightarrow ((p \vee q) \wedge (p \vee r))$$

é uma tautologia (é válida).
Para tal, supomos que $(p \vee (q \wedge r))$ se verifica e tentamos concluir $((p \vee q) \wedge (p \vee r))$. Como temos uma disjunção no antecedente, temos que supor separadamente que p se verifica ou que $(q \wedge r)$ se verifica (*eliminação de* $\vee$). Suponhamos p, então $p \vee q$ também se verifica (pois numa disjunção basta que um se verifique) (*introdução de* $\vee$), e também temos $p \vee r$ (*introdução de* $\vee$). Mas então, também se verifica a sua conjunção $((p \vee q) \wedge (p \vee r))$ (*introdução de* $\wedge$). Agora se $(q \wedge r)$ se verifica, então q e r verificam-se (*eliminação de* $\wedge$). Então $(p \vee q)$ e $(p \vee r)$, também se verifica, assim como a sua conjunção. Temos a seguinte *árvore de demonstração*:

p	$q \wedge r$
$p \vee q$	q
$p \vee r$	r
$((p \vee q) \wedge (p \vee r))$	$p \vee q$
	$p \vee r$
	$((p \vee q) \wedge (p \vee r))$

Como supondo $(p \vee (q \wedge r))$ se "deduz" $((p \vee q) \wedge (p \vee r))$, podemos concluir que $(p \vee (q \wedge r)) \rightarrow ((p \vee q) \wedge (p \vee r))$ (*introdução de* $\rightarrow$).
Vamos agora formalizar as regras de inferência DN.
Uma regra de inferência é da forma:

$$\text{de } \varphi_1 \ldots \varphi_k \text{ infere-se } \varphi_n$$

e pode ser representada graficamente (em árvore) por:

$$\frac{\varphi_1, \ldots, \varphi_k}{\varphi_n}$$

1.4.3.1 Regras DN para a conjunção

Introdução de $\wedge$

$$\frac{\varphi \quad \psi}{\varphi \wedge \psi}\wedge\mathbf{I}$$

Se já deduzimos φ e ψ então podemos deduzir $\varphi \wedge \psi$

Eliminação de $\wedge$

$$\frac{\varphi \wedge \psi}{\varphi}\wedge\mathbf{E}_1 \qquad \frac{\varphi \wedge \psi}{\psi}\wedge\mathbf{E}_2$$

Se deduzimos $\varphi \wedge \psi$ podemos deduzir φ; e podemos também deduzir ψ.

Exemplo 1.10. *Mostrar que* $p \wedge q, r \vdash q \wedge r$.

Resolução 1.10.1. *Podemos construir a dedução numa* árvore*:*

$$\frac{\dfrac{p \wedge q}{q}\wedge \boldsymbol{E}_2 \qquad r}{q \wedge r}\wedge \boldsymbol{I}$$

em que as folhas são as premissas. Mas estas árvores podem ficar muito grandes, portanto vamos considerar uma representação linear para as deduções: numeram-se os passos, separam-se as premissas e, para cada passo, indica-se qual a regra a aplicar e quais as fórmulas que intervêm. Para a dedução anterior, temos

1	$p \wedge q$	
2	r	
3	q	$\wedge E$, 1
4	$q \wedge r$	$\wedge I$, 3, 2

Esta notação para a representação de deduções naturais denomina-se notação de Fitch.

Exemplo 1.11. *Mostrar que* $(p \wedge q) \wedge r \vdash r \wedge q$

Resolução 1.11.1.

1	$(p \wedge q) \wedge r$	
2	$p \wedge q$	$\wedge E$, 1
3	r	$\wedge E$, 1
4	q	$\wedge E$, 2
5	$r \wedge q$	$\wedge I$, 3, 4

As regras para a conjunção, em notação de Fitch, são:

$$\begin{array}{|l} \vdots \\ \varphi \wedge \psi \\ \vdots \\ \varphi \qquad\qquad \wedge\mathrm{E} \end{array}$$

$$\begin{array}{|l} \vdots \\ \varphi \wedge \psi \\ \vdots \\ \psi \qquad\qquad \wedge\mathrm{E} \end{array}$$

$$\begin{array}{|l} \vdots \\ \varphi \\ \vdots \\ \psi \\ \vdots \\ \varphi \wedge \psi \qquad \wedge\mathrm{I} \\ \vdots \end{array}$$

1.4.3.2 Regras DN para a disjunção

Introdução de $\vee$

$$\frac{\varphi}{\varphi \vee \psi}\vee\mathbf{I_1} \qquad \frac{\psi}{\varphi \vee \psi}\vee\mathbf{I_2}$$

Se já deduzimos φ podemos deduzir qualquer disjunção que contenha φ.

Eliminação de $\vee$

$$\frac{\begin{array}{ccc} & [\varphi] & [\psi] \\ & \vdots & \vdots \\ \varphi \vee \psi & \gamma & \gamma \end{array}}{\gamma}\vee\mathbf{E}$$

- se já deduzimos a disjunção $\varphi \vee \psi$

- se supusermos φ deduzirmos γ (numa sub-dedução)
- e se supusermos ψ deduzirmos γ (numa sub-dedução)
- então podemos deduzir γ

Nestas regras, expressões $[\varphi]$ indicam que estamos iniciar uma sub-dedução com premissa φ, que só deve ser considerada nessa sub-dedução. As sub-deduções têm de terminar pela ordem em que foram iniciadas. Neste caso as sub-deduções terminam quando γ for deduzido. Na notação de Fitch, cada sub-dedução é iniciada com uma indentação.

Exemplo 1.12. *Mostrar que* $(p \wedge q) \vee (q \wedge r) \vdash q$.

Resolução 1.12.1.

1	$(p \wedge q) \vee (q \wedge r)$	
2	$\quad p \wedge q$	
3	$\quad q$	$\wedge E$, 2
4	$\quad q \wedge r$	
5	$\quad q$	$\wedge E$, 4
6	q	$\vee E$, 1, 2–3, 4–5

As regras para a disjunção, em notação de Fitch, são:

$$\begin{array}{|l} \vdots \\ \varphi \\ \vdots \\ \varphi \vee \psi \qquad \vee\mathrm{I} \end{array}$$

$$\begin{array}{|l} \vdots \\ \varphi \\ \vdots \\ \psi \vee \varphi \qquad \vee\mathrm{I} \end{array}$$

$$
\begin{array}{|l}
\vdots \\
\varphi \vee \psi \\
\vdots \\
\quad \begin{array}{|l} \varphi \\ \hline \vdots \\ \gamma \end{array} \\
\quad \begin{array}{|l} \psi \\ \hline \vdots \\ \gamma \end{array} \\
\gamma \qquad\qquad \vee\mathrm{E}
\end{array}
$$

1.4.3.3 Regra DN de Repetição

Numa dedução podemos sempre repetir uma conclusão já obtida. A essa regra chamaremos **repetição**:

$$\frac{\varphi}{\varphi}\mathbf{R}$$

Exemplo 1.13. *Mostrar que* $(p \wedge q) \vee q \vdash q$

Resolução 1.13.1.

$$
\begin{array}{lll|l}
1 & (p \wedge q) \vee q & & \\
\hline
2 & \quad \begin{array}{|l} p \wedge q \\ \hline \end{array} & & \\
3 & \quad \begin{array}{|l} q \end{array} & & \wedge E,\ 2 \\
4 & \quad \begin{array}{|l} q \\ \hline \end{array} & & \\
5 & \quad \begin{array}{|l} q \end{array} & & R,\ 4 \\
6 & q & & \vee E,\ 1,\ 2\text{–}3,\ 4\text{–}5
\end{array}
$$

1.4.3.4 Utilização de sub-deduções

Como já foi referido, uma dedução pode ser composta por sub-deduções que introduzem novas premissas. Mas nem essa premissa, nem as fórmulas delas deduzidas podem ser usadas depois da sub-dedução em que ocorrem terminar. Considera a seguinte *dedução*:

1	$(p \wedge q) \vee (q \wedge r)$	
2	$\quad p \wedge q$	
3	$\quad p$	$\wedge$E, 2
4	$\quad q$	$\wedge$E, 2
5	$\quad q \wedge r$	
6	$\quad q$	$\wedge$E, 5
7	q	$\vee$E, 1, 2–4, 5–6
8	$q \wedge p$	$\wedge$I, 7, 3

Esta dedução está **ERRADA**! No passo 8 é usado um passo que ocorre numa sub-dedução que já terminou. Uma sub-dedução é iniciada com a introdução de novas hipóteses (premissas) e as deduções aí feitas dependem delas. Quando termina a sub-dedução, essas hipóteses deixam ser assumidas e portanto não se podem utilizar!

1.4.3.5 Regras DN para a Negação

Eliminação de $\neg$

Corresponde a uma das partes do princípio da dupla negação.

$$\frac{\neg\neg\varphi}{\varphi}\,\neg\mathbf{E}$$

Introdução de $\neg$

Esta regra corresponde a demonstrações por contradição. Representamos por **F** uma contradição (p.e., $\varphi \wedge \neg\varphi$).

$$\begin{array}{c} [\varphi] \\ \vdots \\ \mathbf{F} \\ \hline \neg\varphi \end{array}\,\neg\mathbf{I}$$

Se supondo φ podemos deduzir uma contradição, então podemos deduzir $\neg\varphi$ das premissas originais.

Na notação de Fitch temos:

$$\begin{array}{|l} \vdots \\ \neg\neg\varphi \\ \vdots \\ \varphi \qquad\qquad \neg\mathrm{E} \end{array}$$

$$\begin{array}{|l} \vdots \\ \quad \begin{array}{|l} \varphi \\ \hline \quad \vdots \\ \quad \mathbf{F} \end{array} \\ \neg\varphi \qquad\qquad \neg\mathrm{I} \end{array}$$

1.4.3.6 Regras *DN* para F

Se não considerarmos **F** como uma abreviatura de $\varphi \wedge \neg\varphi$, temos de ter uma regra para o introduzir[3]:

Introdução de F

$$\dfrac{\begin{array}{c}\varphi \\ \vdots \\ \neg\varphi\end{array}}{\mathbf{F}}\,\mathbf{FI}$$

Se deduzimos φ e $\neg\varphi$ então temos uma contradição.

Exemplo 1.14. *Mostrar que* $\varphi \vdash \neg\neg\varphi$.

Resolução 1.14.1.

1	φ	
2	$\quad \neg\varphi$	
3	$\quad \boldsymbol{F}$	$\boldsymbol{FI}$, 1, 2
4	$\neg\neg\varphi$	$\neg I$, 2–3

Eliminação de F

$$\dfrac{\mathbf{F}}{\varphi}\,\mathbf{FE}$$

Se deduzimos uma contradição, então podemos deduzir qualquer fórmula.

[3] caso contrário podemos ignorar...

Definição 1.14. *Um conjunto de fórmulas* Σ *diz-se* inconsistente *se* $\Sigma \vdash \boldsymbol{F}$.

$$\begin{array}{|l} \vdots \\ \varphi \\ \vdots \\ \neg\varphi \\ \vdots \\ \mathbf{F} \quad \mathbf{FI} \end{array}$$

$$\begin{array}{|l} \vdots \\ \mathbf{F} \\ \vdots \\ \varphi \quad \mathbf{FE} \\ \vdots \end{array}$$

1.4.3.7 Métodos de demonstração

Vamos ilustrar algumas aplicações das regras anteriores, em demonstrações em matemática.

Demonstração por casos A regra da **eliminação da disjunção** corresponde ao método de demonstração por casos. Consideremos o seguinte problema:

Mostrar que existem irracionais b *e* c *tal que* b^c *é racional*

Demonstração. Demonstração por casos: Seja $\sqrt{2}^{\sqrt{2}}$. Este número é racional ou irracional.

- Se $\sqrt{2}^{\sqrt{2}}$ é racional então basta tomar $b = c = \sqrt{2}$
- Se $\sqrt{2}^{\sqrt{2}}$ é irracional, então seja $b = \sqrt{2}^{\sqrt{2}}$ e $c = \sqrt{2}$. Vem $b^c = \sqrt{2}^{\sqrt{2}.\sqrt{2}} = \sqrt{2}^2 = 2$, que é racional.

□

É de referir que a demonstração anterior não é construtiva, uma vez que não foi determinado se $\sqrt{2}^{\sqrt{2}}$ é ou não racional.

Demonstração por contradição A regra da **introdução da negação**, é usada nas demonstrações por contradição, isto é, supõe-se a negação do que se quer provar e chega-se a um absurdo. Suponhamos o seguinte problema:

Mostrar que $\sqrt{2}$ não é racional.

Demonstração. Suponhamos que $\sqrt{2}$ é racional. Então existem p e q tal que $\sqrt{2} = p/q$, com um deles ímpar (porquê?). Então $\frac{p^2}{q^2} = 2$. E $p^2 = 2q^2$. Então p^2 é par e p também (verifica!). E $4 \mid p^2$ e $4 \mid 2q^2$. Mas então q^2 também é par! Temos então uma contradição. □

Exemplo 1.15. *Mostrar usando o sistema DN:*

a) $\neg p \vee \neg q \vdash \neg(p \wedge q)$

b) $\vdash \neg(p \wedge \neg p)$

c) $\neg(\neg p \vee q) \vdash p$

Resolução 1.15.1. *a)* $\neg p \vee \neg q \vdash \neg(p \wedge q)$

1	$\neg p \vee \neg q$	
2	$\quad \neg p$	
3	$\quad\quad p \wedge q$	
4	$\quad\quad p$	$\wedge E$, 3
5	$\quad\quad \boldsymbol{F}$	$\boldsymbol{FI}$, 2, 4
6	$\quad \neg(p \wedge q)$	$\neg I$, 3–5
7	$\quad \neg q$	
8	$\quad\quad p \wedge q$	
9	$\quad\quad q$	$\wedge E$, 8
10	$\quad\quad \boldsymbol{F}$	$\boldsymbol{FI}$, 7, 9
11	$\quad \neg(p \wedge q)$	$\neg I$, 8–10
12	$\neg(p \wedge q)$	$\vee E$, 1, 2–11

b) $\vdash \neg(p \wedge \neg p)$

1	$\quad p \wedge \neg p$	
2	$\quad p$	$\wedge E$, 1
3	$\quad \neg p$	$\wedge E$, 1
4	$\quad \boldsymbol{F}$	$\boldsymbol{FI}$, 2, 3
5	$\neg(p \wedge \neg p)$	$\neg I$, 1–4

c) $\neg(\neg p \vee q) \vdash p$

1	$\neg(\neg p \vee q)$	
2	$\quad \neg p$	
3	$\quad \neg p \vee q$	$\vee I$, 2
4	$\quad \boldsymbol{F}$	$\boldsymbol{FI}$, 1, 3
5	$\neg\neg p$	$\neg I$, 2–4
6	p	$\neg E$, 2–4

Leituras suplementares [BE00] (Cap. 2,5,6)

1.4.3.8 Regras *DN* para a implicação

Eliminação de $\rightarrow$ (*modus ponens*)

Esta regra é habitualmente conhecida por *modus ponens* (em latim, *modo que afirma*) e corresponde a raciocínios condicionais:

$$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi} \rightarrow \mathbf{E}$$

Se já deduzimos φ e $\varphi \rightarrow \psi$, então podemos deduzir ψ.

Exemplo 1.16. *Mostrar que* $p, p \rightarrow q, p \rightarrow (q \rightarrow r) \vdash r$.

Resolução 1.16.1.

1	$p \rightarrow (q \rightarrow r)$	
2	$p \rightarrow q$	
3	p	
4	$q \rightarrow r$	$\rightarrow E$, 1, 3
5	q	$\rightarrow E$, 2, 3
6	r	$\rightarrow E$, 4, 5

Introdução de $\rightarrow$ (regra da dedução)

A regra para introduzir uma implicação necessita duma sub-dedução: supondo φ tentamos deduzir ψ. Se tal acontecer, terminamos a sub-dedução (retirando a suposição) e concluímos $\varphi \rightarrow \psi$:

$$\begin{array}{c} [\varphi] \\ \vdots \\ \psi \end{array} \rightarrow \mathbf{I}$$

Exemplo 1.17. *Mostrar que* $(p \vee q) \to r \vdash p \to r$.

Resolução 1.17.1.

1	$(p \vee q) \to r$	
2	$\quad p$	
3	$\quad p \vee q$	$\vee I$, 2
4	$\quad r$	$\to E$, 1, 3
5	$p \to r$	$\to I$, 2–4

As regras para a conjunção, em notação de Fitch, são:

$\vdots$	
φ	
$\vdots$	
$\varphi \to \psi$	
$\vdots$	
ψ	$\to$E

$\vdots$	
$\quad \varphi$	
$\quad \vdots$	
$\quad \psi$	
$\varphi \to \psi$	$\to$I

1.4.3.9 Deduções sem premissas

Com a introdução da implicação (regra da dedução) podemos converter qualquer dedução com premissas numa dedução sem premissas:

Exemplo 1.18. *Mostrar que* $\vdash \varphi \to \neg\neg\varphi$.

Resolução 1.18.1.

1	$\quad \varphi$	
2	$\quad\quad \neg\varphi$	
3	$\quad\quad \boldsymbol{F}$	$\boldsymbol{F I}$, 1, 2
4	$\quad \neg\neg\varphi$	$\neg I$, 2–3
5	$\varphi \to \neg\neg\varphi$	$\to I$, 1–4

Em geral temos:

Lema 1.4. ***(da dedução)*** $\Sigma \cup \{\varphi\} \vdash \psi$ *se e só se* $\Sigma \vdash \varphi \to \psi$.

Demonstração. $\Rightarrow$: Se $\Sigma \cup \{\varphi\} \vdash \psi$, obtemos $\Sigma \vdash \varphi \to \psi$, supondo φ, usando a dedução anterior até obter ψ e aplicando a regra $\to \mathbf{I}$.
$\Leftarrow$: Se $\Sigma \vdash \varphi \to \psi$, para obter $\Sigma \cup \{\varphi\} \vdash \psi$, usamos apenas dedução de ψ supondo φ. □

1.4.3.10 Algumas regras derivadas de *DN*

A partir das regras base podemos obter regras derivadas que correspondem a teoremas no sistema *DN* (eventualmente usando o lema da dedução).

Modus Tollens (em latim, *modo que nega*)

$$\frac{\varphi \to \psi \qquad \neg\psi}{\neg\varphi}\,\mathbf{MT}$$

Exemplo 1.19. *Mostrar que* $\varphi \to \psi, \neg\psi \vdash \neg\varphi$.

Resolução 1.19.1.

1	$\varphi \to \psi$	
2	$\neg\psi$	
3	$\quad\varphi$	
4	$\quad\psi$	$\to E$, 1, 3
5	$\quad\mathbf{F}$	$\mathbf{F}\boldsymbol{I}$, 2, 4
6	$\neg\varphi$	$\neg I$, 3–5

Introdução da dupla negação

$$\frac{\varphi}{\neg\neg\varphi}\,\neg\neg\mathbf{I}$$

Redução ao absurdo

$$\frac{\begin{array}{c}[\neg\varphi]\\ \vdots \\ \mathbf{F}\end{array}}{\varphi}\,\mathbf{RA}$$

Se tivermos uma dedução de $\mathbf{F}$ supondo $\neg\varphi$ podemos ter uma dedução de $\neg\varphi \to \mathbf{F}$. Então basta mostrar $\neg\varphi \to \mathbf{F} \vdash \varphi$:

1	$\neg\varphi \to \mathbf{F}$	
2	$\quad\neg\varphi$	
3	$\quad\mathbf{F}$	$\to$E, 1, 2
4	$\neg\neg\varphi$	$\neg$I, 2–3
5	φ	$\neg$E, 4

Terceiro excluído

$$\frac{}{\varphi \vee \neg\varphi}\,\mathbf{TE}$$

1	$\neg(\varphi \vee \neg\varphi)$	
2	$\quad\varphi$	
3	$\quad\varphi \vee \neg\varphi$	$\vee$I, 2
4	$\quad\mathbf{F}$	**FI**, 1, 3
5	$\neg\varphi$	$\neg$I, 2–4
6	$\varphi \vee \neg\varphi$	$\vee$I, 5
7	$\mathbf{F}$	**FI**, 1, 5
8	$\varphi \vee \neg\varphi$	**RA**, 1–7

Exemplo 1.20. *Mostrar* $\neg q \to \neg p \vdash p \to \neg\neg q$.

Resolução 1.20.1.

1	$\neg q \to \neg p$	
2	$\quad p$	
3	$\quad\neg\neg p$	$\neg I$, 2
4	$\quad\neg\neg q$	***MT***, 1, 3
5	$p \to \neg\neg q$	$\to I$, 2–4

Observações para a obtenção de uma dedução duma fórmula φ

- Se a fórmula φ for uma implicação $\psi \to \theta$, supor ψ e deduzir θ; aplicando de seguida a regra da introdução da implicação.
- Se a fórmula φ for uma negação $\neg\psi$, supor ψ e deduzir **F**; aplicando de seguida a regra da introdução da negação.
- Por redução ao absurdo: supor $\neg\varphi$ e deduzir **F**. Caso haja uma negação $\neg\psi$ nas premissas (ou já deduzida), poder-se-á deduzir ψ para obter **F**.

- Suponhamos que uma das premissas é $\varphi \vee \psi$ e se pretende deduzir γ. Podemos: supor φ e deduzir γ, supor ψ e deduzir γ; aplicando de seguida a regra de eliminação da disjunção.
- Suponhamos que uma premissa é $\neg\varphi$ e pretendemos obter **F**: podemos tentar obter φ e aplicar a regra da introdução de **F**.
- Suponhamos que temos uma implicação $\phi \to \theta$. Podemos entao tentar deduzir ϕ e aplicar a regra da eliminação da implicação (modus ponens) para obter θ.

Notar que cada passo da dedução tem de ser obtido por aplicação duma regra ou ser uma repetição duma fórmula já deduzida ou uma premissa. Assim cada passo de dedução tem de ser ou uma suposição ou resultar da aplicação duma regra a fórmulas já deduzidas, pelo que não se podem usar "equivalências semânticas".

1.4.3.11 Equivalência dedutiva

Dadas dumas fórmulas φ e ψ, dizemos que φ e ψ são *dedutivamente equivalentes* se e só se $\varphi \vdash \psi$ e $\psi \vdash \varphi$. E denotamos por $\varphi \dashv\vdash \psi$.

Exercício 1.19. *Mostra que* $\varphi \dashv\vdash \psi$ *se e só se* $\vdash (\varphi \to \psi) \wedge (\psi \to \varphi)$. $\diamond$

Proposição 1.7 (Contraposição)**.**

$$\varphi \to \psi \dashv\vdash \neg\psi \to \neg\varphi$$

Demonstração. Vamos mostrar só $\varphi \to \psi \vdash \neg\psi \to \neg\varphi$:

1	$\varphi \to \psi$	
2	$\quad \neg\psi$	
3	$\quad \neg\varphi$	**MT**, 1, 2
4	$\neg\psi \to \neg\varphi$	$\to$I, 2–3

□

1.4.3.12 Ilustração da aplicação das regras

Demonstração duma afirmação condicional Como já vimos, a regra da **introdução da implicação** corresponde à demonstração de uma condicional. Suponhamos que queremos mostrar que:

Se n^2 *é par então* n *é par*

Demonstração. Suponhamos que n^2 é par. Vamos provar que n é par, por contradição. Suponhamos que n é ímpar, então $n = 2m + 1$, para algum m. Então:

$$n^2 = (2m+1)^2 = 4m^2 + 4m + 1 = 2(2m^2 + 2m) + 1$$

que é ímpar! Então n é par. E temos demonstrada a implicação. □

Demonstração por contraposição De igual modo temos as demonstrações por contraposição. Suponhamos que queremos mostrar que:

Se n não é par então n^2 não é par

Demonstração. Suponhamos que n não é par, i.e é da forma $n = 2m + 1$, para algum m. Então:

$$n^2 = (2m+1)^2 = 4m^2 + 4m + 1 = 2(2m^2 + 2m) + 1$$

o que mostra que n^2 é ímpar, e portanto não é par. Então temos demonstrada a implicação. □

Neste caso evita-se a redução ao absurdo...

Exemplo 1.21. *Mostra que:*

a) $\vdash p \vee \neg(p \wedge q)$

b) $\neg(\varphi \wedge \psi) \vdash \neg\varphi \vee \neg\psi$

c) $\vdash (p \wedge q) \vee \neg p \vee \neg q$

d) $\varphi \to \psi \vdash \neg\varphi \vee \psi$

e) $\vdash \varphi \to (\psi \to \varphi)$

f) $\vdash (\varphi \to (\psi \to \theta)) \to ((\varphi \to \psi) \to (\varphi \to \theta))$

g) $\vdash (\neg\psi \to \neg\varphi) \to ((\neg\psi \to \varphi) \to \psi)$

Resolução 1.21.1. *a)* $\vdash p \vee \neg(p \wedge q)$

*Supondo a regra do terceiro excluído (**TE**)*

1	$\neg(p \wedge q) \vee (p \wedge q)$	***TE***
2	$\quad p \wedge q$	
3	$\quad p$	$\wedge E$, 2
4	$\quad p \vee \neg(p \wedge q)$	$\vee I$, 3
5	$\quad \neg(p \wedge q)$	
6	$\quad p \vee \neg(p \wedge q)$	$\vee I$, 5
7	$p \vee \neg(p \wedge q)$	$\vee E$, 1, 2–4, 5–6

b) $\neg(\varphi \wedge \psi) \vdash \neg\varphi \vee \neg\psi$

1	$\neg(\varphi \wedge \psi)$	
2	$\quad\neg(\neg\varphi \vee \neg\psi)$	
3	$\quad\quad\neg\varphi$	
4	$\quad\quad\neg\varphi \vee \neg\psi$	$\vee I$, 3
5	$\quad\quad$ ***F***	***FI***, 2, 4
6	$\quad\neg\neg\varphi$	$\neg I$, 3–5
7	$\quad\varphi$	$\neg E$, 6
8	$\quad\quad\neg\psi$	
9	$\quad\quad\neg\varphi \vee \neg\psi$	$\vee I$, 3
10	$\quad\quad$ ***F***	***FI***, 2, 9
11	$\quad\neg\neg\psi$	$\neg I$, 8–10
12	$\quad\psi$	$\neg E$, 11
13	$\quad\varphi \wedge \psi$	$\wedge I$, 7, 12
14	$\quad$ ***F***	***FI***, 1, 13
15	$\neg\neg(\neg\varphi \vee \neg\psi)$	$\neg I$, 2, 14
16	$\neg\varphi \vee \neg\psi$	$\neg E$, 15

c) $\vdash (p \wedge q) \vee \neg p \vee \neg q$

Usar a regra do terceiro excluído com $(p \wedge q) \vee \neg(p \wedge q)$ *e ver a resolução do exercício anterior*

d) $\varphi \to \psi \vdash \neg\varphi \vee \psi$

1	$\varphi \to \psi$	
2	$\quad \neg(\neg\varphi \vee \psi)$	
3	$\quad\quad \neg\varphi$	
4	$\quad\quad \neg\varphi \vee \psi$	$\vee I$, 3
5	$\quad\quad \boldsymbol{F}$	$\boldsymbol{F I}$, 2, 4
6	$\quad \neg\neg\varphi$	$\neg I$, 3–5
7	$\quad \varphi$	$\neg E$, 6
8	$\quad \psi$	$\to E$, 1, 7
9	$\quad \neg\varphi \vee \psi$	$\vee I$, 8
10	$\quad \boldsymbol{F}$	$\boldsymbol{F I}$, 2, 9
11	$\neg\neg(\neg\varphi \vee \psi)$	$\neg I$, 2–10
12	$\neg\varphi \vee \psi$	$\neg E$, 11

e) $\vdash \varphi \to (\psi \to \varphi)$

1	$\quad \varphi$	
2	$\quad\quad \psi$	
3	$\quad\quad \varphi$	R, 1
4	$\quad \psi \to \varphi$	$\to I$, 2–3
5	$\varphi \to (\psi \to \varphi)$	$\to I$, 1–4

f) $\vdash (\varphi \to (\psi \to \theta)) \to ((\varphi \to \psi) \to (\varphi \to \theta))$

1	$\quad \varphi \to (\psi \to \theta)$	
2	$\quad\quad (\varphi \to \psi)$	
3	$\quad\quad\quad \varphi$	
4	$\quad\quad\quad \psi$	$\to E$, 2, 3
5	$\quad\quad\quad \psi \to \theta$	$\to E$, 1, 3
6	$\quad\quad\quad \theta$	$\to E$, 4, 5
7	$\quad\quad \varphi \to \theta$	$\to I$, 3–6
8	$\quad (\varphi \to \psi) \to (\varphi \to \theta)$	$\to I$, 2–7
9	$\varphi \to (\psi \to \theta) \to (\varphi \to \psi) \to (\varphi \to \theta)$	$\to I$, 1–8

g) $\vdash (\neg\psi \to \neg\varphi) \to ((\neg\psi \to \varphi) \to \psi)$

1	$\neg\psi \to \neg\varphi$	
2	$\neg\psi \to \varphi$	
3	$\neg\psi$	
4	φ	$\to E$, 2, 3
5	$\neg\varphi$	$\to E$, 1, 2
6	$\boldsymbol{F}$	$\boldsymbol{F I}$, 4, 5
7	$\neg\neg\psi$	$\neg I$, 3–6
8	ψ	$\neg E$, 7
9	$(\neg\psi \to \varphi) \to \psi$	$\to I$, 2–8
10	$(\neg\psi \to \neg\varphi) \to (\neg\psi \to \varphi) \to \psi$	$\to I$, 1–9

Leituras suplementares [BE00] (Cap. 8.1-2)

1.5 Integridade e completude de um sistema dedutivo

Um sistema dedutivo é um conjunto de regras, puramente sintácticas. Para que possa ser usado para obter a validade ou consequência lógica de fórmulas é necessário (desejável) que seja **íntegro** e **completo**. Isto é: que o que se deduza seja consequência semântica das premissas, e se uma fórmula for consequência semântica das premissas então existe uma dedução para ela.

Definição 1.15 (Integridade)**.** *Um sistema dedutivo é* íntegro*, se dado um conjunto* Σ *de premissas e uma conclusão* φ*, se existe uma dedução de* φ *com premissas* Σ*, i.e* $\Sigma \vdash_{\mathcal{D}} \varphi$*, então a conclusão* φ *é consequência semântica de* Σ*, i.e* $\Sigma \models \varphi$*. Em particular, se* $\vdash_{\mathcal{D}} \varphi$ *então* φ *é uma tautologia (*$\models \varphi$*) .*

Definição 1.16 (Completude)**.** *Um sistema dedutivo é* completo*, se dado um conjunto* Σ *de premissas e uma conclusão* φ*, se* φ *é consequência semântica de* Σ*, i.e* $\Sigma \models \varphi$*, então existe uma dedução de* φ *com premissas* Σ*, i.e* $\Sigma \vdash_{\mathcal{D}} \varphi$*. Em particular, se* φ *é uma tautologia (*$\models \varphi$*) então* $\vdash_{\mathcal{D}} \varphi$*.*

1.5.1 Integridade do sistema de dedução natural *DN*

Proposição 1.8. *Se* $\Sigma \vdash \varphi$ *então* $\Sigma \models \varphi$*.*

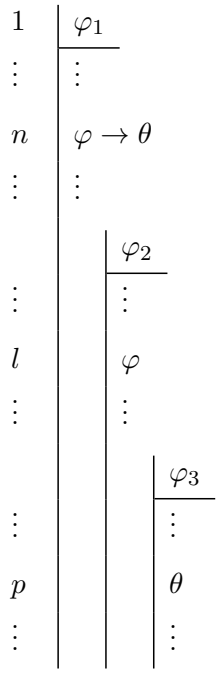
Demonstração. Suponhamos que temos uma dedução de φ, $\varphi_1, \ldots, \varphi_n = \varphi$. Vamos mostrar que em cada passo a fórmula que aí ocorre é consequência semântica das premissas (ou hipóteses) que aí são assumidas.

Provamos por redução ao absurdo: Suponhamos que existe um passo p que contém uma fórmula que não é consequência semântica das premissas assumidas em p. E seja p o primeiro desses passos . Vamos ver que qualquer que seja a regra de DN aplicada em p, temos uma contradição. O que permite concluir que não existe tal passo p. Fazemos a demonstração por casos, considerando cada uma das regras. Basicamente temos dois tipos de casos: os que correspondem a regras com sub-deduções e os a regras sem sub-deduções.

Suponhamos que a regra a aplicar é a eliminação da implicação:

$\rightarrow$ **E**

$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$: Seja θ a fórmula deduzida no passo p por aplicação de $\rightarrow$ **E** a $\varphi \rightarrow \theta$ e φ. E sejam $\varphi_1, \ldots, \varphi_k$ as premissas assumidas em θ, e, por hipótese, θ não é consequência semântica delas. Mas as premissas para $\varphi \rightarrow \theta$ e φ estão entre os $\varphi_1, \ldots, \varphi_k$ e ambos são consequência semânticas delas:



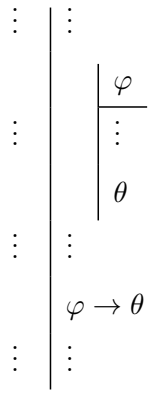
Considera a tabela de verdade para as fórmulas $\varphi_1, \ldots, \varphi_k, \varphi, \varphi \rightarrow \theta$ e θ. Por hipótese, existe uma valoração v tal que $v(\varphi_i) = \mathbf{V}$, $1 \leq i \leq k$ e $v(\theta) = \mathbf{F}$. Mas também tem-se que $v(\varphi) = v(\varphi \rightarrow \theta) = \mathbf{V}$. Mas isto contradiz a tabela de verdade para a implicação. Portanto concluímos que esta regra não pode ser aplicada no passo p (se existir).

Vejamos agora o caso da regra da introdução da implicação:

$\rightarrow$ **I**

$$\dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \psi\end{array}}{\varphi\rightarrow\psi}$$

: Suponhamos que o passo p deduz $\varphi \rightarrow \theta$ por aplicação da regra $\rightarrow$ **I** a uma sub-dedução com premissa φ e conclusão θ. Sejam $\varphi_1, \ldots, \varphi_k$ as premissas assumidas em $\varphi \rightarrow \theta$. Em θ as premissas são algumas das $\varphi_1, \ldots, \varphi_k$ e φ. E φ é consequência semântica dessas premissas.



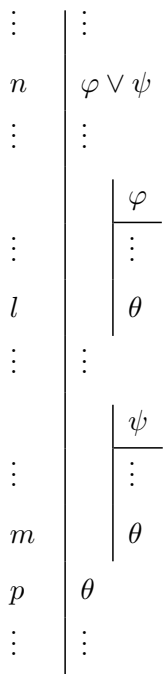
Seja a tabela de verdade para $\varphi_1, \ldots, \varphi_k$, φ, $\varphi \rightarrow \theta$ e θ. Por hipótese, existe uma valoração v tal que $v(\varphi_i) = \mathbf{V}$, $1 \leq i \leq k$ e $v(\varphi \rightarrow \theta) = \mathbf{F}$. Então temos também que $v(\varphi) = \mathbf{V}$ e $v(\theta) = \mathbf{F}$. Mas isto *contradiz* o facto de θ ser consequência semântica de $\varphi_1, \ldots, \varphi_k$ e φ. Logo esta regra também não pode ser aplica no passo p.

Suponhamos que a regra a aplicar é a eliminação da disjunção:

$\vee$**E**

$$\dfrac{\varphi\vee\psi \qquad \begin{array}{cc}[\varphi] & [\psi]\\ \vdots & \vdots\\ \gamma & \gamma\end{array}}{\gamma}$$

: Suponhamos que o passo p deduz θ por aplicação da regra $\vee$**E** a $\varphi \vee \psi$ e duas sub-deduções uma com premissa φ outra com premissa φ e ambas com conclusão θ. Sejam $\varphi_1, \ldots, \varphi_k$ as premissas assumidas em θ (no passo p) e por, hipótese, θ não é consequência semântica delas. Mas $\varphi \vee \psi$ é consequência semântica de algumas delas.



Seja a tabela de verdade para $\varphi_1, \ldots, \varphi_k$, $\varphi \vee \psi$, e θ. Por hipótese, existe uma valoração v tal que $v(\varphi_i) = \mathbf{V}$, $1 \leq i \leq k$, $v(\varphi \vee \psi) = \mathbf{V}$ e $v(\theta) = \mathbf{F}$. Então também, ou $v(\varphi) = \mathbf{V}$ ou $v(\psi) = \mathbf{V}$. Ambos os casos obrigavam a $v(\theta) = \mathbf{V}$, no passo l ou no passo m respectivamente, uma vez que nesses passos θ é consequência semântica de $\varphi_1, \ldots, \varphi_k$, e φ ou ψ. Mas isto *contradiz* a hipótese de se ter $v(\theta) = \mathbf{F}$.

Vamos apenas considerar mais um caso. Suponhamos que se aplica a regra da eliminação de $\mathbf{F}$:

$\mathbf{F}\mathbf{E}$: $\dfrac{\mathbf{F}}{\varphi}$ Suponhamos que no passo p se deduz φ de $\mathbf{F}$. Sejam $\varphi_1, \ldots, \varphi_k$ as premissas assumidas em φ, que são as premissas assumidas em $\mathbf{F}$ (e das quais $\mathbf{F}$ é consequência semântica). Mas isto só pode ser se $\varphi_1, \ldots, \varphi_k$ forem todas contradições. E portanto φ é (vacuosamente) consequência semântica de $\varphi_1, \ldots, \varphi_k$.

Depois de analisados os restantes casos e em todos obtermos uma contradição, podemos concluir que uma dedução no sistema DN não pode ter passos que não sejam consequência semântica das premissas. □

Corolário 1.3. *Se* $\vdash \varphi$ *então* $\models \varphi$.

A integridade da dedução natural permite-nos determinar se não existe uma dedução de uma fórmula ψ a partir de premissas $\varphi_1, \ldots, \varphi_n$ (i.e $\varphi_1, \ldots, \varphi_n \nvdash \psi$): basta encontrar uma valoração v tal que $v(\varphi_i) = \mathbf{V}$, $1 \leq i \leq n$ e $v(\psi) = \mathbf{F}$ (i.e que $\varphi_1, \ldots, \varphi_n \not\models \psi$). Nota, contudo, que a

integridade não permite concluir que se ψ é consequência semântica de $\varphi_1, \ldots, \varphi_n$ então existe uma dedução...isso é a completude.

Exercício 1.20. *Mostra que* $\neg p \vee (q \to p) \nvdash \neg p \wedge q$. $\diamond$

Exercício 1.21. *Termina a demonstração da proposição 1.8.* $\diamond$

1.5.2 Completude do sistema de dedução natural DN

Vamos ver que a dedução natural DN é completa para a lógica proposicional: qualquer consequência semântica pode ser deduzida em DN; em particular todas as tautologias são teoremas de DN.
Seja v uma atribuição de valores às variáveis. Para cada fórmula ψ, define-se

$$\psi^v = \begin{cases} \psi & \text{se } v(\psi) = \mathbf{V} \\ \neg\psi & \text{se } v(\psi) = \mathbf{F} \end{cases}$$

Lema 1.5. *Seja φ uma fórmula cujas variáveis proposicionais são $q_1, \ldots, q_n$, e seja v uma atribuição de valores às variáveis. Então*

$$q_1^v, \ldots, q_n^v \vdash \varphi^v$$

Por exemplo, seja $p \wedge q$, $v(p) = \mathbf{V}$ e $v(q) = \mathbf{F}$. Temos que $p^v = p$, $q^v = \neg q$ e $(p \wedge q)^v = \neg(p \wedge q)$. Então pelo lema, vem que $p, \neg q \vdash \neg(p \wedge q)$.

Demonstração. Por indução estrutural na fórmula φ (= no número de conectivas que ocorrem em φ):

$\varphi = q_1$ Então é claro que $q_1^v \vdash q_1^v$

$\varphi = \neg\varphi_1$ e tem-se que $q_1^v, \ldots, q_n^v \vdash \varphi_1^v$ por hipótese de indução. Se $v(\varphi) = \mathbf{V}$, então $v(\varphi_1) = \mathbf{F}$, donde $\varphi^v = \neg\varphi_1 = \varphi_1^v$ e portanto $q_1^v, \ldots, q_n^v \vdash \varphi^v$. Caso contrário, $v(\varphi) = \mathbf{F}$, então $v(\varphi_1) = \mathbf{V}$, então $\varphi_1^v = \varphi_1$ e $\varphi^v = \neg\neg\varphi_1$. Podemos estender a dedução de $q_1^v, \ldots, q_n^v \vdash \varphi_1$ usando a regra $\neg\neg\mathbf{I}$ e obtemos uma dedução $q_1^v, \ldots, q_n^v \vdash \neg\neg\varphi_1 = \varphi^v$.

$\varphi = \varphi_1 \circ \varphi_2$ onde $\circ$ pode ser $\wedge$, $\vee$ ou $\to$. Sejam $p_1, \ldots, p_l$ e $r_1, \ldots, r_k$, respectivamente as variáveis proposicionais que ocorrem em φ_1 e φ_2, e $\{q_1, \ldots, q_n\} = \{p_1, \ldots, p_l\} \cup \{r_1, \ldots, r_k\}$. De $p_1^v, \ldots, p_l^v \vdash \varphi_1^v$ e $r_1^v, \ldots, r_k^v \vdash \varphi_2^v$ podemos deduzir, usando a regra $\wedge\mathbf{I}$:

$$q_1^v, \ldots, q_n^v \vdash \varphi_1^v \wedge \varphi_2^v$$

donde temos de deduzir φ^v.

$\varphi = \varphi_1 \to \varphi_2$ Se $v(\varphi) = \mathbf{F}$, então $v(\varphi_1) = \mathbf{V}$ e $v(\varphi_2) = \mathbf{F}$. Então $\varphi_1^v = \varphi_1$ e $\varphi_2^v = \neg\varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \neg\varphi_2$. Para ter $q_1^v, \ldots, q_n^v \vdash \neg(\varphi_1 \to \varphi_2)$, basta que $\varphi_1 \wedge \neg\varphi_2 \vdash \neg(\varphi_1 \to \varphi_2)$ (mostra!).

Se $v(\varphi) = \mathbf{V}$, temos 3 casos:

- $v(\varphi_1) = v(\varphi_2) = \mathbf{F}$. Então $\varphi_1^v = \neg\varphi_1$ e $\varphi_2^v = \neg\varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \neg\varphi_1 \wedge \neg\varphi_2$. Para ter $q_1^v, \ldots, q_n^v \vdash \varphi_1 \to \varphi_2$, basta que $\neg\varphi_1 \wedge \neg\varphi_2 \vdash \varphi_1 \to \varphi_2$ (mostra!).
- $v(\varphi_1) = \mathbf{F}$ e $v(\varphi_2) = \mathbf{V}$. Então $\varphi_1^v = \neg\varphi_1$ e $\varphi_2^v = \varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \neg\varphi_1 \wedge \varphi_2$. Para ter $q_1^v, \ldots, q_n^v \vdash \varphi_1 \to \varphi_2$, basta que $\neg\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \to \varphi_2$ (mostra!)
- $v(\varphi_1) = v(\varphi_2) = \mathbf{V}$. Então $\varphi_1^v = \varphi_1$ e $\varphi_2^v = \varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \varphi_2$. Para ter $q_1^v, \ldots, q_n^v \vdash \varphi_1 \to \varphi_2$, basta que $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \to \varphi_2$ (mostra!)

$\varphi = \varphi_1 \wedge \varphi_2$ Temos 4 casos:

- $v(\varphi_1) = v(\varphi_2) = \mathbf{V}$. Então $\varphi_1^v = \varphi_1$ e $\varphi_2^v = \varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \varphi_2$ e então $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \varphi_2 = \varphi^v$
- $v(\varphi_1) = \mathbf{F}$ e $v(\varphi_2) = \mathbf{V}$. Então $\varphi_1^v = \neg\varphi_1$ e $\varphi_2^v = \varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \neg\varphi_1 \wedge \varphi_2$. Para que $q_1^v, \ldots, q_n^v \vdash \neg(\varphi_1 \wedge \varphi_2) = \varphi^v$ basta que $\neg\varphi_1 \wedge \varphi_2 \vdash \neg(\varphi_1 \wedge \varphi_2)$ (mostra!)
- $v(\varphi_1) = \mathbf{V}$ e $v(\varphi_2) = \mathbf{F}$. Então $\varphi_1^v = \varphi_1$ e $\varphi_2^v = \neg\varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \neg\varphi_2$. Para que $q_1^v, \ldots, q_n^v \vdash \neg(\varphi_1 \wedge \varphi_2) = \varphi^v$ basta que $\varphi_1 \wedge \neg\varphi_2 \vdash \neg(\varphi_1 \wedge \varphi_2)$ (mostra!)
- $v(\varphi_1) = v(\varphi_2) = \mathbf{F}$. Então $\varphi_1^v = \neg\varphi_1$ e $\varphi_2^v = \neg\varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \neg\varphi_1 \wedge \neg\varphi_2$. Para o $q_1^v, \ldots, q_n^v \vdash \neg(\varphi_1 \wedge \varphi_2) = \varphi^v$ basta que $\neg\varphi_1 \wedge \neg\varphi_2 \vdash \neg(\varphi_1 \wedge \varphi_2)$ (mostra!)

$\varphi = \varphi_1 \vee \varphi_2$ Temos 4 casos:

- $v(\varphi_1) = v(\varphi_2) = \mathbf{F}$. Então $\varphi_1^v = \neg\varphi_1$ e $\varphi_2^v = \neg\varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \neg\varphi_1 \wedge \neg\varphi_2$. Para que $q_1^v, \ldots, q_n^v \vdash \neg(\varphi_1 \vee \varphi_2) = \varphi^v$ basta que $\neg\varphi_1 \wedge \neg\varphi_2 \vdash \neg(\varphi_1 \vee \varphi_2)$ (mostra!)
- $v(\varphi_1) = v(\varphi_2) = \mathbf{V}$. Então $\varphi_1^v = \varphi_1$ e $\varphi_2^v = \varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \varphi_2$. Para que $q_1^v, \ldots, q_n^v \vdash \varphi_1 \vee \varphi_2 = \varphi^v$ basta que $\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \vee \varphi_2$ (mostra!)
- $v(\varphi_1) = \mathbf{F}$ e $v(\varphi_2) = \mathbf{V}$. Então $\varphi_1^v = \neg\varphi_1$ e $\varphi_2^v = \varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \neg\varphi_1 \wedge \varphi_2$. Para que $q_1^v, \ldots, q_n^v \vdash \varphi_1 \vee \varphi_2 = \varphi^v$ basta que $\neg\varphi_1 \wedge \varphi_2 \vdash \varphi_1 \vee \varphi_2$ (mostra!)
- $v(\varphi_1) = \mathbf{V}$ e $v(\varphi_2) = \mathbf{F}$. Então $\varphi_1^v = \varphi_1$ e $\varphi_2^v = \neg\varphi_2$, e $q_1^v, \ldots, q_n^v \vdash \varphi_1 \wedge \neg\varphi_2$. Para o $q_1^v, \ldots, q_n^v \vdash \varphi_1 \vee \varphi_2 = \varphi^v$ basta que $\varphi_1 \wedge \neg\varphi_2 \vdash \varphi_1 \vee \varphi_2$ (mostra!)

□

Proposição 1.9. *Se $\models \varphi$ então $\vdash \varphi$, i.e, se φ é uma tautologia então φ é um teorema.*

Proposição 1.10. *Se* $\varphi_1, \ldots, \varphi_n \models \varphi$ *então* $\varphi_1, \ldots, \varphi_n \vdash \varphi$.

Demonstração. **Prop.1.9**
Seja φ uma tautologia e $p_1, \ldots, p_n$ as variáveis proposicionais que nela ocorrem. De $\models \varphi$ e pelo lema 1.5, $v(\varphi) = \mathbf{V}$ e $p_1^v, \ldots, p_n^v \vdash \varphi$, para toda a valoração v. Tem-se então que:

$$p_1^v, \ldots, p_{n-1}^v, p_n \vdash \varphi \text{ e } p_1^v, \ldots, p_{n-1}^v, \neg p_n \vdash \varphi$$

para toda a valoração v. Usando **TE** para $p_n \vee \neg p_n$ e a regra $\vee$**E**, podemos combinar as duas deduções anteriores numa de:

$$p_1^v, \ldots, p_{n-1}^v \vdash \varphi$$

Esquematicamente temos:

$$\begin{array}{|l}
p_1^v \\
\vdots \\
p_{n-1}^v \\
\hline
p_n \vee \neg p_n \qquad \textbf{TE} \\
\quad \begin{array}{|l} p_n \\ \hline \boxed{\text{A}} \\ \varphi \end{array} \\
\quad \begin{array}{|l} \neg p_n \\ \hline \boxed{\text{B}} \\ \varphi \end{array} \\
\varphi \qquad \vee\text{E}
\end{array}$$

$$\begin{array}{|l}
p_1^v \\
\vdots \\
p_n \\
\hline
\boxed{\text{A}} \\
\varphi
\end{array}$$

$$\begin{array}{|l}
p_1^v \\
\vdots \\
\neg p_n \\
\hline
\boxed{\text{B}} \\
\varphi
\end{array}$$

$p_1^v, \ldots, p_{n-1}^v, p_n \vdash \varphi$ $\quad$ $p_1^v, \ldots, p_{n-1}^v, \neg p_n \vdash \varphi$ $\quad$ $p_1^v, \ldots, p_{n-1}^v \vdash \varphi$

Repetindo o processo $n-1$ vezes para $p_1, \ldots, p_{n-1}$ obtemos $\vdash \varphi$ □

Lema 1.6. $\varphi_1, \ldots, \varphi_n \models \psi$ *se e só se* $\models (\varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots)))$.

Demonstração. Por contradição. Para qualquer valoração $v((\varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots)))) = \mathbf{F}$ se $v(\varphi_i) = \mathbf{V}$ para todos $1 \leq i \leq n$ e $v(\psi) = \mathbf{F}$ (mostra!). Mas isto contradiz $\varphi_1, \ldots, \varphi_n \models \psi$! □

Demonstração. (**Prop. 1.10**) Pelo lema 1.6, $\models (\varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots)))$. Pela proposição 1.9, $\vdash (\varphi_1 \to (\varphi_2 \to (\ldots(\varphi_n \to \psi)\ldots)))$. E pelo lema 1.4 (da dedução, página 41), $\varphi_1, \ldots, \varphi_n \vdash \psi$ □

O resultado anterior também se verifica para o caso Σ ser infinito e portanto juntando a integridade e a completude, temos para qualquer conjunto de fórmulas:

$$\Sigma \models \varphi \quad \text{se e só se} \quad \Sigma \vdash \varphi$$

Leituras suplementares [BE00] (Cap. 8.3)

1.6 Decidibilidade da lógica proposicional

Proposição 1.11. *Dado um conjunto de fórmulas Σ e uma fórmula φ é decidível se $\Sigma \vdash \varphi$, i.e, existe um algoritmo que determina se φ é dedutível de Σ.*

Demonstração. Pela completude e integridade, decidir $\Sigma \vdash \varphi$ equivale a decidir se $\Sigma \models \varphi$. E podemos supor $\Sigma = \emptyset$ (usando a lema 1.4(da dedução)). Então basta construir a tabela de verdade para φ (que é finita) e verificar se φ é uma tautologia. □

Corolário 1.4. *É decidível se uma fórmula φ é um teorema (= é válida).*

Corolário 1.5. *É decidível se uma fórmula φ é satisfazível.*

Demonstração. A fórmula φ é satisfazível se e só se $\neg\varphi$ não é uma tautologia. □

No entanto, utilizar o método das tabelas de verdade para determinar se uma fórmula é satisfazível é um algoritmo pouco eficiente que tem complexidade temporal exponencial. Mas, embora, existam outros algoritmos, até ao momento não se conhece nenhum com complexidade temporal polinomial (este facto está relacionado com a conjectura $P = NP$, como poderão ver em disciplinas de Complexidade).

1.7 Outros sistemas dedutivos

1.7.1 Sistemas dedutivos de Hilbert, H

Usados inicialmente nas tentativas de mecanização das demonstrações matemáticas (Séc. XIX e início de XX) (também usados por Peano, G. Frege e B. Russel)

Supondo apenas o conjunto completo de conectivas $\{\neg, \rightarrow\}$, pode tomar a forma:

Axiomas

- $\varphi \rightarrow (\psi \rightarrow \varphi)$
- $(\varphi \rightarrow (\psi \rightarrow \theta)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \theta))$

- $(\neg\psi \to \neg\varphi) \to ((\neg\psi \to \varphi) \to \psi)$

Regras de inferência

- *Modus ponens*: de φ e de $\varphi \to \psi$, inferir ψ

Proposição 1.12. *(da dedução) Se* $\Sigma \cup \{\psi\} \vdash_H \theta$ *então* $\Sigma \vdash_H \psi \to \theta$.

Proposição 1.13. $\Sigma \vdash_{ND} \varphi$ *se e só se* $\Sigma \vdash_H \varphi$

Demonstração. ($\Leftarrow$): Basta ver que os axiomas de H são teoremas de DN (ver exercício 1.21). A regra de inferência corresponde à regra da eliminação de implicação de DN ($\to \mathbf{E}$)
($\Rightarrow$): é possível transformar uma dedução em DN, numa dedução em H. (Não o faremos neste curso...) □

Corolário 1.6. *O sistema de dedução* H *é integro e completo para a lógica proposicional.*

A diferença entre os dois sistemas é a impossibilidade de se fazer novas suposições (assumir novas premissas) no sistema H: isso torna as deduções neste sistema mais difíceis e menos automatizáveis...

1.7.2 Sistemas dedutivos analíticos

Mesmo no sistema DN não é fácil construir um algoritmo para determinar se uma fórmula é um teorema, excepto se se usar a completude (e construir a dedução a partir da tabela de verdade). Mas existem outros sistemas de dedução automatizáveis e em que para determinar se uma fórmula φ é um teorema não é necessária a semântica. Estes sistemas dizem-se *analíticos* (ou automatizáveis). A propriedade essencial destes sistemas é que, partindo da fórmula que se pretende deduzir (i.e considerando a dedução da conclusão para as premissas), em cada passo duma dedução, as fórmulas são sub-fórmulas de alguma fórmula de um passo "anterior". É fácil ver que a regra de *Modus ponens* não verifica esta propriedade.
Entre os sistemas analíticos, destacamos:

***Sequentes* de Gentzen** Variante do sistema de dedução natural.

***Tableaux* analíticos (ou semânticos)** Para deduzir φ, deduz-se que a fórmula $\neg\varphi$ é uma contradição.

Resolução também por contradição, mas necessita de $\neg\varphi$ em forma normal conjuntiva.

Um sistema dedutivo diz-se de *refutação* se para deduzir φ de Σ, se deduz **F** de $\Sigma \cup \{\neg\varphi\}$. A consequência semântica é preservada: se $\Sigma \models \varphi$ sse $\Sigma \cup \{\neg\varphi\}$ não é satisfazível. Tanto os *tableaux* como a resolução são sistemas de refutação.

1.7.3 *Tableaux* semânticos

Para deduzir uma fórmula φ, inicia-se com $\neg\varphi$ e produz-se uma contradição. Uma dedução é uma árvore em que os nós são etiquetados por fórmulas, sendo a raíz da árvore etiquetada por $\neg\varphi$. Em cada passo, expande-se uma folha da árvore de acordo com as regras de expansão dos *Tableaux* (Tabela 1.2). Nestas regras utiliza-se a *notação uniforme* (de R.M. Smullyan [Smu95]), que permite agrupar as fórmulas da forma $\varphi \circ \psi$ e $\neg(\varphi \circ \psi)$ em duas categorias: as *conjuntivas*, α, e as *disjuntivas*, β. Para cada fórmula α (ou β), associam-se duas componentes que denotamos por α_1 e α_2 (β_1 ou β_2). Na Tabela 1.1 apresenta-se os valores destas componentes para $\circ \in \{\wedge, \vee, \rightarrow\}$.

α	α_1	α_2	β	β_1	β_2
$\varphi \wedge \psi$	φ	ψ	$\neg(\varphi \wedge \psi)$	$\neg\varphi$	$\neg\psi$
$\neg(\varphi \vee \psi)$	$\neg\psi$	$\neg\varphi$	$\varphi \vee \psi$	φ	ψ
$\neg(\varphi \rightarrow \psi)$	φ	$\neg\psi$	$\varphi \rightarrow \psi$	$\neg\varphi$	ψ

Tabela 1.1: Notação uniforme: fórmulas α e β

É fácil ver que:

Proposição 1.14. *Para toda a atribuição de valores às variáveis v, e para todas as fórmulas α e β:*

$$\begin{aligned} v(\alpha) &= v(\alpha_1) \wedge v(\alpha_2) \\ v(\beta) &= v(\beta_1) \vee v(\beta_2). \end{aligned}$$

As regras de expansão de *tableaux* permitem transformar uma árvore $\mathbf{T}$ etiquetada por fórmulas, noutra árvore $\mathbf{T}^\star$. Suponhamos que num ramo r da árvore ocorre uma fórmula não literal ψ. Se ψ é $\neg\neg\varphi$, podemos expandir a folha desse ramo com mais um nó etiquetado por φ. Análogamente se ψ é $\neg\mathbf{F}$, adiciona-se $\mathbf{V}$ ou se é $\neg\mathbf{V}$, adiciona-se $\mathbf{F}$. Se ψ é um α, adicionam-se a r dois nós, um etiquetado com α_1 e outro com α_2 (filho de α_1). Se ψ é um β, adiciona-se à folha de r dois filhos, um etiquetado por β_1 outro por β_2.

$\frac{\neg\neg\varphi}{\varphi}$	$\frac{\neg\mathbf{F}}{\mathbf{V}}$	$\frac{\neg\mathbf{V}}{\mathbf{F}}$	$\frac{\alpha}{\begin{array}{c}\alpha_1\\ \alpha_2\end{array}}$	$\frac{\beta}{\beta_1 \mid \beta_2}$

Tabela 1.2: Regras de expansão dos *tableaux*

Um *tableau* pode ser definido indutivamente, como na definição seguinte.

Definição 1.17. *Seja* $\{\varphi_1, \ldots, \varphi_n\}$ *um conjunto de fórmulas.*

1. *Um* tableau *para* $\{\varphi_1, \ldots, \varphi_n\}$ *é uma árvore de um só ramo:*

$$\begin{array}{c} \varphi_1 \\ \varphi_2 \\ \vdots \\ \varphi_n \end{array}$$

2. *Se* $\mathbf{T}$ *é um* tableau *para* $\{\varphi_1, \ldots, \varphi_n\}$ *e* $\mathbf{T}^\star$ *resulta de* $\mathbf{T}$ *por aplicação duma regra de expansão de* tableaux*, então* $\mathbf{T}^\star$ *é um* tableau *para* $\{\varphi_1, \ldots, \varphi_n\}$.

Um *tableau* para o conjunto $\{p \wedge (\neg q \vee \neg p)\}$ é:

$$\begin{array}{ccc} & p \wedge (\neg q \vee \neg p) & \\ & p & \\ & \neg q \vee \neg p & \\ \neg q & & \neg p \end{array}$$

Definição 1.18. *Um ramo* r *de um* tableau *diz-se* fechado *se existe uma fórmula* φ *tal que* φ *e* $\neg\varphi$ *ocorrem em* r*. Um* tableau *diz-se* fechado *se todos os seus ramos estão fechados.*

O *tableau* do exemplo anterior tem, um ramo fechado mas não é fechado.

Definição 1.19. *Uma dedução por* tableau *de* φ *é um* tableau *fechado para* $\{\neg\varphi\}$*. A fórmula* φ *é um teorema se* φ *tem uma dedução por* tableau*.*

Definição 1.20. *Um ramo* r *de um* tableau *é* satisfazível *se o conjunto de fórmulas que etiquetam os seus nós é satisfazível. Um* tableau $\mathbf{T}$ *é satisfazível se pelo menos um dos seus ramos é satisfazível.*

A proposição seguinte demonstra-se por análise de casos e usando a Proposição 1.14.

Proposição 1.15. *Pela aplicação de qualquer regra de expansão de* tableau *a um* tableau *satisfazível, obtêm-se um* tableau *satisfazível.*

Proposição 1.16. *Se existe um* tableau *fechado para um conjunto de fórmulas* Γ*, então* Γ *não é satisfazível.*

Temos, então, a integridade do sistema de *tableaux*:

Teorema 1.1. *Se* φ *tem uma dedução por* tableau*, então* φ *é uma tautologia.*

Demonstração. Uma dedução por *tableau* é um *tableau* fechado para $\{\neg\varphi\}$. Então, pela Proposição 1.16, $\{\neg\varphi\}$ não é satisfazível, logo φ é uma tautologia. □

Apresentamos, sem demonstração, a completude do sistema de *tableaux*:

Teorema 1.2. *Se φ é uma tautologia, φ tem uma dedução por* tableau.

Exercício 1.22. *Constrói deduções de* tableaux *para as seguintes fórmulas:*

a) $\neg((p \vee q) \wedge (\neg p \wedge \neg q))$

b) $((p \to q) \wedge (q \to r)) \to \neg(\neg r \wedge p)$

c) $(\neg p \to q) \to ((p \to q) \to q)$

d) $((p \to q) \to p) \to p$

◇

1.7.4 Resolução

Recorda que um *literal* é uma fórmula atómica ou a sua negação: p, $\neg p$. Uma *cláusula* é uma disjunção de literais: $p \vee \neg q \vee \neg p \vee s$ e pode representar-se pelo conjunto $\{p, \neg q, \neg p, s\}$. Então uma fórmula da lógica proposicional em FNC, p.e.

$$\neg p \wedge (q \vee r \vee q) \wedge (\neg r \vee \neg s) \wedge (p \vee s) \wedge (\neg q \vee \neg s)$$

pode ser vista como um conjunto de cláusulas:

$$\{\{\neg p\}, \{q, r\}, \{\neg r, \neg s\}, \{p, s\}, \{\neg q, \neg s\}\}$$

Seja Σ um conjunto de cláusulas e representamos por **F** a cláusula vazia.
O sistema dedutivo por *resolução* não tem axiomas e apenas uma regra de inferência (de *resolução*):

$$\frac{C \cup \{p\} \qquad C' \cup \{\neg p\}}{C \cup C'}$$

e a conclusão diz-se a *resolvente* das premissas.
A dedução de **F** a partir de um conjunto $\mathcal{C}$ de cláusulas diz-se uma refutação $\mathcal{C}$.
No caso anterior, tinha-se a seguinte dedução:

$$\dfrac{\dfrac{\{p,s\}\ \{\neg p\}}{\{s\}}\ \dfrac{\dfrac{\{q,r\}\ \{\neg r,\neg s\}}{\{q,\neg s\}}\ \{\neg q,\neg s\}}{\{\neg s\}}}{\mathbf{F}}$$

$$\dfrac{\dfrac{\{p,s\}\ \{\neg p\}}{\{s\}}\ \dfrac{\dfrac{\{q,r\}\ \{\neg r,\neg s\}}{\{q,\neg s\}}\ \{\neg q,\neg s\}}{\{\neg s\}}}{\mathbf{F}}$$

Uma *dedução por resolução de* uma fórmula φ é uma refutação de cláusulas que correspondem à FNC de $\neg\varphi$.

Teorema 1.3. *(Integridade) Seja $\mathcal{C} = \{C_1, \ldots, C_n\}$ um conjunto não vazio de cláusulas da lógica proposicional.*

i. Sejam C_i e C_j cláusulas de $\mathcal{C}$ e R uma resolvente de C_i e C_j. Então $\mathcal{C} \models R$.

ii. Se $\mathcal{C} \vdash_R \mathbf{F}$, isto é, existe uma dedução de $\mathbf{F}$ a partir de $\mathcal{C}$ usando apenas a regra da Resolução, então $\mathcal{C}$ não é satisfazível.

Exercício 1.23. *Mostra o teorema 1.3.* ◇

Teorema 1.4. *(Completude) Se um conjunto de cláusulas C é não satisfazível então existe uma dedução $\mathcal{C} \vdash_R \mathbf{F}$.*

Exercício 1.24. *Considera a fórmula seguinte em forma normal conjuntiva (FNC):*

$$(p \vee \neg q \vee r) \wedge \neg p \wedge (q \vee r \vee p) \wedge (p \vee \neg r)$$

i. Converte a fórmula para um conjunto de cláusulas.

ii. A partir desse conjunto constrói uma dedução de $\mathbf{F}$, usando apenas a regra da resolução.

◇

Exercício 1.25. *Encontra uma dedução de $\mathbf{F}$ (uma refutação) para:*

$$\{\{\neg p, \neg q, r\}, \{p, r\}, \{q, r\}, \{\neg r\}\}$$

◇

Exercício 1.26. *Encontra uma dedução por resolução de:* $((p \to q) \wedge (q \to r)) \to \neg(\neg r \wedge p)$

◇

Exercício 1.27. *Constrói deduções de* tableau *e por* ***resolução*** *para as seguintes fórmulas:*

a) $(p \to q) \to ((\neg p \to q) \to q)$

b) $q \to (p \to (q \wedge p))$

c) $q \to (\neg p \to \neg(q \to p))$

◇

Leituras suplementares [Bro00], [Fit90], [Smu95], [GLM97], [BA01].

Capítulo 2

Lógica de primeira ordem

Na lógica proposicional não é possível representar adequadamente frases como:

- Todos os pássaros têm penas
- Alguns pássaros cantam
- Nem todos os pássaros voam
- Todos os homens são mortais
- Todas as crianças são mais novas que os seus pais
- Um inteiro ou é par ou ímpar
- Todos os pássaros são mais leves que algum mamífero

Por outro lado, proposições como as seguintes podem-se representar usando relações n-árias entre objectos:

Proposição	Representação
O Carlos é irmão da Joana	`irmao(carlos,joana)`
A Joana deu um livro ao Carlos	`dar(joana,livro,carlos)`
2 é par	`par(2)`
$2 = 1 + 1$	`igual(2,soma(1,1))`

Na lógica de primeira ordem vai ser possível representar:

Objectos: Termos que podem ser simples (constantes ou variáveis) ou complexos. Ex: `joana`, `x`, `mãe(joana)`, `soma(1,1)`

Propridades ou relações entre objectos: Fórmulas que podem ser predicados simples ou complexos (usando conectivas). Ex: `dar(joana,livro,carlos)`,`par(2)` $\wedge$ $\neg$`par(3)`

Propriedades ou relações entre conjuntos de objectos Fórmulas quantificadas sobre objectos. Os quantificadores permitem representar as palavras `todos`, `alguns`, etc:

$\forall$ (universal) e $\exists$ (existêncial).

Ex:

$\forall$`x` (`passaro(x)` $\rightarrow$ `penas(x)`)

$\exists$`x` (`passaro(x)` $\wedge$ `canta(x)`)

$\exists$`x` (`passaro(x)` $\wedge$ $\neg$`voa(x)`)

$\forall$`x` (`passaro(x)` $\rightarrow$ ($\exists$`y`(`mamifero(y)` $\wedge$ `mais_leve(x,y)`))))

2.1 Linguagens da lógica de primeira ordem

Uma linguagem $\mathcal{L}$ de lógica de 1ª ordem é caracterizada pelos seguintes conjuntos de símbolos:

Símbolos lógicos que estão presentes em qualquer linguagem:

- um conjunto numerável de *variáveis*, $Var = \{x, y, \ldots, x_0, y_0, \ldots\}$;
- as *conectivas lógicas* $\wedge$, $\vee$, $\neg$ e $\rightarrow$;
- os *quantificadores* $\forall$ (universal) e $\exists$ (existencial);
- os parêntesis (e);
- possivelmente o *símbolo de igualdade*, $=$.

Símbolos não lógicos que caracterizam a linguagem:

- um conjunto, possivelmente vazio, de *símbolos funcionais* n-ários para cada $n \geq 0$, $\mathcal{F}_n$; os elementos de $\mathcal{F}_0$ chamam-se *constantes* e representam-se por a, b, c,.... Os restantes símbolos por f, g, h, ...;
- um conjunto, possivelmente vazio, de *símbolos de predicado* (ou relacionais) n-ários para cada $n \geq 0$, $\mathcal{R}_n$. Os símbolos relacionais representam-se por P, R, Q, etc...

Definição 2.1 (Termos)**.** *Seja $\mathcal{L}$ uma linguagem de 1ª ordem. Um* termo*, é uma sequência de símbolos de $\mathcal{L}$, representado por $t, s, \ldots t_1, s_1, \ldots$ e definido indutivamente por:*

- *Uma variável x é um termo;*

- *Uma constante a é um termo;*
- *Se $t_1, \ldots, t_n$ são termos e $f \in \mathcal{F}_n$ um símbolo funcional de aridade $n > 0$, então $f(t_1, \ldots, t_n)$ é um termo .*

O conjunto dos termos $\mathcal{T}_{\mathcal{L}}$ também pode ser definido pela seguinte gramática independente de contexto, em notação BNF:

$$t ::= x \mid a \mid f(t, \ldots, t)$$

onde $x \in Var$, $a \in \mathcal{F}_0$ e $f \in \mathcal{F}_n$.
Um termo diz-se *fechado* se não tiver ocorrências de variáveis e denotamos por $\mathcal{T}_O$ o seu conjunto. Ex: a, $f(a, g(b, c))$ e $f(f(a, a)))$

Exemplo 2.1. *Supondo $\mathcal{F}_0 = \{a, d\}$, $\mathcal{F}_2 = \{f, h\}$, $\mathcal{F}_1 = \mathcal{F}_3 = \{g\}$, indica quais das seguintes sequências de símbolos são termos e quais são termos fechados:*

- $f(a, g(x, g(a), a))$
- $g(d, h(y, z), d, f(a)))$
- $h(d, f(a, h(d)))$
- $f(f(x, x), f(y, y))$
- $x(d, g(y))$
- $f(a, a(x))$
- $h(h(h(g(a), d), g(a)), d)$

Escreve cada um deles em forma de árvore sintáctica

O *comprimento* de um termo t é o comprimento da sequência de caracteres que o representa. Por exemplo, $f(a, g(a, a))$ tem comprimento 11.

Exemplo 2.2. *Considerando o Exemplo 2.1 determina todos os termos fechados de $\mathcal{L}$ com menos de comprimento 9 e em que um mesmo símbolo funcional não ocorre mais que uma vez.*

Definição 2.2 (Fórmulas atómicas)**.** *O conjunto de* fórmulas atómicas *duma linguagem $\mathcal{L}$ é dado por:*

- *se $t_1, \ldots, t_n$ são termos e $R \in \mathcal{R}_n$ é um símbolo de predicado n-ário, então $R(t_1, \ldots, t_n)$ é uma* fórmula atómica*;*

- *se $\mathcal{L}$ tiver a igualdade e se t_1 e t_2 são termos então $t_1 = t_2$ é uma* fórmula atómica*;*

Exemplo 2.3. *Exemplos de fórmulas atómicas são:*

- $R(a)$
- $R(x, y, z)$
- $T(f(b, z), a)$
- $R(f(m, x), x, g(a, a), k(x))$.

r

Definição 2.3 (Fórmulas)**.** *O conjunto das* fórmulas *é definido indutivamente por:*

- *uma fórmula atómica é uma* fórmula*;*
- *se φ é uma fórmula $\neg\varphi$ é uma* fórmula*;*
- *se φ e ψ são fórmulas então $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ e $(\varphi \to \psi)$ são* fórmulas
- *se φ é uma fórmula e x uma variável, então $\forall x\ \varphi$ e $\exists x\ \varphi$ são* fórmulas.

O conjunto das fórmulas pode também ser definido por uma gramática, em notação BNF:

$$\varphi ::= R(t_1, \ldots, t_n) \mid t_1 = t_2 \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \to \varphi) \mid \forall x\varphi \mid \exists x\varphi$$

onde $R \in \mathcal{R}_n$, $t_i \in \mathcal{T}$ e $x \in Var$.

Exemplo 2.4. *Sendo $\mathcal{F}_0 = \{m, d\}$, $\mathcal{F}_1 = \{f\}$, $\mathcal{F}_2 = \{g\}$ e $\mathcal{R}_2 = \{R, S\}$, determina quais das seguintes expressões são fórmulas:*

- $S(m, x)$
- $f(m)$
- $R(R(m, x))$
- $(R(x, y) \to (\exists z\ S(z, f(y))))$
- $R(x, y) \to R(\exists z R(z, z)))$
- $\forall x\ R(f(d), g(f(m), y))$
- $\forall x \forall y (g(x, y) \to S(y, x))$

Nas fórmulas da lógica de $1^{\underline{a}}$ ordem, podem-se omitir os parêntesis, supondo que as convenções para a lógica proposicional e que os quantificadores têm maior precedência que qualquer operador lógico. Por exemplo, as seguintes fórmulas são diferentes: $\exists x P(x) \ \vee \ \neg P(x)$ e $\exists x(P(x) \ \vee \ \neg P(x))$.

Exemplo 2.5 (Linguagem para a aritmética)**.** *Pretende-se uma linguagem para exprimir ir fórmulas sobre os números naturais e as operações de adição e multiplicação. Seja $\mathcal{A}$ a linguagem com $=$ caracterizada por:*

$$\mathcal{F}_0 = \{0, 1\},\ \mathcal{F}_2 = \{+, \times\} \text{ e } \mathcal{R}_2 = \{<\}$$

Os termos seguintes representam os naturais:
$0, 1, +(1,1), +(1,+(1,1)), +(1,+(+(1,1),1)) \ldots$
Outros termos são: $(\times(\times(1,0),1),\ (+(0,+(0,1))),\ \ldots$
E exemplos de fórmulas são:
$< (\times(1,1), +(1,1))$
$\forall x\ (+(0,x) = x)$
$\forall x \forall y(+(x,1) = +(y,1) \rightarrow x = y)$
Podemos ainda adoptar a notação usual infixa para os símbolos funcionais e relacionais, e escrever: $(1 \times 1) < (1 + 1)$, $(0 + x)$, *etc...*

Exemplo 2.6. *Como exprimir que um inteiro é par? Primeiro temos de ter a noção de x ser divisível por y:*

$$\exists q\ x = q \times y$$

Então, a fórmula seguinte caracteriza um número par:

$$\exists q\ x = q \times (1 + 1)$$

Da mesma forma se pode exprimir que um número x é primo:

$$\neg(x < 2) \wedge \forall y((\exists q\ x = q \times y) \rightarrow (y = 1 \ \vee \ y = x))$$

Exemplo 2.7 (Tradução de frases para fórmulas)**.** *Na tabela seguinte apresentam-se algumas frases em linguagem natural e a sua representação como fórmulas da lógica de $1^{\underline{a}}$ ordem.*

Frases do tipo	*Representação*
Todos os P's são Q's	$\forall x(P(x) \rightarrow Q(x))$
Alguns P's são Q's	$\exists x(P(x) \ \wedge \ Q(x))$
Nenhum P é Q	$\forall x(P(x) \rightarrow \neg Q(x))$
Nem todos os P's são Q's	$\exists x(P(x) \ \wedge \ \neg Q(x))$

Considerando predicados adicionais, exprime na linguagem da aritmética:

- *Todo o número par é primo*
- *Nem todos os números primos são pares*
- *Alguns primos não são pares*
- *Nenhum primo é par*
- *Todo o primo é não par ou igual a 2*

Definição 2.4 (Variáveis livres e ligadas)**.** *Uma variável x tem uma ocorrência* não livre *(ou ligada) numa fórmula φ se φ tem uma sub-fórmula da forma $\forall x\psi$ ou $\exists x\psi$ e x ocorre em ψ. Caso contrário, diz-se que x* ocorre livre *em φ.*

Uma variável pode ocorrer livre e ligada numa mesma fórmula. Em

$$\forall x((P(x) \to Q(x))) \to (\neg P(x) \ \vee \ Q(y))$$

x tem duas ocorrências ligadas e uma ocorrência livre e y tem apenas uma ocorrência livre.

Definição 2.5 (Proposição)**.** *Uma fórmula sem variáveis livres diz-se uma* proposição.

Por exemplo, $R(a, f(a, b)) \ \wedge \ \forall x(R(x, x) \to \exists y P(y) \ \vee \ P(x))$ é uma proposição.

Exemplo 2.8. *Indica quais as ocorrências livres e ligadas de cada uma das variáveis das fórmulas seguintes. Indica quais as fórmulas que são proposições.*

- $\exists x(P(y, z) \ \wedge \ \forall y(\neg Q(x, y) \ \vee \ P(y, z)))$;
- $\neg(\forall x \exists y P(x, y, z) \ \wedge \ \forall z P(x, y, z))$;
- $\forall x((P(x) \ \wedge \ C(x)) \to \exists y L(x, y))$;
- $P(a, f(a, b))$;
- $\exists x(P(x) \to \neg Q(x))$.

2.2 Semântica da lógica de primeira ordem

Na lógica proposicional a semântica duma fórmula complexa é determinada a partir da semântica dos seus constituintes básicos, i.e, as proposições $p, q \ldots$ cujo valor pode ser **V** ou **F**. Por exemplo, se o valor de p for **V** e o de q, **F**, então a fórmula seguinte tem o valor **V** (verifica!):

$$(p \ \vee \ \neg q) \to (q \to p)$$

Como avaliar uma fórmula de primeira ordem? Por exemplo, como avaliar:

$$\forall x \exists y (P(x) \ \vee \ \neg Q(y)) \rightarrow (Q(x) \rightarrow P(y))$$

Temos que determinar qual o significado

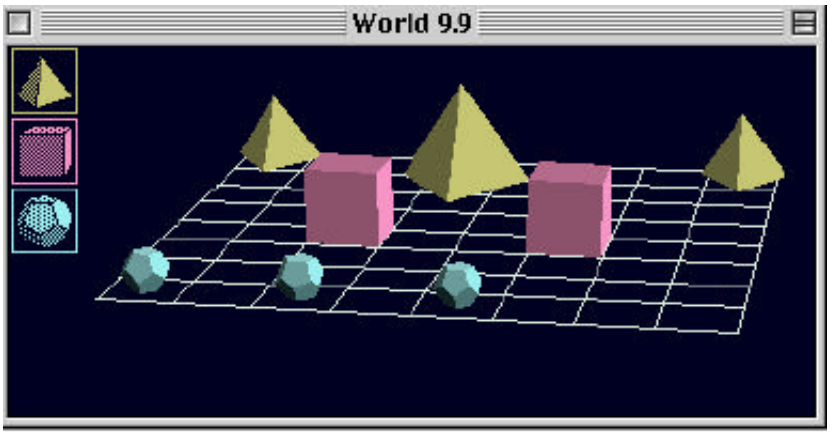
- das variáveis (livres e ligadas)
- dos quantificadores
- dos símbolos funcionais
- dos símbolos relacionais

num dado *universo* ou *domínio de discurso*, por exemplo: os números inteiros, números reais, conjuntos, grafos, objectos geométricos, etc,etc.
Informalmente para os quantificadores e para um dado universo:

$\forall x P(x)$ será verdadeira se e só se $P(x)$ satisfaz *todos* os objectos do universo

$\exists x P(x)$ será verdadeira se e só se $P(x)$ satisfaz *algum* objecto do universo

Exemplo 2.9 (*Mundo dos Blocos*)**.** *Pretende-se descrever um conjunto de blocos com diferentes formas geométricas e tamanhos, colocados num tabuleiro de xadrez...*



Uma linguagem para definir este conjunto pode ser a seguinte:

$$\begin{aligned}
\mathcal{F}_1 &= \{maisafrente, maisaesquerda, maisadireita\} \\
\mathcal{R}_1 &= \{Cubo, Tetra, Dodec, Pequeno, Medio, Grande\} \\
\mathcal{R}_2 &= \{Menor, Maior, Esquerda, Direita, MesmoT, \\
&\quad MesmaF, MesmaL, MesmaC\} \\
\mathcal{R}_3 &= \{Entre\}
\end{aligned}$$

Algumas fórmulas:

- $\exists y\ (Cubo(y)\ \wedge\ Medio(y))$
- $\forall x\ (Cubo(x) \rightarrow \forall y(\neg Cubo(y) \rightarrow Maior(x,y)))$
- $\exists y\ maisaesquerda(y) = y$
- $\forall x\ (Pequeno(x) \rightarrow maisafrente(x) = x)$
- $\forall x\ (maisaesquerda(x) \neq x \rightarrow Cubo(maisaesquerda(x)))$
- $Cubo(x)$
- $MesmoT(z,y)\ \wedge\ MesmaF(z,y)$

Um domínio (conjunto de objectos) irá permitir determinar quais as fórmulas que são satisfeitas nesse "mundo"(estrutura). Para além do domínio é necessário indicar também qual o significado de cada símbolo funcional e cada símbolo relacional.
No mundo (estrutura) $\mathcal{A}$ apresentado na figura:

- *o domínio tem 8 objectos $A = \{o_1, \ldots o_8\}$ (ordenados de cima para baixo e da esquerda pra a direita...)*
- *às constantes não associamos nenhum objecto*
- *a $maisafrente$ associamos uma função $maisafrente^{\mathcal{A}}$ que para cada objecto indica qual o objecto mais à frente na mesma coluna: $maisafrente^{\mathcal{A}}(o_1) = o_6$,$maisafrente^{\mathcal{A}}(o_4) = o_7$, e para os restantes $maisafrente^{\mathcal{A}}(o) = o$. Analogamente associa-se a $maisaesquerda$ e $maisadireita$, a função que para cada objecto indica qual o objecto mais à esquerda, respectivamente, mais à direita.*
- *a $Cubo$ associamos uma relação unária que corresponde aos objectos do domínio que são cubos: $Cubo^{\mathcal{A}} = \{o_4, o_5\}$. Analogamente,*

  $Tetra^{\mathcal{A}} = \{o_1, o_2, o_3\}$

  $Dodec^{\mathcal{A}} = \{o_6, o_7, o_8\}$.

  $Pequeno^{\mathcal{A}} = \{o_6, o_7, o_8\}$

  $Grande^{\mathcal{A}} = \{o_2\}$ *e para os restantes* $o \in Medio^{\mathcal{A}}$

  $Esquerda^{\mathcal{A}} = \{(o_1, o_2), (o_4, o_5), (o_6, o_7), (o_7, o_8)\}$

  $MesmaL^{\mathcal{A}} = (Esquerda^{\mathcal{A}})^{\star}$ *(fecho de reflexivo e transitivo)*

  e de modo análogo se definem as restantes relações binárias.

  $Entre^{\mathcal{A}} = \{(o_7, o_6, o_8)\}$

Como determinar se uma fórmula é satisfazível nesta estrutura?
$\exists y(Cubo(y) \wedge Medio(y))$ *é satisfazível em* $\mathcal{A}$ *pois* $o_4 \in Cubo^{\mathcal{A}}$ *e* $o_4 \in Medio^{\mathcal{A}}$
$\forall x(Dodec(x) \rightarrow Pequeno(x))$ *é satisfazível em* $\mathcal{A}$ *pois para todo* $o \in Dodec^{\mathcal{A}}$, $o \in Pequeno^{\mathcal{A}}$
Mas, $\forall x(Medio(x) \rightarrow Cubo(x))$ *não é satisfazível em* $\mathcal{A}$ *porque* $o_2 \in Medio^{\mathcal{A}}$ *e* $o_2 \notin Cubo^{\mathcal{A}}$
E, $\forall x\ (Pequeno(x) \rightarrow maisafrente(x) = x)$ *é satisfazível em* $\mathcal{A}$*. Porquê?*
E para as fórmulas como $Cubo(x)$ *ou* $MesmoT(z, y) \wedge MesmaF(z, y)$*?*
Neste caso temos ainda de ter primeiro uma interpretação *s para as variáveis livres...*
Para $s(x) = o_4$, $s(y) = o_7$ *e* $s(z) = o_6$*, as duas fórmulas são satisfazíveis, pois* $o_4 \in Cubo^{\mathcal{A}}$ *e* $o_7, o_6 \in MesmoT^{\mathcal{A}} \cap MesmaF^{\mathcal{A}}$

Definição 2.6 (Estrutura duma linguagem)**.** *Uma* estrutura *duma uma linguagem de primeira ordem* $\mathcal{L}$ *é: um par* $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ *onde* A *é um conjunto não vazio, e que se diz o domínio (ou o universo)* $\mathcal{A}$ *e* $.^{\mathcal{A}}$ *uma função tal que:*

- *associa a cada constante* c *um elemento* $c^{\mathcal{A}} \in A$
- *associa a cada símbolo funcional* $f \in \mathcal{F}_n$, $n > 0$ *uma função* n*-ária* $f^{\mathcal{A}}$ *de* A^n *em* A
- *associa a cada símbolo relacional* n*-ário* $R \in \mathcal{R}_n$*, uma relação* $R^{\mathcal{A}} \subseteq A^n$

Definição 2.7 (Interpretação de variáveis)**.** *Dada uma linguagem* $\mathcal{L}$ *e uma estrutura* $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ *de* $\mathcal{L}$*.*
Uma interpretação *das variáveis é uma função* $s : Var \longrightarrow A$*.*
Podemos estender uma interpretação s *ao conjunto dos termos de* $\mathcal{L}$, $s : \mathcal{T} \longrightarrow A$*:*

- *para* $x \in Var$ *o valor de* $s(x)$ *já está definido*
- *para* $c \in \mathcal{F}_0$, $s(c) = c^{\mathcal{A}}$
- *se* $t_1, \ldots, t_n$ *são termos e* $f \in \mathcal{F}_n$, $n \geq 1$,

$$s(f(t_1, \ldots, t_n)) = f^{\mathcal{A}}(s(t_1) \ldots, s(t_n))$$

Se $t \in \mathcal{T}_O$ então $s(t)$ não depende de s e escrevemos $t^{\mathcal{A}}$.

Definição 2.8 (Relação de satisfazibilidade)**.** *Dada uma linguagem* $\mathcal{L}$*, uma estrutura* $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ *de* $\mathcal{L}$ *e uma interpretação* $s : Var \longrightarrow A$*. A relação* $\mathcal{A}$ *satisfaz uma fórmula* φ *para a interpretação* s*, denota-se por* $\mathcal{A} \models_s \varphi$ *é definida por indução na estrutura de* φ*:*

(i) $\mathcal{A} \models_s t_1 = t_2$ *se* $s(t_1) = s(t_2)$

(ii) $\mathcal{A} \models_s R(t_1, \ldots, t_n)$ *se* $(s(t_1), \ldots, s(t_n)) \in R^{\mathcal{A}}$

(iii) $\mathcal{A} \models_s \neg\varphi$ *se* $\mathcal{A} \not\models_s \varphi$ *(i.e não é verdade que* $\mathcal{A} \models_s \varphi$*)*

(iv) $\mathcal{A} \models_s \varphi \wedge \psi$ *se* $\mathcal{A} \models_s \varphi$ *e* $\mathcal{A} \models_s \psi$

(v) $\mathcal{A} \models_s \varphi \vee \psi$ *se* $\mathcal{A} \models_s \varphi$ *ou* $\mathcal{A} \models_s \psi$

(vi) $\mathcal{A} \models_s \varphi \to \psi$ *se* $\mathcal{A} \not\models_s \varphi$ *ou* $\mathcal{A} \models_s \psi$

(vii) $\mathcal{A} \models_s \forall x\, \varphi$ *se para todo o* $a \in A$ *se tem* $\mathcal{A} \models_{s[a/x]} \varphi$ *onde:*

$$s[a/x](y) = \begin{cases} s(y) & \text{se } y \neq x \\ a & \text{se } y = x \end{cases}$$

(viii) $\mathcal{A} \models_s \exists x\, \varphi$ *se existe um* $a \in A$ *tal que tem* $\mathcal{A} \models_{s[a/x]} \varphi$

Exemplo 2.10. *Considera mais uma vez o "Mundo dos Blocos"do Exemplo 2.9. Para verificar que*

$$\mathcal{A} \models_s \exists y(Cubo(y) \wedge Medio(y))$$

por (viii) da Definição 2.8 temos que ver se existe $o \in A$ *tal que* $\mathcal{A} \models_{s[o/y]} (Cubo(y) \wedge Medio(y))$*. Aplicando (iv), se existe* $o \in A$ *tal que* $\mathcal{A} \models_{s[o/y]} Cubo(y)$ *e* $\mathcal{A} \models_{s[o/y]} Medio(y)$*. Sendo* $s[o/y](y) = o$ *e por (ii), se existe* $o \in A$ *tal que* $o \in Cubo^{\mathcal{A}}$ *e* $o \in Medio^{\mathcal{A}}$*. O que é verdade para* o_4*, pelo que a fórmula é satisfeita em* $\mathcal{A}$ *para qualquer interpretação* s*. Do mesmo modo se pode verificar se*

$$\mathcal{A} \models_s \forall x(Dodec(x) \to Pequeno(x)).$$

Por (vii), equivale a ver se para todo o $o \in A$*,* $\mathcal{A} \models_{s[o/x]} (Dodec(x) \to Pequeno(x))$*. Por (vi) fica. para todo* $o \in A$*,* $(A \not\models_{s[o/x]} Dodec(x)$ *ou* $A \not\models_{s[o/x]} Pequeno(x))$*, que equivale a para todo* $o \in A$*,* $(o \notin Dodec^{\mathcal{A}}$ *ou* $o \in Pequeno^{\mathcal{A}})$*. O que é verdade porque* $Dodec^{\mathcal{A}} \subseteq Pequeno^{\mathcal{A}}$*. Finalmente podemos tentar ver se*

$$\mathcal{A} \models_s \forall x(Medio(x) \to Cubo(x)).$$

Por (vii), temos que ver se para todo $o \in A$*,* $\mathcal{A} \models_{s[o/x]} (Medio(x) \to Cubo(x))$*. Ou seja se, para todo* $o \in A$*,* $(\mathcal{A} \not\models_{s[o/x]} Medio(x)$ *ou* $\mathcal{A} \not\models_{s[o/x]} Cubo(x)))$*. Ou ainda, por (ii), se para todo* o $o \in A$*,* $(o \notin Medio^{\mathcal{A}}$ *ou* $o \in Cubo^{\mathcal{A}})$*. Mas isto é falso porque, por exemplo,* $o_1 \in Medio^{\mathcal{A}}$ *e* $o_1 \notin Cubo^{\mathcal{A}}$*. Concluímos que* $\mathcal{A} \not\models_s \forall x(Dodec(x) \to Pequeno(x))$*, isto é a fórmula não é satisfazível em* $\mathcal{A}$ *qualquer que seja a interpretação* s*.*

Exemplo 2.11. *Seja* $\mathcal{L}_N$ *a linguagem de* $1^{\underline{a}}$ *ordem com igualdade para os números naturais já dada:* $\mathcal{F}_0 = \{0, 1\}$*,* $\mathcal{F}_2 = \{+, \times\}$ *e* $\mathcal{R}_2 = \{<\}$
Seja $\mathcal{N} = (\mathbb{N}, \cdot^{\mathcal{N}})$ *uma estrutura de* $\mathcal{L}_N$*, onde* $\cdot^{\mathcal{N}}$ *é definido por:*

- $0^{\mathcal{N}} = 0,\ 1^{\mathcal{N}} = 1$
- $+^{\mathcal{N}}(n,m) = n+m,\ \times^{\mathcal{N}}(n,m) = n \times m$
- $<^{\mathcal{N}} = \{(n,m) \in \mathbb{N}^2 \mid n < m\}$

a) Verificar que, para qualquer interpretação $s : Var \longrightarrow \mathbb{N}$*:*

$\mathcal{N} \models_s \forall x\ < (x, +(x,1))$

Por (vii) da Definição 2.8, temos que ver se para todo o $n \in \mathbb{N}$ *se tem*

$$\mathcal{N} \models_{s[n/x]} < (x, +(x,1))$$

Por (ii) temos:

$$(s[n/x](x), s[n/x](+(x,1)) \in <^{\mathcal{N}}$$

e usando a definição de interpretação (2.7) vem,

$$(n, n+1) \in <^{\mathcal{N}}$$

i.e, $n < n+1$*, o que realmente é verdade para todo o* $n \in \mathbb{N}$*.*

b) Se $s(x) = 2$*, verificar que* $\mathcal{N} \models_s x = +(1,1)$*.*

Neste caso, por (i) da Definição 2.8, temos que ver se

$$s(x) = s(+(1,1))$$

Isto é:

$$2 = +^{\mathcal{N}}(s(1), s(1))$$

ou seja, $2 = 1+1$ *o que é verdade em* $\mathbb{N}$*.*

2.2.1 Satisfazibilidade, validade e consequência

Uma fórmula φ duma linguagem de 1ª ordem é:

satisfazível: se existir uma estrutura $\mathcal{A}$ e uma interpretação s, tal que $\mathcal{A} \models_s \varphi$

válida: se para toda a estrutura $\mathcal{A}$ e toda a interpretação s, se tem $\mathcal{A} \models_s \varphi$. E escreve-se $\models \varphi$

Definição 2.9. *Uma estrutura* $\mathcal{A}$ satisfaz *um conjunto de fórmulas* Γ *para uma interpretação* s*, se satisfizer todas as fórmulas de* Γ*, e escreve-se* $\mathcal{A} \models_s \Gamma$*.*
Uma fórmula φ *é* consequência semântica *de* Γ *se para toda a estrutura* $\mathcal{A}$ *de* $\mathcal{L}$ *e toda a interpretação* s *tal que* $\mathcal{A} \models_s \Gamma$*, se tem* $\mathcal{A} \models_s \varphi$*. E escreve-se* $\Gamma \models \varphi$*.*

Para que uma estrutura satisfaça uma fórmula φ para uma interpretação s, *apenas interessa* o valor que s atribui às variáveis que ocorrem livres em φ. Em particular, temos a seguinte proposição:

Proposição 2.1. *Seja φ uma fórmula duma linguagem $\mathcal{L}$ e $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ uma estrutura de $\mathcal{L}$. Sejam duas interpretações s_1 e s_2 tal que $s_1(x) = s_2(x)$ se x ocorre livre em φ. Então $\mathcal{A} \models_{s_1} \varphi$ se e só se $\mathcal{A} \models_{s_2} \varphi$*

Demonstração. Ideia: mostrar por indução na estrutura dos termos que então $s_1(t) = s_2(t)$ se t é um termo de φ e em t só ocorrem variáveis livres de φ. Mostrar o resultado por indução na estrutura de φ. □

2.2.1.1 Proposições e modelos

A satisfazibilidade duma proposição numa estrutura não depende das interpretações.

Proposição 2.2. *Se φ é uma proposição de $\mathcal{L}$ e $\mathcal{A}$ uma estrutura de $\mathcal{L}$ então verifica-se uma das seguintes condições:*

1. *$\mathcal{A} \models_s \varphi$ para toda a interpretação s*
2. *$\mathcal{A} \not\models_s \varphi$ para toda a interpretação s*

Demonstração. Consequência imediata da proposição 2.1. □

Se $\mathcal{A} \models \varphi$ diz-se que φ é *verdadeira* em $\mathcal{A}$ ou $\mathcal{A}$ é um *modelo* de φ. Se $\mathcal{A} \not\models \varphi$ diz-se que φ é *falsa* em $\mathcal{A}$.
Se Σ for um conjunto de proposições e $\mathcal{A} \models \psi$ para todo $\psi \in \Sigma$ então A é um *modelo* de Σ e escreve-se $\mathcal{A} \models \Sigma$.

Corolário 2.1. *Seja $\Sigma \cup \{\varphi\}$ um conjunto de proposições. $\Sigma \models \varphi$ se e só se todo o modelo de Σ for um modelo de φ.*

Exercício 2.1. *Verifica o corolário anterior.* ◇

Exercício 2.2. *Seja $\mathcal{L}$ uma linguagem de $1^{\underline{a}}$ ordem com igualdade e tal que $\mathcal{F}_0 = \{a, b\}$, $\mathcal{F}_1 = \{g\}$, $\mathcal{F}_2 = \{f, h\}$, $\mathcal{R}_1 = \{R, S\}$ e $\mathcal{R}_2 = \{P, Q\}$.*
Seja $\mathcal{A}$ uma estrutura de $\mathcal{L}$ definida por:

- *o domínio de $\mathcal{A}$ é o conjunto dos números naturais $\mathbb{N} = \{0, 1, \ldots\}$*
- $a^{\mathcal{A}} = 1$, $b^{\mathcal{A}} = 0$
- $g^{\mathcal{A}}(n) = n + 4$, $n \in \mathbb{N}$

- $h^{\mathcal{A}}(n,m) = n+m,\ f^{\mathcal{A}}(n,m) = n \times m,\ n,m \in \mathbb{N}$
- $S^{\mathcal{A}} = \{n \in \mathbb{N} \mid n \text{ é ímpar}\}$,$R^{\mathcal{A}} = \{n \in \mathbb{N} \mid n \text{ é par}\}$
- $P^{\mathcal{A}} = \{(n,m) \in \mathbb{N}^2 \mid n > m\}$
- $Q^{\mathcal{A}} = \emptyset$

i. Para cada uma das seguintes proposições diz, ***justificando*** *se é verdadeira ou falsa em* $\mathcal{A}$

a) $\forall x \exists y\ f(x,y) = a$

b) $\exists x \exists y\ f(x,y) = a$

c) $\forall x \exists y\ f(x,y) = a \rightarrow \exists y \forall x f(x,y) = a$

d) $\exists x R(x) \rightarrow \forall x R(x)$

ii. Considera as seguintes interpretações $s_i : Var \longrightarrow \mathbb{N}$ *para* $i = 1,2,3$ *e onde:*

- $s_1(x) = 2$, *para todo* $x \in Var$;
- $s_2(x) = 0$, *para todo* $x \in Var$.
- $s_3(x) = 3$, $s_3(y) = 1$ *e* $s_3(z) = 5$.

Para cada uma das fórmulas φ *seguintes e cada uma das interpretações* s_i, *diz se* $\mathcal{A} \models_{s_i} \varphi$, *para* $i = 1,2,3$*:*

a) $\exists x \exists y\ f(x,y) = z$

b) $\exists z\ f(x,y) = z \rightarrow \forall y\ (S(y)\ \vee\ R(y))$

$\diamond$

Resolução 2.2.i

a) Pretende-se que para qualquer interpretação $s : Var \longrightarrow \mathbb{N}$,

$$\mathcal{A} \models_s \forall x \exists y\ f(x,y) = a \tag{2.1}$$

Iremos usar indutivamente a definição de $\models_s$, 2.8.

Por (vii), (2.1) é verdade, se para todo o $n \in \mathbb{N}$ se tem

$$\mathcal{A} \models_{s[n/x]} \exists y\ f(x,y) = a \tag{2.2}$$

Por (viii), (2.2) é verdade, se existe um $m \in \mathbb{N}$ tal que

$$\mathcal{A} \models_{s[n/x][m/y]} f(x,y) = a \tag{2.3}$$

Por (i), (2.3) é verdade, se

$$s[n/x][m/y](f(x,y)) = s[n/x][m/y](a) \tag{2.4}$$

Mas, pela definição de interpretação estendida a termos e de $s[a/x]$ para a pertencente ao domínio de $\mathcal{A}$,

$$s[n/x][m/y](f(x,y)) = f^{\mathcal{A}}(s[n/x][m/y](x), s[n/x][m/y](y)) = f^{\mathcal{A}}(n,m) = n+m$$

e $s[n/x][m/y](a) = a^{\mathcal{A}} = 1$

Em resumo, (2.1) é verdade se

> Para todo o $n \in \mathbb{N}$, existe um $m \in \mathbb{N}$ tal que $n + m = 1$.

o que é **Falso**. Por exemplo, para $n = 3$ ter-se-ia $m = -2 \notin \mathbb{N}$.

b) Analogamente se obteria, para qualquer interpretação s

$$\mathcal{A} \models_s \exists x \exists y \; f(x,y) = a \tag{2.5}$$

se e só se

> Existe um $n \in \mathbb{N}$ e existe um $m \in \mathbb{N}$ tal que $n + m = 1$.

o que é **Verdade**, por exemplo para $n = 0$ e $m = 1$.

c) Pretende-se que, para qualquer interpretação s

$$\mathcal{A} \models_s \forall x \exists y \; f(x,y) = a \rightarrow \exists y \forall x f(x,y) = a \tag{2.6}$$

Por (vi) da definição de $\models_s$, (2.6) é verdade se

$$\mathcal{A} \not\models_s \forall x \exists y \; f(x,y) = a \text{ ou } \mathcal{A} \models_s \exists y \forall x f(x,y) = a$$

Pela alínea (a), vimos que não é verdade que $\mathcal{A} \models_s \forall x \exists y \; f(x,y) = a$, logo $\mathcal{A} \not\models_s \forall x \exists y \; f(x,y) = a$ é verdade. E também (2.6).

d) Pretende-se que, para qualquer interpretação s

$$\mathcal{A} \models_s \exists x R(x) \rightarrow \forall x R(x) \tag{2.7}$$

Por (vi) da definição de $\models_s$, (2.7) é verdade se

$$\mathcal{A} \not\models_s \exists x R(x) \text{ ou } \mathcal{A} \models_s \forall x R(x)$$

Vamos determinar se $\mathcal{A} \models_s \exists x R(x)$. Isto é verdade, se existe $n \in \mathbb{N}$ tal que

$$\mathcal{A} \models_{s[n/x]} R(x) \tag{2.8}$$

Por (ii), (2.8) é verdade se

$$s[n/x](x) \in R^{\mathcal{A}} \tag{2.9}$$

Como $s[n/x](x) = n$, vem que (2.8) é verdade se

Existe um $n \in \mathbb{N}$ tal que n é ímpar

o que é **Verdade**. Então $\mathcal{A} \not\models_s \exists x R(x)$ é **Falso**.

Temos que analisar a veracidade de $\mathcal{A} \models_s \forall x R(x)$. Analogamente obtemos que é verdade se

Para todo o $n \in \mathbb{N}$, n é ímpar

o que também é **Falso**. Donde (2.7) é **Falsa**.

Resolução 2.2i.ii

a) Sendo s_1 uma interpretação que atribuí o valor 2 a qualquer variável, pretende-se que

$$\mathcal{A} \models_{s_1} \exists x \exists y \; f(x, y) = z \tag{2.10}$$

Isto é verdade se, existe um $n \in \mathbb{N}$ e existe um $m \in \mathbb{N}$ tal que

$$\mathcal{A} \models_{s_1[n/x][m/y]} f(x, y) = z \tag{2.11}$$

Isto é se,

$$s_1[n/x][m/y](f(x, y) = s_1[n/x][m/y](z) \tag{2.12}$$

Mas $s_1[n/x][m/y](f(x, y) = f^{\mathcal{A}}(n, m) = n + m$ e $s_1[n/x][m/y](z) = s_1(z) = 2$

Então (2.10) é verdade se

Existe um $n \in \mathbb{N}$ e existe um $m \in \mathbb{N}$ tal que $m + n = 2$

o que é **Verdade**. Basta tomar $n = m = 1$.

b) Sendo s_1 uma interpretação que atribuí o valor 2 a qualquer variável, pretende-se que

$$\mathcal{A} \models_{s_1} \exists z \; f(x, y) = z \rightarrow \forall y \; (S(y) \; \vee \; R(y)) \tag{2.13}$$

Mais uma vez isto é verdade se $\mathcal{A} \not\models_{s_1} \exists z \; f(x, y) = z$ ou se $\mathcal{A} \models_{s_1} \forall y \; (S(y) \; \vee \; R(y))$

Temos que:

$$\mathcal{A} \models_{s_1} \exists z\ f(x,y) = z \tag{2.14}$$

se existe $n \in \mathbb{N}$ tal que

$$\mathcal{A} \models_{s_1[n/z]} f(x,y) = z \tag{2.15}$$

Isto é se,

$$s_1[n/z](f(x,y)) = s_1[n/z](z) \tag{2.16}$$

e $s_1[n/z](f(x,y)) = f^{\mathcal{A}}(s_1[n/z](x), s_1[n/z](y)) = f^{\mathcal{A}}(s_1(x), s_1(y)) = 2+2$ e $s_1[n/z](z) = n$

Então (2.14) é verdade se

Existe um $n \in \mathbb{N}$ tal que $2 + 2 = n$

o que é **Verdade**. Tomar $n = 4$.

Mas então $\mathcal{A} \not\models_{s_1} \exists z\ f(x,y) = z$ é **Falso**.

Temos que

$$\mathcal{A} \models_{s_1} \forall y\ (S(y)\ \vee\ R(y)) \tag{2.17}$$

se, para todo o $n \in \mathbb{N}$

$$\mathcal{A} \models_{s_1[n/y]} (S(y)\ \vee\ R(y)) \tag{2.18}$$

Por (v) da definição de $\models_s$, (2.18) é verdade se $\mathcal{A} \models_{s_1[n/y]} S(y)$ ou $\mathcal{A} \models_{s_1[n/y]} R(y)$

$$\mathcal{A} \models_{s_1[n/y]} S(y) \tag{2.19}$$

se

$$s_1[n/y](y) \in S^{\mathcal{A}} \tag{2.20}$$

isto é se n é par.

Analogamente $\mathcal{A} \models_{s_1[n/y]} R(y)$, se n é ímpar. Então, (2.17) é verdade se

Para todo o $n \in \mathbb{N}$, n é par ou n é ímpar

o que é **Verdade**. Nota que é importante a disjunção estar no âmbito do quantificador e não o contrário: é falso que *para todo* $n \in \mathbb{N}$, n *é par ou para todo* $n \in \mathbb{N}$, n *é ímpar.*

Finalmente, temos que (2.13) é **Verdade**.

Exercício 2.3. *Seja $\mathcal{L}$ uma linguagem de primeira ordem com igualdade e tal que $\mathcal{F}_0 = \{a\}$, $\mathcal{F}_1 = \{g\}$, $\mathcal{R}_1 = \{R\}$ e $\mathcal{R}_2 = \{P, Q\}$. Seja $\mathcal{A}$ a estrutura de $\mathcal{L}$ definida por:*

- *o universo de $\mathcal{A}$ é o conjunto de números naturais $\mathbb{N} = \{0, 1, \ldots\}$;*

- $a^{\mathcal{A}} = 2$;
- $g^{\mathcal{A}}(n) = n + 1$, *para* $n \in \mathbb{N}$;
- $R^{\mathcal{A}} = \{n \in \mathbb{N} \mid n \text{ é par}\}$;
- $P^{\mathcal{A}} = \{(n, m) \in \mathbb{N}^2 \mid n \leq m\}$;
- $Q^{\mathcal{A}} = \emptyset$.

Para cada uma das proposição indica se é verdadeira ou falsa em $\mathcal{A}$.

1. $\exists x g(x) = a \wedge \forall x g(x) \neq x$
2. $\exists x \exists y Q(x, y)$
3. $\neg R(a) \rightarrow \forall x \forall y\, Q(x, y)$
4. $\forall x (R(x) \rightarrow \forall x\, R(g(g(x))))$
5. $\forall x \exists y P(x, y)$
6. $\exists y \forall x P(x, y)$
7. $\exists y \forall x P(y, x)$
8. $\forall x (R(x) \vee \exists y (y = g(x) \wedge R(y))$

$\diamond$

Exercício 2.4. *Seja* $\mathcal{L}$ *a mesma linguagem do exercício 2.3. Para cada uma das proposição seguintes indica uma estrutura de* $\mathcal{L}$ *onde ela é verdadeira e outra onde ela é falsa. Pode concluir que nenhuma das proposição e também nenhuma negação de uma das proposição é uma fórmula válida?*

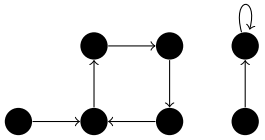
1. $\forall x \forall y\, x = y$
2. $\forall x \exists y P(x, y) \rightarrow \exists y \forall x P(x, y)$
3. $\forall x\, x = a$
4. $\forall x \forall y (R(x) \rightarrow R(y))$
5. $\exists z (g(z) \neq a) \rightarrow \exists z (g(z) = a)$
6. $\forall x\, Q(x, g(x)) \leftrightarrow \exists x\, P(a, x)$
7. $\forall y (\forall x (R(x) \rightarrow P(x, x)) \rightarrow (R(y) \rightarrow \forall x P(x, x)))$

$\diamond$

2.2.1.2 Caracterização de modelos por proposições

Uma proposição pode ser vista como uma caracterização de um conjunto de modelos (que a satisfazem).

Estruturas para a teoria dos grafos Vamos ilustrar isso considerando que as estruturas são grafos dirigidos e considerar proposições que caracterizam propriedades desses grafos.
Seja $\mathcal{G}_1 = (V, E)$, com $E \subseteq V \times V$ o seguinte digrafo (grafo dirigido):



Como podemos escrever fórmulas que caracterizem propriedades de G_1? Definindo uma linguagem de 1ª ordem, $\mathcal{L}_G$, com igualdade, sem símbolos funcionais e apenas com um símbolo relacional binário $R_2 = \{G\}$. E considerando, $\mathcal{G}_1$ uma estrutura para a interpretação das fórmulas, onde V é o domínio e E a interpretação de um símbolo relacional binário.
Algumas fórmulas são: $G(x, x)$, $\exists x \forall y\ G(y, x)$, $\forall x \forall y (G(x, y) \rightarrow G(y, x))$
Qualquer estrutura $\mathcal{G}$ para $\mathcal{L}_G$ é um digrafo.
Seja φ_1 a fórmula $\forall x \exists y G(x, y)\ \wedge\ \forall x \forall y \forall z ((G(x, y)\ \wedge\ G(x, z)) \rightarrow y = z)$
Tem-se que $\mathcal{G}_1 \models \varphi_1$. E que outros digrafos satisfazem φ_1? Aqueles que cada nó tem exactamente grau de saída 1 (isto é representam funções). Podemos dizer que φ_1 caracteriza um conjunto de digrafos.

Exercício 2.5. *Descreve os grafos que são os modelos de cada uma das seguintes fórmulas:*

1. $\forall x \forall y (G(x, y) \rightarrow G(y, x)))$
2. $\forall x \forall y \forall z ((G(x, z)\ \wedge\ G(z, y)) \rightarrow G(x, y))$

$\diamond$

Inversamente, dado um grafo $\mathcal{G}$ podemos estar interessados em saber se ele tem uma dada propriedade. Se essa propriedade for expressa por uma fórmula φ de $\mathcal{L}_\mathcal{G}$ (não necessariamente uma proposição) então o que queremos é saber se $\mathcal{G} \models \varphi$.

Exercício 2.6. *Mostra que dado φ de $\mathcal{L}_\mathcal{G}$ e $\mathcal{G}$ de domínio finito, existe um algoritmo que determina se $\mathcal{G} \models \varphi$.* $\diamond$

Resolução 2.6
Por indução na estrutura de φ.

Estruturas para autómatos finitos Seja $\mathcal{L}_A$ uma linguagem de $1^{\underline{a}}$ ordem com igualdade e $R_1 = \{I, F\}$, $R_2 = \{R_a, R_b\}$. Uma estrutura $\mathcal{A}$ para $\mathcal{L}_A$ é um autómato finito não determinístico sobre $\{a, b\}$ se e só se $|I^A| = 1$.

Exemplo 2.12. *a) Define uma proposição φ de $\mathcal{L}_A$ tal que uma estrutura $\mathcal{A}$ de $\mathcal{L}_A$ é um modelo de φ se e só se $\mathcal{A}$ é um autómato finito determinístico.*

Basta considerar φ a fórmula:

$$\forall x \forall y \forall y_1 (((R_a(x,y) \ \wedge \ R_a(x,y_1)) \rightarrow y = y_1) \ \wedge \ ((R_b(x,y) \ \wedge \ R_b(x,y_1)) \rightarrow y = y_1))$$

*b) Indica, **justificando**, duas estruturas de $\mathcal{L}_A$, $\mathcal{A}_1$ e $\mathcal{A}_2$ tal que $\mathcal{A}_1 \models \varphi$ e $\mathcal{A}_2 \models \neg\varphi$.*

Seja a estrutura $\mathcal{A}_1$ de $\mathcal{L}_A$:

- *o universo é $Q = \{q_0, q_1\}$*
- $I^{\mathcal{A}_1} = \{q_0\}$
- $F^{\mathcal{A}_1} = \{q_1\}$
- $R_a^{\mathcal{A}_1} = \{(q_0, q_0)\}$
- $R_b^{\mathcal{A}_1} = \{(q_0, q_1), (q_1, q_1)\}$

Verifica que $\mathcal{A}_1 \models \varphi$.

Considera agora $\mathcal{A}_2$ de $\mathcal{L}_A$ igual a $\mathcal{A}_1$, excepto que $R_a^{\mathcal{A}_1} = \{(q_0, q_0), (q_0, q_1)\}$.

Verifica que $\mathcal{A}_2 \models \neg\varphi$.

Exercício 2.7. *a) Define uma proposição φ de $\mathcal{L}_A$ tal que uma estrutura $\mathcal{A}$ de $\mathcal{L}_A$ é um modelo de φ se e só se $\mathcal{A}$ é um autómato finito que reconhece alguma palavra de $\Sigma = \{a, b\}$ de comprimento 2.*

*b) Indica, **justificando**, duas estruturas de $\mathcal{L}_A$, $\mathcal{A}_1$ e $\mathcal{A}_2$ tal que $\mathcal{A}_1 \models \varphi$ e $\mathcal{A}_2 \models \neg\varphi$*

$\diamond$

2.2.1.3 Validade e satisfazibilidade

Proposição 2.3. *Seja φ uma fórmula duma linguagem de $1^{\underline{a}}$ ordem $\mathcal{L}$.*

1. *φ é válida se e só se $\neg\varphi$ não é satisfazível*
2. *φ é satisfazível se e só se $\neg\varphi$ não é válida*
3. *φ é válida se e só se $\forall x \ \varphi$ é válida*

4. φ é satisfazível se e só se $\exists x\ \varphi$ é satisfazível

Demonstração. Consequência directa das definições. □

Seja φ uma fórmula duma linguagem de 1ª ordem $\mathcal{L}$. Uma fórmula φ de $\mathcal{L}$ é válida se a correspondente forma booleana é uma tautologia.

Definição 2.10. *Dada uma fórmula φ o* conjunto das suas sub-fórmulas principais*, $SP(\varphi)$ é definido indutivamente na estrutura de φ por:*

- *se φ é uma fórmula atómica, $SP(\varphi) = \{\varphi\}$*
- *se φ é $\forall x\psi$, $SP(\varphi) = \{\varphi\}$*
- *se φ é $\exists x\psi$, $SP(\varphi) = \{\forall x\neg\psi\}$*
- *se φ é $\neg\psi$, $SP(\varphi) = SP(\psi)$*
- *se φ é $\psi_1 \circ \psi_2$, onde $\circ$ pode ser $\wedge$, $\vee$ ou $\rightarrow$, $SP(\varphi) = SP(\psi_1) \cup SP(\psi_2)$*

Sendo φ a fórmula $\forall x P(x,y)\ \wedge\ \exists x P(x,y)\ \wedge\ (P(z,x)\ \vee\ \forall x P(x,y))$

$$SP(\varphi) = \{\forall x P(x,y), \forall x \neg P(x,y), P(z,x)\}$$

Definição 2.11. *Associando a cada sub-fórmula principal duma fórmula φ, uma variável proposicional diferente e negando a fórmula correspondente a um quantificador existêncial, obtemos a* forma booleana *de φ.*

Para o exemplo anterior,

$$p_1\ \wedge\ \neg p_2\ \wedge\ (p_3\ \vee\ p_1)$$

onde $p_1 = \forall x P(x,y)$, $p_2 = \forall x \neg P(x,y)$ e $p_3 = P(z,x)$

Proposição 2.4. *Se a forma booleana duma fórmula φ é uma tautologia, então φ é válida.*

Demonstração. Seja $\mathcal{A}$ uma estrutura de $\mathcal{L}$ e s uma interpretação . Para cada $\psi \in SP(\varphi)$, se $\mathcal{A} \models_s \psi$ atribui-se à correspondente variável proposicional o valor **V**, se $\mathcal{A} \not\models_s \psi$, atribui-se o valor **F**. Como a forma booleana é satisfeita para essa valorização, também se tem que $\mathcal{A} \models_s \varphi$. □

2.2.2 Equivalência semântica

Definição 2.12. *Sejam φ e ψ duas fórmulas duma linguagem de $1^{\underline{a}}$ ordem $\mathcal{L}$. Se $\varphi \models \psi$ e $\psi \models \varphi$ então φ e ψ são* semanticamente equivalentes *e escreve-se $\varphi \equiv \psi$.*

As *Leis de DeMorgan* para quantificadores são fórmulas semanticamente equivalentes são as

Proposição 2.5. *(Leis de DeMorgan)*

$$\begin{aligned} \neg\forall x\varphi &\equiv \exists x\neg\varphi \\ \neg\exists x\varphi &\equiv \forall x\neg\varphi \end{aligned}$$

Demonstração. Suponhamos que para toda a estrutura $\mathcal{A}$, e interpretação s se tem $\mathcal{A} \models_s \neg\forall x\varphi$. Isto equivale a dizer que $\mathcal{A} \not\models_s \forall x\varphi$, isto é, não se verifica que para todo o $a \in A$, $\mathcal{A} \models_{s[a/x]} \varphi$. Isto é, existe pelo menos um $a \in A$ tal que $\mathcal{A} \not\models_{s[a/x]} \varphi$, i.e, $\mathcal{A} \models_{s[a/x]} \neg\varphi$, que é a definição de $\mathcal{A} \models_s \exists x\neg\varphi$. O que prova a primeira equivalência (dado termos usado só as definições). Analogamente se prova a segunda lei de DeMorgan. □

Temos também:

Proposição 2.6.

$$\begin{aligned} \forall x(\varphi \wedge \psi) &\equiv \forall x\varphi \wedge \forall x\psi \quad \textit{(mas não com } \vee\textit{)} \\ \exists x\varphi \vee \exists x\psi &\equiv \exists x(\varphi \vee \psi) \quad \textit{(mas não com } \wedge\textit{)} \\ \forall x\forall y\varphi &\equiv \forall y\forall x\varphi \\ \exists x\exists y\varphi &\equiv \exists y\exists x\varphi \end{aligned}$$

Se x não ocorre livre em ψ

$$\begin{aligned} \forall x(\varphi \wedge \psi) &\equiv \forall x\varphi \wedge \psi \\ \forall x(\varphi \vee \psi) &\equiv \forall x\varphi \vee \psi \\ \exists x(\varphi \wedge \psi) &\equiv \exists x\varphi \wedge \psi \\ \exists x(\varphi \vee \psi) &\equiv \exists x\varphi \vee \psi \\ \forall x(\psi \to \varphi) &\equiv \psi \to \forall x\varphi \\ \exists x(\varphi \to \psi) &\equiv \forall x\varphi \to \psi \\ \exists x(\psi \to \varphi) &\equiv \psi \to \exists x\varphi \\ \forall x(\varphi \to \psi) &\equiv \exists x\varphi \to \psi \end{aligned}$$

Demonstração. Queremos provar que $\forall x(\varphi \wedge \psi) \equiv \forall x\varphi \wedge \forall x\psi$. Suponhamos que dada uma estrutura $\mathcal{A}$, e interpretação s se tem $\mathcal{A} \models_s \forall x(\varphi \wedge \psi)$. Isto equivale a dizer que para todo o $a \in A$, $\mathcal{A} \models_{s[a/x]} \varphi$ e $\mathcal{A} \models_{s[a/x]} \psi$. Mas isso é o mesmo que ter $\mathcal{A} \models_s \forall x\varphi$ e $\mathcal{A} \models_s \forall x\psi$.
Mas a equivalência não se verifica se se substituir $\wedge$ por $\vee$. Para um contra-exemplo considera a linguagem $\mathcal{L}$ com $\mathcal{R}_1 = \{P, Q\}$ e a estrutura de $\mathcal{L}$, $\mathcal{A} = (A, .^{\mathcal{A}})$ com $A = \{1, 2\}$, $P^{\mathcal{A}} = \{1\}$ e $Q^{\mathcal{A}} = \{2\}$. Então $\mathcal{A} \models \forall x(P(x) \vee Q(x))$ e $\mathcal{A} \not\models \forall xP(x) \vee \forall xQ(x)$ (verifica).
Para provar que $\exists x\varphi \vee \exists x\psi \equiv \exists x(\varphi \vee \psi)$, basta usar a equivalência anterior e a Proposição 2.5, notando que se duas fórmulas são equivalentes as suas negações também o são. Para provar que a equivalência não se verifica se se substituir o $\vee$ por $\wedge$, basta considerar a linguagem $\mathcal{L}$ e a estrutura $\mathcal{A}$ definidas acima. Então $\mathcal{A} \models \exists xP(x) \wedge \exists xQ(x)$, mas $\mathcal{A} \not\models \exists x(P(x) \wedge Q(x))$ (verifica).
As duas equivalências seguintes saem directamente da Definição 2.8, casos vii. e viii.. □

Exercício 2.8. *Mostra as restantes equivalências da Proposição 2.6.* ◇

2.2.3 Substituição de variáveis

Seja φ uma fórmula duma linguagem de 1ª ordem $\mathcal{L}$, t um termo de $\mathcal{L}$ e $x \in Var$ uma variável. Denota-se por $\varphi[t/x]$ (ou φ^t_x) a fórmula que se obtém de φ *substituindo* todas as ocorrências livres de x em φ por t.
Seja φ a fórmula $\forall x(P(x) \wedge Q(x)) \to (\neg P(x) \vee Q(y))$. Então, $\varphi[f(x, y)/x]$ é a fórmula

$$\forall x(P(x) \wedge Q(x)) \to (\neg P(f(x, y)) \vee Q(y)).$$

E se φ for $\forall y(\neg P(x) \vee Q(y))$, qual a fórmula $\varphi[f(x, y)/x]$? Fica

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

mas aqui a variável y do termo $f(x, y)$ passou a estar ligada! Para evitar isto, define-se:

Definição 2.13 (Variável substituível)**.** *Uma variável x é* substituível *por t em φ se nenhuma ocorrência livre de x em φ for numa sub-fórmula $\forall y\psi$ ou $\exists y\psi$, para qualquer variável y que ocorra em t. Também se pode dizer que t é* livre *para x em φ.*

Exercício 2.9. *Seja φ a fórmula $\exists x(P(y, z) \wedge \forall y(\neg Q(x, y) \vee P(y, z)))$*
Determina:

1. *$\varphi[w/x]$, $\varphi[w/y]$, $\varphi[f(x)/y]$ e $\varphi[g(y, z)/z]$*
2. *Para quais dos termos w, $f(x)$ e $g(y, z)$ é x substituível em φ? e y?*

◇

Proposição 2.7.

1. *Se x é substituível por t em φ, são válidas as fórmulas:*

   a) $\forall x\, \varphi \to \varphi[t/x]$

   b) $\varphi[t/x] \to \exists x \varphi$

2. *Se y não ocorre em φ, então* $\forall x \varphi \equiv \forall y \varphi[y/x]$

Demonstração. Vamos apenas considerar 2.7.1a. Seja $\mathcal{A}$ uma estrutura e s uma interpretação. Temos que mostrar que se $\mathcal{A} \models_s \forall x \varphi$ então $\mathcal{A} \models \varphi[t/x]$. Para todo o $a \in A$, temos $\mathcal{A} \models_{s[a/x]} \varphi$. Seja $a' = s(t)$, e como t é livre para x em φ sabemos que esse será o valor da interpretação de todas as ocorrências de t em $\varphi[t/x]$. Mas então temos também $\mathcal{A} \models_s \varphi[t/x]$. O caso 1b demonstra-se por contraposição. □

Exercício 2.10. *Termina a demonstração da Proposição 2.7.* ⋄

2.2.4 Forma normal prenexa

Uma fórmula duma linguagem de 1ª ordem $\mathcal{L}$ está em *forma normal prenexa* se ou:

- não contém quantificadores
- é da forma

  $$Q_1 x_1 Q_2 x_2 \ldots Q_n x_n \varphi$$

  onde cada Q_i é ou $\forall$ ou $\exists$, $1 \leq i \leq n$, e φ é uma fórmula sem quantificadores.

Por exemplo,

$$\forall x \forall y \forall z \neg((\neg P(x,x) \ \vee \ (\neg P(x,y) \ \wedge \ P(z,z))) \ \vee \ \neg P(w,0))$$

Proposição 2.8. *Qualquer fórmula duma linguagem de 1ª ordem é semanticamente equivalente a uma fórmula em forma normal prenexa.*

Demonstração. **Ideia:**
Transformar φ de modo a que:

a) só ocorram as conectivas $\wedge$, $\vee$ e $\neg$

b) nenhuma variável ocorra livre e ligada

c) uma mesma variável só aparece quantificada no máximo uma vez.

As duas últimas condições obtêm-se substituindo, sempre que necessário, as variáveis quantificadas por variáveis novas usando a Proposição 2.7.2. Seja φ_1 a fórmula resultante, que é semanticamente equivalente a φ (justifica!). Aplicar às sub-fórmulas de φ_1, as seguintes transformações, até que nenhuma se aplique (onde $\circ$ é $\wedge$ ou $\vee$):

$$\begin{array}{llll} \neg\forall x\varphi & \longrightarrow & \exists x\neg\varphi & \qquad \forall x\psi \circ \varphi \longrightarrow \forall x(\psi \circ \varphi) \\ \neg\exists x\varphi & \longrightarrow & \forall x\neg\varphi & \qquad \varphi \circ \exists x\psi \longrightarrow \exists x(\varphi \circ \psi) \\ \varphi \circ \forall x\psi & \longrightarrow & \forall x(\varphi \circ \psi) & \qquad \exists x\psi \circ \varphi \longrightarrow \exists x(\psi \circ \varphi) \end{array}$$

Por indução no número de quantificadores que estão no âmbito de conectivas φ_1 mostra-se que a fórmula resultante está em forma normal prenexa e é semanticamente equivalente a φ. □

Exercício 2.11. *Obtém uma forma normal prenexa para*

$$\forall x(P(x,x) \wedge (\forall y P(x,y) \vee \exists y\neg P(y,y))) \wedge G(w,0)$$

◇

Resolução 2.11

1. Substituir as variáveis ligadas por variáveis novas (Prop.2.7:2)

$$\forall x(P(x,x) \wedge (\forall y P(x,y) \vee \exists z\neg P(z,z))) \wedge G(w,0)$$

2. Mover $\forall x$ para fora:

$$\forall x((P(x,x) \wedge (\forall y P(x,y) \vee \exists z\neg P(z,z))) \wedge G(w,0))$$

3. Mover $\forall y$ para fora:

$$\forall x\forall y((P(x,x) \wedge (P(x,y) \vee \exists z\neg P(z,z))) \wedge G(w,0))$$

4. Mover $\exists z$ para fora:

$$\forall x\forall y\exists z((P(x,x) \wedge (P(x,y) \vee \neg P(z,z))) \wedge G(w,0))$$

Exercício 2.12. *Encontra a forma normal prenexa de*

$$\forall x R(x,y) \wedge \exists y(\neg(\forall z(R(y,z) \to Q(u,y))))$$

◇

Leituras suplementares [BE00] (Cap. 10,11)

2.3 Sistema de dedução natural para a lógica de 1ª ordem

As regras para as conectivas são as mesmas que para a lógica proposicional.
Haverá novas regras de **introdução** e **eliminação** para:

- a igualdade (=)
- para os quantificadores a introdução corresponderá a uma **generalização** e a eliminação uma **instanciação**:

  (∀-E) de $\forall x P(x)$ deduzir $P(t)$

  (∀-I) pode-se deduzir $\forall x P(x)$ se para qualquer y se deduzir $P(y)$

  (∃-I) de $P(t)$ deduzir $\exists x P(x)$

  (∃-E) se se deduziu $\exists x P(x)$ podemos supor $P(t)$

2.3.1 Regras de inferência DN:igualdade

Introdução de =

$$\frac{}{t = t}{=}\mathbf{I}$$

Podemos deduzir sempre que um termo t é igual a si próprio...a igualdade é reflexiva

Eliminação de =

$$\frac{t_1 = t_2 \qquad \varphi[t_1/x]}{\varphi[t_2/x]}{=}\mathbf{E}$$ e x é substituível por t_1 e por t_2 em φ

Se t_1 é igual a t_2 podemos substituir em φ, t_1 por t_2.

Exercício 2.13. *Mostra que* $t_1 = t_2 \vdash t_2 = t_1$ *e* $t_1 = t_2, t_2 = t_3 \vdash t_1 = t_3$, *i.e. , que são deduzíveis as propriedades de simetria e transitividade de* $=$. $\diamond$

Resolução 2.13

1	$t_1 = t_2$	
2	$t_1 = t_1$	$=$**I**
3	$t_2 = t_1$	$=$**E** (φ é $x = t_1$), 1, 2

Notar que para aplicar a regra $=$ **E**, sendo φ a fórmula $x = t_1$, $(x = t_1)[t_1/x]$ é precisamente $t_1 = t_1$ e a igualdade é $t_1 = t_2$. A conclusão $t_2 = t_1$ é $(x = t_1)[t_2/x]$.

1	$t_1 = t_2$	
2	$t_2 = t_3$	
3	$t_1 = t_3$	$=$**E** (φ é $t_1 = x$), 2, 2

Notar que $(t_1 = x)[t_2/x]$ é precisamente $t_1 = t_2$ e $t_1 = t_3$ é $(t_1 = x)[t_3/x]$.

2.3.2 Regras de inferência DN:quantificador universal

Eliminação de $\forall$

$$\frac{\forall x\ \varphi}{\varphi[t/x]}\ \forall\mathbf{E}$$ onde x é substituível por t em φ

Se se deduziu $\forall x\ \varphi$ podemos substituir x em φ por qualquer termo t (**instanciar** o x com qualquer valor).

Introdução de $\forall$

$$\frac{\begin{array}{c}[v]\\ \vdots\\ \varphi[v/x]\end{array}}{\forall x\ \varphi}\forall\mathbf{I}$$ onde v é uma variável nova (não ocorre antes)

Se supondo uma variável nova y podemos deduzir $\varphi[y/x]$, então podemos deduzir φ com qualquer valor (**generalização**). Para supor y temos de começar uma nova sub-dedução mas sem novas premissas. Na notação de Fitch isso é indicando com uma nova identação. Esta sub-dedução termina quando concluímos a regra.

Exercício 2.14. *Mostrar que*

$$\forall x(P(x) \to Q(x)), \forall x P(x) \vdash \forall x Q(x)$$

$\diamond$

Resolução 2.14

1	$\forall x(P(x) \to Q(x))$		
2	$\forall x P(x)$		
3	v | $P(v) \to Q(v)$		$\forall$E, 1
4	| $P(v)$		$\forall$E, 2
5	| $Q(v)$		$\to$E, 3, 4
6	$\forall x Q(x)$		$\forall$I, 3–5

Exercício 2.15. *Mostrar que*

$$P(t), \forall x(P(x) \to \neg Q(x)) \vdash \neg Q(t)$$

◇

Resolução 2.15

1	$P(t)$	
2	$\forall x(P(x) \to \neg Q(x))$	
3	$P(t) \to \neg Q(t)$	$\forall$E, 2
4	$\neg Q(t)$	$\to$E, 3, 1

Exercício 2.16. *Mostra que*

$$\vdash \forall x(P(x) \to Q(x)) \to (\forall x P(x) \to \forall x Q(x))$$

◇

Resolução 2.16

1	$\forall x(P(x) \to Q(x))$		
2	$\forall x P(x)$		
3	u $P(u)$		$\forall$E, 2
4	$P(u) \to Q(u)$		$\forall$E, 1
5	$Q(u)$		$\to$E, 3, 5
6	$\forall x Q(x)$		$\forall$I, 3–5
7	$\forall x P(x) \to \forall x Q(x)$		$\to$I, 2–6
8	$\forall x(P(x) \to Q(x)) \to (\forall x P(x) \to \forall x Q(x))$		$\to$I, 1–7

Exercício 2.17. *Mostra que*

$$\vdash (\forall x P(x) \ \wedge \ \forall x Q(x)) \to \forall x(P(x) \ \wedge \ Q(x))$$

◇

Resolução 2.17

1		$\forall x P(x) \land \forall x Q(x)$	
2	u	$\forall x P(x)$	$\land$E, 1
3		$\forall x Q(x)$	$\land$E, 1
4		$P(u)$	$\forall$E, 2
5		$Q(u)$	$\forall$E, 3
6		$P(u) \land Q(u)$	$\land$I, 2, 3
7		$\forall x(P(x) \land Q(x))$	$\forall$I, 2–6
8		$(\forall x P(x) \land \forall x Q(x)) \to \forall x(P(x) \land Q(x))$	$\to$I, 1–7

Exercício 2.18. *Mostra que*

$$\forall x(P(x) \lor Q(x)), \neg\forall x Q(x), \forall x(R(x) \to \neg P(x)) \vdash \neg\forall x R(x)$$

◇

Resolução 2.18

1		$\forall x(P(x) \lor Q(x))$	
2		$\neg\forall x Q(x)$	
3		$\forall x(R(x) \to \neg P(x))$	
4		$\forall x R(x)$	
5	u	$R(u) \to P(u)$	$\forall$E, 3
6		$P(u) \lor Q(u)$	$\forall$E, 1
7		$P(u)$	
8		$R(u)$	$\forall$E, 4
9		$\neg P(u)$	$\to$E, 5, 8
10		**F**	**F**I, 7, 9
11		$Q(u)$	**F**E, 10
12		$Q(u)$	
13		$Q(u)$	R, 11
14		$Q(u)$	$\lor$E, 6, 7–11, 12–13
15		$\forall x Q(x)$	$\forall$I, 5–13
16		**F**	**F**I, 2, 15
17		$\neg\forall x R(x)$	

2.3.3 Regras de inferência DN:quantificador existencial

Introdução de $\exists$

$$\frac{\varphi[t/x]}{\exists x\ \varphi}\ \exists\mathbf{I}$$ onde x é substituível por t em φ

Podemos deduzir $\exists x\ \varphi$ se já tivermos deduzido $\varphi[t/x]$ para algum termo t.

Eliminação de $\exists$

$$\frac{\exists x\ \varphi \qquad \begin{array}{c}[v \quad \varphi[v/x]] \\ \vdots \\ \psi\end{array}}{\psi}\ \exists\mathbf{E}$$ onde v é uma variável nova que não ocorre antes nem em ψ

Se deduzimos $\exists x\varphi$, então existe um valor em que φ se verifica. Assim, supondo que v representa esse valor e se supondo $\varphi[v/x]$ se deduzir ψ onde v não ocorra, podemos deduzir ψ (para um valor genérico). Neste caso temos então uma nova sub-dedução em que é considerada uma nova variável (v) e onde existe uma premissa nova ($\varphi[v/x]$).

Exercício 2.19. *Mostrar que:*

a) $\forall x\ \varphi \vdash \exists x\ \varphi$

$\forall x\ (P(x) \to Q(x)), \exists x\ P(x) \vdash \exists x\ Q(x)$

$\diamond$

Resolução 2.19

	1	$\forall x\ \varphi$		
a)	2	$\varphi[t/x]$		$\forall$E, 1
	3	$\exists x\ \varphi$		$\exists$I, 2

	1	$\forall x\ (P(x) \to Q(x))$		
	2	$\exists x\ P(x)$		
	3	v	$P(v)$	
b)	4		$P(v) \to Q(v)$	$\forall$E, 1
	5		$Q(v)$	$\to$E, 4, 3
	6		$\exists x\ Q(x)$	$\exists$I, 5
	7	$\exists x\ Q(x)$		$\exists$E, 3–6

Como no caso das regras de introdução e eliminação de conectivas, também aqui se tem de ter em conta o âmbito das sub-deduções e em especial as variáveis (livres) só podem ser usadas nas sub-deduções em que são supostas ou ocorrem em premissas dadas.
Por exemplo, a seguinte é uma dedução está *errada*:

1	$\forall x\ (P(x) \to Q(x))$		
2	$\exists x\ P(x)$		
3	v | $P(v)$		
4	| $P(v) \to Q(v)$		$\forall$E, 1
5	| $Q(v)$		$\to$E, 4, 3
6	$Q(v)$		$\exists$I, 3–5
7	$\exists x\ Q(x)$		$\exists$E, 6

Na fórmula da linha 6, ocorre uma variável que só devia ocorrer na sub-dedução iniciada na linha 3!

Exercício 2.20. *Mostrar que*

$$\forall x(Q(x) \to R(x)), \exists x\ (P(x)\ \wedge\ Q(x)) \vdash \exists x(P(x)\ \wedge\ R(x))$$

$\diamond$

Resolução 2.20

1	$\forall x(Q(x) \to R(x))$		
2	$\exists(P(x)\ \wedge\ Q(x))$		
3	v | $P(v)\ \wedge\ Q(v)$		
4	| $Q(v) \to R(v)$		$\forall$E, 1
5	| $Q(v)$		$\wedge$E, 3
6	| $R(v)$		$\to$E, 4
7	| $P(v)$		$\wedge$E, 3
8	| $P(v)\ \wedge\ R(v)$		$\wedge$I, 7, 6
9	| $\exists x\ (P(x)\ \wedge\ R(x))$		$\exists$I, 8
10	$\exists x\ (P(x)\ \wedge\ R(x))$		$\exists$E, 2, 3–9

Exercício 2.21. *Mostrar que*

$$\exists x\ P(x), \forall x \forall y\ (P(x) \to Q(y)) \vdash \forall y\ Q(y)$$

◇

Resolução 2.21

1	$\exists x\, P(x)$		
2	$\forall x \forall y\, (P(x) \to Q(y))$		
3	v | u | $P(u)$		
4		$\forall y\, (P(u) \to Q(y)$	$\forall$E, 2
5		$(P(u) \to Q(v)$	$\forall$E, 4
6		$Q(v)$	$\to$E, 5, 3
7		$Q(v)$	$\exists$E, 1, 3–6
8	$\forall y\, Q(y)$		$\forall$I, 3–7

Todas as regras do sistema DN para a lógica de primeira ordem encontram-se na Figura 2.1

2.4 Equivalência dedutiva

Dadas duas fórmulas φ e ψ dizemos que são dedutivamente equivalentes se e só se $\varphi \vdash \psi$ e $\psi \vdash \varphi$. E denotamos por $\varphi \dashv\vdash \psi$.

Proposição 2.9. *Sejam φ e ψ dumas fórmulas duma linguagem da lógica de $1^{\underline{a}}$ ordem. Então:*

1. $\neg\forall x\varphi \dashv\vdash \exists x\neg\varphi$

2. $\neg\exists x\varphi \dashv\vdash \forall x\neg\varphi$

3. $\forall x\varphi \ \wedge\ \forall x\psi \dashv\vdash \forall x(\varphi \ \wedge\ \psi)$

4. $\exists x\varphi \ \vee\ \exists x\psi \dashv\vdash \exists x(\varphi \ \vee\ \psi)$

5. *Se x não ocorre livre em ψ, e $\circ$ é $\wedge$ ou $\vee$:*

   (a) $\forall x\varphi \circ \psi \dashv\vdash \forall x(\varphi \circ \psi)$

   (b) $\exists x\varphi \circ \psi \dashv\vdash \exists x(\varphi \circ \psi)$

	Introdução	**Eliminação**
$\wedge$	$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\wedge\mathbf{I}$	$\dfrac{\varphi \wedge \psi}{\varphi}\wedge\mathbf{E}_1 \qquad \dfrac{\varphi \wedge \psi}{\psi}\wedge\mathbf{E}_2$
$\vee$	$\dfrac{\varphi}{\varphi \vee \psi}\vee\mathbf{I}_1 \qquad \dfrac{\psi}{\varphi \vee \psi}\vee\mathbf{I}_2$	$\dfrac{\varphi \vee \psi \qquad \begin{array}{c}[\varphi]\\ \vdots\\ \gamma\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots\\ \gamma\end{array}}{\gamma}\vee\mathbf{E}$
$\neg$	$\dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \mathbf{F}\end{array}}{\neg\varphi}\neg\mathbf{I}$	$\dfrac{\neg\neg\varphi}{\varphi}\neg\mathbf{E}$
$\mathbf{F}$	$\dfrac{\neg\varphi \quad \varphi}{\mathbf{F}}\mathbf{F}\mathbf{I}(*)$	$\dfrac{\mathbf{F}}{\varphi}\mathbf{F}\mathbf{E}$
$\rightarrow$	$\dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \psi\end{array}}{\varphi\rightarrow\psi}\rightarrow\mathbf{I}$	$\dfrac{\varphi \quad \varphi\rightarrow\psi}{\psi}\rightarrow\mathbf{E}$
$=$	$\dfrac{}{t=t}=\mathbf{I}$	$\dfrac{t_1=t_2 \quad \varphi[t_1/x]}{\varphi[t_2/x]}=\mathbf{E}$ e x é substituível por t_1 e por t_2 em φ
$\forall$	$\dfrac{\begin{array}{c}[v]\\ \vdots\\ \varphi[v/x]\end{array}}{\forall x\ \varphi}\forall\mathbf{I}$ onde v é uma variável nova (não ocorre antes)	$\dfrac{\forall x\ \varphi}{\varphi[t/x]}\ \forall\mathbf{E}$ onde x é substituível por t em φ
$\exists$	$\dfrac{\varphi[t/x]}{\exists x\ \varphi}\ \exists\mathbf{I}$ onde x é substituível por t em φ	$\dfrac{\exists x\ \varphi \qquad \begin{array}{c}[v \quad \varphi[v/x]]\\ \vdots\\ \psi\end{array}}{\psi}\ \exists\mathbf{E}$ onde v é uma variável nova que não ocorre antes, nem em ψ
R	$\dfrac{\varphi}{\varphi}\mathbf{R}$	

Figura 2.1: As regras do sistema DN para a lógica de primeira ordem

Demonstração. $\neg\forall x\varphi \vdash \exists x\neg\varphi$ Por redução ao absurdo:

1	$\neg\forall x\varphi$	
2	$\quad\neg\exists x\neg\varphi$	
3	$\quad u \quad\quad \neg\varphi[u/x]$	
4	$\quad\quad\quad \exists x\neg\varphi$	$\exists$I, 3
5	$\quad\quad\quad \mathbf{F}$	**F**I, 4, 2
6	$\quad\quad \varphi[u/x]$	**RA**, 3–5
7	$\quad\forall x\varphi$	$\forall$I, 3–6
8	$\quad\mathbf{F}$	**F**I, 7, 1
9	$\exists x\neg\varphi$	**RA**, 2–8

$\exists x\neg\varphi \vdash \neg\forall x\varphi$

1	$\exists x\neg\varphi$	
2	$\quad\forall x\varphi$	
3	$\quad u \quad \neg\varphi[u/x]$	
4	$\quad\quad\quad \varphi[u/x]$	$\forall$E, 2
5	$\quad\quad\quad \mathbf{F}$	**F**I, 3, 4
6	$\quad\mathbf{F}$	$\exists$E, 1, 2–5
7	$\neg\forall x\varphi$	$\neg$I, 2–6

$\forall x\varphi \wedge \psi \vdash \forall x(\varphi \wedge \psi)$ e x não ocorre livre em ψ

1	$\forall x\varphi \wedge \psi$	
2	$\forall x\varphi$	$\wedge$E, 1
3	ψ	$\wedge$E, 1
4	$v \quad \varphi[v/x]$	$\forall$E, 2
5	$\quad\quad \varphi[v/x] \wedge \psi$	$\wedge$I, 3, 4
6	$\quad\quad (\varphi \wedge \psi)[v/x]$	R, 5
7	$\forall x(\varphi \wedge \psi)$	$\forall$I, 4–6

$\forall x(\varphi \wedge \psi) \vdash \forall x\varphi \wedge \psi$ e x não ocorre livre em ψ

1	$\forall x(\varphi \wedge \psi)$	
2	v | $(\varphi \wedge \psi)[v/x]$	$\forall$E, 1
3	$\varphi[v/x] \wedge \psi$	R, 2
4	ψ	$\wedge$E, 3
5	$\varphi[v/x]$	$\wedge$E, 3
6	$\forall x\varphi$	$\forall$I, 2–5
7	$(\forall x\varphi) \wedge \psi$	$\wedge$I, 4, 6

$(\exists x\varphi) \vee (\exists x\psi) \dashv\vdash \exists x(\varphi \vee \psi)$

1	$(\exists x\varphi) \vee (\exists x\psi)$	
2	$\exists x\varphi$	
3	v | $\varphi[v/x]$	
4	$\varphi[v/x] \vee \psi[v/x]$	$\vee$I, 3
5	$(\varphi \vee \psi)[v/x]$	idêntico, 4
6	$\exists x(\varphi \vee \psi)$	$\exists$I, 5
7	$\exists x(\varphi \vee \psi)$	$\exists$E, 2, 3–5
8	$\exists x\psi$	
9	v | $\psi[v/x]$	
10	$\varphi[v/x] \vee \psi[v/x]$	$\vee$I, 9
11	$(\varphi \vee \psi)[v/x]$	idêntico, 10
12	$\exists x(\varphi \vee \psi)$	$\exists$I, 11
13	$\exists x(\varphi \vee \psi)$	$\exists$E, 8, 9–12
14	$\exists x(\varphi \vee \psi)$	$\vee$E, 1, 2–7, 8–13

1	$\exists x(\varphi \vee \psi)$	
2	v | $(\varphi \vee \psi)[v/x]$	
3	$\varphi[v/x] \vee \psi[v/x]$	idêntico, 2
4	| $\varphi[v/x]$	
5	| $\exists x\varphi$	$\exists$I, 4
6	| $\exists x\varphi \vee \exists x\psi$	$\vee$I, 5
7	| $\psi[v/x]$	
8	| $\exists x\psi$	$\exists$I, 7
9	| $\exists x\varphi \vee \exists x\psi$	$\vee$I, 8
10	$\exists x\varphi \vee \exists x\psi$	$\vee$E, 3, 4–6, 7–9
11	$\exists x\varphi \vee \exists x\psi$	$\exists$E, 1, 2–10

□

Exercício 2.22. *Mostra que:*

a) $\vdash \forall x(\varphi \to \psi) \to (\exists x\varphi \to \exists x\psi)$

b) $\vdash \forall x(\varphi \to \psi) \to (\forall x\varphi \to \forall x\psi)$

c) $\exists xP(x), \forall x\forall y(P(x) \wedge P(y)) \to x = y) \vdash \exists! xP(x)$

onde $\exists! xP(x)$ *representa* $\exists x(P(x) \wedge \forall y(P(y) \to y = x))$ *e diz-se* existe um único x tal que $P(x)$

d) $\vdash \forall x\forall y(x = y \to f(x) = f(y))$

e) $\vdash x = f(y) \to (\forall zP(x, z) \to P(f(y), z))$

◇

Resolução 2.22

a) $\vdash \forall x(\varphi \to \psi) \to (\exists x\varphi \to \exists x\psi)$

1		$\forall x(\varphi \to \psi)$	
2		$\exists x\varphi$	
3		u | $\varphi[u/x]$	
4		$(\varphi \to \psi)[u/x]$	$\forall$E, 1
5		$\varphi[u/x] \to \psi[u/x]$	R, 4
6		$\psi[u/x]$	$\to$E, 3, 5
7		$\exists x\psi$	$\exists$I, 6
8		$\exists x\psi$	$\exists$E, 2, 3–7
9		$\exists x\varphi \to \exists x\psi$	$\to$I, 2–8

b) $\vdash \forall x(\varphi \to \psi) \to (\forall x\varphi \to \forall x\psi)$

1		$\forall x(\varphi \to \psi)$	
2		$\forall x\varphi$	
3		u | $\varphi[u/x]$	$\forall$E, 2
4		$(\varphi \to \psi)[u/x]$	$\forall$E, 1
5		$\varphi[u/x] \to \psi[u/x]$	R, 4
6		$\psi[u/x]$	$\to$E, 3, 5
7		$\forall x\psi$	$\forall$I, 3–6
8		$\forall x\varphi \to \forall x\psi$	$\to$I, 2–7

c) $\exists x P(x), \forall x\forall y(P(x) \ \wedge \ P(y) \to x = y) \vdash \exists x(P(x) \ \wedge \ \forall y(P(y) \to y = x))$

1	$\exists x P(x)$			
2	$\forall x \forall y (P(x) \wedge P(y) \to x = y)$			
3	u	$P(u)$		
4		v	$P(v)$	
5			$P(v) \wedge P(u)$	$\wedge$I, 3, 4
6			$\forall y((P(v) \wedge P(y) \to v = y)$	$\forall$E, 2
7			$(P(v) \wedge P(u)) \to v = u)$	$\forall$E, 6
8			$v = u$	$\to$E, 5, 7
9		$P(v) \to v = u$		$\to$I, 4–8
10		$\forall y(P(y) \to y = u)$		$\to$I, 4–8
11		$P(u) \wedge \forall y(P(y) \to y = u)$		$\wedge$I, 3, 10
12		$\exists x(P(x) \wedge \forall y(P(y) \to y = x))$		$\exists$I, 11
13	$\exists x(P(x) \wedge \forall y(P(y) \to y = x))$			$\exists$E, 1, 2–12

d) $\vdash \forall x \forall y\ (x = y \to f(x) = f(y))$

1	u	v	$u = v$		
2			$f(u) = f(u)$		=I
3			$f(u) = f(v)$		=E (φ é $f(u) = f(x)$), 1, 3
4		$u = v \to f(u) = f(v)$			$\to$I, 1, 3
5		$\forall y\ (u = y \to f(u) = f(y))$			$\forall$I, 1–4
6	$\forall x \forall y\ (x = y \to f(x) = f(y))$				$\forall$I, 1–5

e) $\vdash x = f(y) \to \forall(z P(x, z) \to P(f(y), z))$

1		$x = f(y)$			
2		u	$P(x, u)$		
3			$P(f(y), u)$		=E, 1, 2
4		$P(x, u) \to P(f(y), u)$			$\to$I, 2–3
5		$\forall z(P(x, z) \to P(f(y), z))$			$\forall$I, 2–4
6	$x = f(y) \to \forall z(P(x, z) \to P(f(y), z))$				$\to$I, 1–5

Leituras suplementares [BE00] (Cap. 2.3,12,13)

2.5 Integridade e completude

Vamos ver que o sistema dedutivo DN para a lógica de 1ª ordem é **integro** e **completo**, i.e: Dado um conjunto de fórmulas Σ (premissas) e uma fórmula φ (conclusão):

Integridade se existe uma dedução de φ com premissas Σ, $\Sigma \vdash \varphi$, então φ é consequência semântica de Σ, $\Sigma \models \varphi$. Em particular se $\vdash \varphi$ (teorema), então φ é válida, $\models \varphi$.

Completude Se φ é consequência semântica de Σ, $\Sigma \models \varphi$, então existe uma dedução de φ com premissas Σ, $\Sigma \vdash \varphi$. Em particular se $\models \varphi$ então φ é um teorema, $\vdash \varphi$.

2.5.1 Integridade do sistema dedutivo DN

Proposição 2.10. *Se* $\Sigma \vdash \varphi$ *então* $\Sigma \models \varphi$

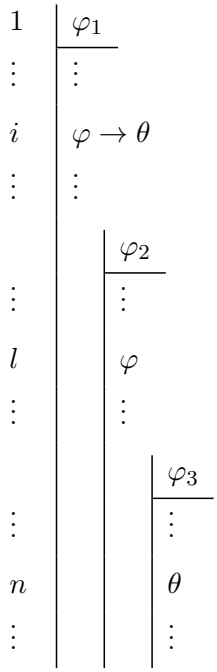
Demonstração. Dada uma dedução de φ com premissas de Σ (onde, podem não ser usadas todas as fórmulas), vamos mostrar que em cada passo p a fórmula que aí ocorre é consequência semântica das premissas (ou hipóteses) que aí são assumidas. Então em particular, φ sendo o último passo, será consequência semântica das suas premissas (ou seja Σ). Provamos por indução no número de passos da dedução:

Base. Se só houver um passo de dedução então φ é uma das premissas ou uma fórmula da forma $t = t$. Em ambos os casos é fácil ver que são consequências semânticas das premissas (mostra!)

Indução. Suponhamos que estamos no passo n e que todos os anteriores verificam a condição. Fazemos uma demonstração por casos considerando cada uma das regras. Suponhamos que a regra a aplicar é a eliminação da implicação:

$\rightarrow \mathbf{E}$

$\frac{\varphi \quad \varphi \rightarrow \psi}{\psi}$: Seja θ a fórmula deduzida no passo n por aplicação de $\rightarrow \mathbf{E}$ a $\varphi \rightarrow \theta$ e φ. E sejam $\varphi_1, \ldots, \varphi_k$ as premissas assumidas em θ. Mas as premissas para $\varphi \rightarrow \theta$ e φ estão entre os $\varphi_1, \ldots, \varphi_k$ e ambos são consequência semânticas delas:



Vamos ver que θ é consequência semântica de $\varphi_1, \ldots, \varphi_k$. Seja $\mathcal{A}$ uma estrutura e s uma interpretação tal que $\mathcal{A} \models_s \varphi_i$, $1 \leq i \leq k$. Então, também $\mathcal{A} \models_s \varphi \to \theta$ e $\mathcal{A} \models_s \varphi$. Mas então, pela definição (vi) de $\models_s$, tem-se que $\mathcal{A} \models_s \theta$. E então $\{\varphi_1, \ldots, \varphi_k\} \models \theta$.

Vejamos agora o caso da regra da eliminação do quantificador existencial:

$$\dfrac{\exists x\, \varphi \qquad\qquad \begin{array}{c}[v \quad \varphi[v/x]] \\ \vdots \\ \psi\end{array}}{\psi}\ \exists\mathbf{E}:$$

Seja θ a fórmula deduzida no passo n por aplicação de $\exists\mathbf{E}$ a $\exists x\varphi$ e a uma sub-dedução que contêm θ. E sejam $\varphi_1, \ldots, \varphi_k$ as premissas assumidas no passo n, em θ. Por hipótese de indução no passo i, $\exists x\varphi$ é consequência semântica de premissas que são um subconjunto de $\varphi_1, \ldots, \varphi_k$ e no passo m, θ é consequência semântica de premissas que são um subconjunto de $\varphi_1, \ldots, \varphi_k$, *mais a premissa* $\varphi[v/x]$.

$$
\begin{array}{l|l}
\vdots & \vdots \\
i & \exists x\varphi \\
\vdots & \vdots \\
 & \begin{array}{l|l} v & \varphi[v/x] \\ \hline & \vdots \\ m & \theta \\ & \vdots \end{array} \\
\vdots & \vdots \\
n & \theta
\end{array}
$$

Vamos ver que θ é consequência semântica de $\varphi_1, \ldots, \varphi_k$. Seja $\mathcal{A}$ uma estrutura e s uma interpretação tal que $\mathcal{A} \models_s \varphi_i$, $1 \leq i \leq k$. E também $\mathcal{A} \models_s \exists x\varphi$. Isto é, existe um $b \in A$ tal que $\mathcal{A} \models_{s[b/x]} \varphi$. Nota, ainda, que v não pode ocorrer em $\varphi_1, \ldots, \varphi_k$, $\exists x\varphi$ e θ. Seja $s' = s[b/v]$ e, claramente, $\mathcal{A} \models_{s[b/v]} \varphi[v/x]$. Mas também $\mathcal{A} \models_{s'} \varphi_i$, $1 \leq i \leq k$ e, então, $\mathcal{A} \models_{s'} \theta$ (porquê?). E como v não ocorre em θ, então $\mathcal{A} \models_s \theta$.

O caso $\forall$**I** é semelhante e os restantes mais simples.

□

Leituras suplementares [BE00] (Cap. 13,18.3)

2.5.2 Conjuntos consistentes e inconsistentes

Definição 2.14. *Um conjunto de fórmulas Σ diz-se* consistente *se e só se não existe nenhuma dedução de $\Sigma \vdash \boldsymbol{F}$, caso contrário diz-se* inconsistente.

Nota que se Σ é inconsistente, $\Sigma \vdash \psi$ para qualquer fórmula ψ.

Exercício 2.23. *Mostra que um conjunto Σ é inconsistente se e só se existe uma fórmula φ tal que $\Sigma \vdash \varphi$ e $\Sigma \vdash \neg\varphi$.* ⋄

Lema 2.1. *Se $\Sigma \cup \{\neg\varphi\}$ é inconsistente se e só se $\Sigma \vdash \varphi$.*

Demonstração. ($\Rightarrow$) Por aplicação da regra **RA** e ($\Leftarrow$) pela aplicação da regra **FI**. □

Lema 2.2. ***(da dedução)*** *$\Sigma \cup \{\varphi\} \vdash \psi$ se e só se $\Sigma \vdash \varphi \to \psi$.*

Demonstração. Igual ao da lógica proposicional. □

2.5.3 Completude do sistema dedutivo DN

Vamos considerar apenas o caso de $\mathcal{L}$ ser uma linguagem numerável, isto é, em que o seu alfabeto (de variáveis, símbolos funcionais e de predicado) é numerável (não necessariamente finito).

Teorema 2.1. *(Teorema da completude de Gödel) Se* $\Sigma \models \varphi$ *então* $\Sigma \vdash \varphi$*, ou,* equivalentemente*, qualquer conjunto consistente de fórmulas é satisfazível.*

Começamos por mostrar a equivalência entre essas duas afirmações:
(1⇐ 2) Se $\Sigma \models \varphi$ então $\Sigma \cup \{\neg\varphi\}$ não é satisfazível. Por 2, $\Sigma \cup \{\neg\varphi\}$ é inconsistente, logo $\Sigma \vdash \varphi$.
(1⇒ 2) Se $\Sigma \cup \{\varphi\}$ é não satisfazível, então $\Sigma \models \neg\varphi$. Por 1, $\Sigma \vdash \neg\varphi$ e então $\Sigma \cup \{\neg\neg\varphi\}$ é inconsistente, mas então também o é $\Sigma \cup \{\varphi\}$.
Iremos mostrar a segunda afirmação...
Temos que encontrar uma estrutura e interpretação que satisfaça um qualquer conjunto consistente..o que não parece tarefa fácil. A demonstração a apresentar é baseada na de L. Henkin (e não na de Gödel...).
A ideia é construir uma estrutura cujo domínio seja o conjunto dos termos da linguagem $\mathcal{L}$. Mas a presença de quantificadores traz dificuldades acrescidas. Por exemplo,

$$\Delta = \{\exists x P(x)\} \cup \{\neg P(t) : t \text{ é um termo de} \mathcal{L}\}$$

é consistente mas não é possível satisfazê-lo com um domínio só com termos de $\mathcal{L}$. E também é necessário relacionar a satisfazibilidade de $P(t)$ e a de $\exists x P(x)$ (o que não é possível, p.e, só com a validade booleana das fórmulas...)
Assim iremos introduzir novas constantes em $\mathcal{L}$, que irão servir para construir **testemunhas** (de uma fórmula existencial) p.e.:

$$\exists x\ P(x) \to P(c_P)$$

E iremos alargar qualquer conjunto consistente de modo a conter todas as *testemunhas* necessárias e a continuar consistente...
Dizemos que um conjunto consistente de fórmulas Σ de uma linguagem $\mathcal{L}$ é **maximal** se para toda a fórmula φ de $\mathcal{L}$ se tem ou $\varphi \in \Sigma$ ou $\neg\varphi \in \Sigma$.

Lema 2.3. *Para qualquer conjunto consistente de fórmulas* Σ *existe um conjunto* $\Delta \supseteq \Sigma$*, da linguagem alargada* $\mathcal{L}' = \mathcal{L} \cup \{c_0, c_1, \ldots\}$ *onde* c_i *são constantes novas e*

1. Δ *é consistente e maximal em* $\mathcal{L}'$

2. *Para qualquer fórmula φ de $\mathcal{L}'$ e qualquer variável $x \in Var$ existe uma constante c tal* $\exists x\varphi \to \varphi[c/x] \in \Delta$

Demonstração. (do lema 2.3)

Σ é consistente em $\mathcal{L}'$ Suponhamos, por contradição que $\Sigma \vdash \mathbf{F}$ (ou alternativamente uma qualquer fórmula da forma $\beta \wedge \neg\beta$). Mas nessa dedução só pode ocorrer um número finito de constantes novas, $c_1, \ldots c_k$. Então podemos substituir cada uma dessas constantes por uma **variável nova**, sejam $y_1, \ldots, y_k$. A dedução resultante é uma dedução de $\mathbf{F}$ a partir de Σ em $\mathcal{L}$. Absurdo! Porque Σ é consistente em $\mathcal{L}$.

Vamos construir Θ tal que $\Sigma \cup \Theta$ satisfaz 2 Como $\mathcal{L}'$ é numerável, então também o são Var e o conjunto das suas fórmulas. Podemos então enumerar $< \varphi_1, x_1 >, < \varphi_2, x_2 >, \ldots$ Seja θ_1 a fórmula $\exists x_1\varphi_1 \to \varphi_1[c_1/x_1]$, onde c_1 é uma constante nova que não ocorre em φ_1. Para $n > 1$, θ_n é $\exists x_n\varphi_n \to \varphi_n[c_n/x_n]$, onde c_n é uma constante nova que não ocorre nem φ_n nem em $\theta_1 \ldots \theta_{n-1}$. Seja

$$\Theta = \{\theta_1, \theta_2, \ldots\}$$

$\Sigma \cup \Theta$ é consistente Suponhamos que não, então existe $m \geq 0$ tal que $\Sigma \cup \{\theta_1, \ldots, \theta_{m+1}\}$ é inconsistente e $\Sigma \cup \{\theta_1, \ldots, \theta_m\}$ consistente. Mas, então

$$\Sigma \cup \{\theta_1, \ldots, \theta_m\} \vdash \neg\theta_{m+1}$$

Como θ_{m+1} é da forma $\exists x\varphi \to \varphi[c/x]$, então também (mostra!)

$$\Sigma \cup \{\theta_1, \ldots, \theta_m\} \vdash \exists x\varphi$$
$$\Sigma \cup \{\theta_1, \ldots, \theta_m\} \vdash \neg\varphi[c/x]$$

mas como c não ocorre em $\Sigma \cup \{\theta_1, \ldots, \theta_m\}$, podemos substituí-lo por uma variável nova e usando a regra $\forall\mathbf{I}$, concluímos também que

$$\Sigma \cup \{\theta_1, \ldots, \theta_m\} \vdash \forall x\neg\varphi$$

mas como $\forall x\ \neg\varphi \dashv\vdash \neg\exists x\varphi$, vem

$$\Sigma \cup \{\theta_1, \ldots, \theta_m\} \vdash \neg\exists x\varphi$$

Absurdo! porque $\Sigma \cup \{\theta_1, \ldots, \theta_m\}$ é consistente.

Estendemos $\Sigma \cup \Theta$ a um conjunto consistente maximal Δ Seja φ_1, φ_2, ... uma enumeração das fórmulas de $\mathcal{L}'$ e definimos

$$\Delta = \bigcup_n \Delta_n$$

onde

$$\begin{array}{rcl} \Delta_0 & = & \Sigma \cup \Theta \\ \Delta_{n+1} & = & \begin{cases} \Delta_n \cup \{\varphi_{n+1}\} & \text{se este conjunto é consistente} \\ \Delta_n \cup \{\neg\varphi_{n+1}\} & \text{caso contrário} \end{cases} \end{array}$$

Mostra-se por indução sobre n que os Δ_n são consistentes (mostrar!). Logo Δ também é consistente e maximal.

□

Exercício 2.24. *Se Δ é um conjunto consistente maximal e $\Delta \vdash \varphi$ então $\varphi \in \Delta$* $\diamond$

Para um conjunto de fórmulas Δ do lema anterior, vamos construir uma estrutura $\mathcal{A}_\Delta = (A, \cdot^{\mathcal{A}_\Delta})$ e uma interpretação s_Δ tal que

$$\mathcal{A}_\Delta \models_{s_\Delta} \psi, \text{ para todo } \psi \in \Delta$$

Considere-se o conjunto de termos $\mathcal{T}$ e seja $\sim$ a relação binária dada por

$$t_1 \sim t_2 \text{ se e só se } t_1 = t_2 \in \Delta$$

Pela definição de Δ o fecho reflexivo de $\sim$ é uma relação de equivalência em $\mathcal{T}$, que designamos por $\sim_\Delta$
A estrutura $\mathcal{A}_\Delta$ é:

- o domínio é o conjunto das classes de equivalência de $\sim_\Delta$
- para cada símbolo relacional $R \in \mathcal{R}_n$, $n > 1$:

$$R^{\mathcal{A}_\Delta} = \{([t_1], \ldots, [t_n]) \mid R(t_1, \ldots, t_n) \in \Delta\}$$

- para cada símbolo funcional $f \in \mathcal{F}_n$, $n > 0$ tem-se:

$$f^{\mathcal{A}_\Delta}([t_1], \ldots, [t_n]) = [f(t_1, \ldots, t_n)]$$

- para cada constante c:

$$c^{\mathcal{A}_\Delta} = [c]$$

Para cada $x \in Var$, $s_\Delta(x) = [x]$

Lema 2.4. *Para qualquer termo $t \in \mathcal{T}$, tem-se que $s_\Delta(t) = [t]$*

Demonstração. Por indução sobre t.

Base. Para as variáveis e constantes por definição de s_Δ.

Indução. Se $f \in \mathcal{F}_n$ e $t_1, \ldots, t_n \in \mathcal{T}$,

$$\begin{aligned} s_\Delta(f(t_1, \ldots, t_n)) &= f^{\mathcal{A}_\Delta}(s_\Delta(t_1), \ldots, s_\Delta(t_n)) \\ &= f^{\mathcal{A}_\Delta}([t_1], \ldots, [t_n]) \\ &= [f(t_1, \ldots, t_n)] \end{aligned}$$

□

Lema 2.5. *Seja Δ um conjunto de fórmulas nas condições do lema 2.3. Então*

$$\mathcal{A}_\Delta \models_{s_\Delta} \psi \quad \textit{se e só se } \psi \in \Delta$$

Demonstração. Por indução sobre ψ.

Base. ψ é atómica. Se ψ é $t_1 = t_2$, então $\mathcal{A}_\Delta \models_{s_\Delta} t_1 = t_2$ sse $s_\Delta(t_1) = s_\Delta(t_2)$ sse $[t_1] = [t_2]$, i.e, $t_1 = t_2 \in \Delta$. Se ψ é $R(t_1, \ldots, t_n)$ então $\mathcal{A}_\Delta \models_{s_\Delta} R(t_1, \ldots, t_n)$ sse $(s_\Delta(t_1), \ldots, s_\Delta(t_n)) \in R^{\mathcal{A}_\Delta}$ i.e $([t_1], \ldots, [t_n]) \in R^{\mathcal{A}_\Delta}$ ou $R(t_1, \ldots, t_n) \in \Delta$.

Indução. Se é ψ é $\neg\varphi$, então $\mathcal{A}_\Delta \models_{s_\Delta} \neg\varphi$ sse $\mathcal{A}_\Delta \not\models_{s_\Delta} \varphi$, e por hipótese de indução $\varphi \notin \Delta$, logo por Δ ser consistente maximal, $\neg\varphi \in \Delta$. Se ψ é $\varphi_1 \vee \varphi_2$ então,$\mathcal{A}_\Delta \models_{s_\Delta} \varphi_1 \vee \varphi_2$ sse $\mathcal{A}_\Delta \models_{s_\Delta} \varphi_i$ pelo menos para um i, e por hipótese de indução $\varphi_i \in \Delta$, mas então por Δ ser consistente maximal, $\varphi_1 \vee \varphi_2 \in \Delta$. Analogamente para ψ é $\varphi_1 \wedge \varphi_2$. Se ψ é $\varphi \to \theta$ então, $\mathcal{A}_\Delta \models_{s_\Delta} \varphi \to \theta$ sse $\mathcal{A}_\Delta \not\models_{s_\Delta} \varphi$ ou $\mathcal{A}_\Delta \models_{s_\Delta} \theta$, e por hipótese de indução ou $\varphi \notin \Delta$ ou $\theta \in \Delta$,mas então por Δ ser consistente maximal, $\varphi \to \theta \in \Delta$. Se ψ é $\forall x\varphi$, tem-se que $\exists x\neg\varphi \to \neg\varphi[c/x] \in \Delta$, para algum c. Logo

$$\begin{aligned} \forall x\varphi \notin \Delta \quad & \textit{sse} \quad \exists x\neg\varphi \in \Delta \\ & \textit{sse} \quad \neg\varphi[c/x] \in \Delta \\ & \textit{sse} \quad \varphi[c/x] \notin \Delta \\ & \textit{sse} \quad \mathcal{A}_\Delta \not\models_{s_\Delta} \varphi[c/x] \\ & \textit{donde} \quad \mathcal{A}_\Delta \not\models_{s_\Delta} \forall x\varphi \end{aligned}$$

Reciprocamente, se $\mathcal{A}_\Delta \not\models_{s_\Delta} \forall x\varphi$, existe t tal que $\mathcal{A}_\Delta \not\models_{s_\Delta[[t]/x]} \varphi$. Existe θ equivalente a φ e onde x é substituível por t em θ, logo $\mathcal{A}_\Delta \not\models_{s_\Delta[[t]/x]} \theta$ e $\mathcal{A}_\Delta \not\models_{s_\Delta} \theta[t/x]$. Por hipótese de indução, $\theta[t/x] \notin \Delta$ e portanto $\forall x\theta \notin \Delta$. E como θ e φ são equivalentes e Δ maximal, $\forall x\varphi \notin \Delta$. Analogamente se ψ é $\exists x\varphi$.

□

Demonstração. (**Teorema da completude** 2.1) Mostrar que dado um conjunto consistente Σ ele é satisfazível. Estendemos $\Sigma \subseteq \Delta$ nas condições do lema 2.3 para $\mathcal{L}'$ e sejam $\mathcal{A}_\Delta$ e s_Δ a estrutura e a interpretação que satisfazem Δ em $\mathcal{L}'$, pelo lema 2.5. A restrição $\mathcal{A}_\Sigma$ de $\mathcal{A}_\Delta$ a $\mathcal{L}$ satisfaz Σ. □

2.5.4 Consequências da completude e integridade

Corolário 2.2. ***(Teorema da compacidade)***

1. $\Sigma \models \varphi$ *se e só se existe um subconjunto finito* $\Sigma_0 \subseteq \Sigma$ *tal que* $\Sigma_0 \models \varphi$.

2. *Um conjunto de fórmulas duma linguagem de* $1^{\underline{a}}$ *ordem é satisfazível se e só se todo o seu subconjunto finito o for.*

Demonstração. As duas afirmações são equivalentes (verifica!), portanto basta demonstrar a primeira. Se $\Sigma \models \varphi$ então $\Sigma \vdash \varphi$. Como qualquer dedução só utiliza um número finito de premissas (hipóteses), existe $\Sigma_0 \subseteq \Sigma$ finito tal que $\Sigma_0 \vdash \varphi$ ou seja $\Sigma_0 \models \varphi$. □

Vamos ver uma aplicação deste teorema que mostra as limitações da expressividade da lógica de primeira ordem. Em particular, que não existe um conjunto de fórmulas que caracterize na estrutura dos números naturais.

Exemplo 2.13. *Modelos não* standard *para a aritmética*
Seja $\mathcal{L}_N$ *a linguagem de* $1^{\underline{a}}$ *ordem com igualdade para os números naturais já dada:* $\mathcal{F}_0 = \{0, 1\}$, $\mathcal{F}_2 = \{+, \times\}$ *e* $\mathcal{R}_2 = \{<\}$
Seja $\mathcal{N} = (\mathbb{N}, \cdot^{\mathcal{N}})$ *uma estrutura de* $\mathcal{L}_N$, *onde* $\cdot^{\mathcal{N}}$ *é definido por:*

- $0^{\mathcal{N}} = 0$, $1^{\mathcal{N}} = 1$
- $+^{\mathcal{N}}(n, m) = n + m$, $\times^{\mathcal{N}}(n, m) = n \times m$
- $<^{\mathcal{N}} = \{(n, m) \in \mathbb{N}^2 \mid n < m\}$

Podemos provar, por exemplo que, $\mathcal{N} \models \forall x\ x < x + 1$ *(verifica!)*

A estrutura $\mathcal{N}$ é designada a estrutura standard *de $\mathcal{L}_N$, pois $\mathcal{L}_N$ foi construída para se poder falar das propriedades dos números naturais. Mas podemos escolher outras estruturas para $\mathcal{L}_N$:*
Por exemplo, para $p > 1$, $\mathcal{N}_p = (\{0, \ldots, p-1\}, \cdot_p^{\mathcal{N}})$ com

- $0^{\mathcal{N}_p} = 0$, $1^{\mathcal{N}_p} = 1$
- $+^{\mathcal{N}_p}(n, m) = (n + m) \mod p$
- $\times^{\mathcal{N}_p} = (n \times m) \mod p$

E, neste caso $\mathcal{N}_p \not\models \forall x\ x < x + 1$ (verifica!)
Isto é, existe uma fórmula da lógica de $1^{\underline{a}}$ ordem que permite distinguir a estrutura $\mathcal{N}$ da $\mathcal{N}_p$. No entanto, usando o teorema da compacidade podemos demonstrar que nem sempre é possível ***distinguir*** *por uma fórmula duas estruturas para a mesma linguagem.*
Seja $\mathcal{N}' = (\mathbb{N} \cup \{n + i \mid n \in \mathbb{N}\}, \cdot^{\mathcal{N}'})$ a estrutura de $\mathcal{L}_N$, onde $i = \sqrt{-1}$ e, por exemplo, para o símbolo funcional $+$ temos:

- $+^{\mathcal{N}_p}(n, m) = (n + m)$
- $+^{\mathcal{N}_p}(n + i, m + i) = (n + m) + i$

Vamos ver que $\mathcal{N}'$ não pode ser distinguido de $\mathcal{N}$ por uma proposição de $\mathcal{L}_N$ (ou doutra linguagem de $1^{\underline{a}}$ ordem).

Corolário 2.3. *Se Σ é um conjunto de proposições tal que $\mathcal{N} \models \Sigma$ então existe um modelo $\mathcal{N}'$ tal que $\mathcal{N}' \models \Sigma$ e o domínio de $\mathcal{N}$ é um subconjunto próprio de $\mathcal{N}'$.*

Demonstração. Considera as proposições φ_n dadas por

$$\exists x((x \neq 0) \ \wedge \ (x \neq 1) \ \wedge \ \ldots \ \wedge \ (x \neq n))$$

Então o conjunto $\Sigma \cup \{\varphi_n \mid n \geq 0\}$ é consistente. Porque se não o fosse, teria um subconjunto finito que era inconsistente. Esse conjunto finito conteria um número finito dos φ_n. Mas obviamente $\mathcal{N}$ satisfaz Σ e qualquer conjunto finito de φ_n. Absurdo!
Portanto o conjunto $\Sigma \cup \{\varphi_n \mid n \geq 0\}$ é consistente, logo é satisfazível. Isto é, tem de existir uma estrutura cujo domínio seja um superconjunto do de $\mathcal{N}$ e que seja um modelo desse conjunto, p.e, $\mathcal{N}'$. □

Corolário 2.4. ***(Teorema de Löwenheim-Skolem I)*** *Se um conjunto de fórmulas Σ é satisfazível por uma estrutura, então Σ é satisfazível por uma estrutura com um domínio numerável.*

Demonstração. Porque o domínio da estrutura construída na demonstração do teorema da completude é numerável. □

Sendo um modelo numerável ele pode ser finito ou infinito. Mas será que todas as proposições têm um modelo numerável infinito?
Sejam, por exemplo, $\forall x \forall y x = y$ ou $\exists x \exists y \forall z (z = x \ \vee \ z = y)$.
Estas proposições não têm modelos infinitos! A primeira não aceita modelos com domínios de cardinalidade maior que 1, e a segunda maior que 2.

Corolário 2.5. ***(Teorema de Löwenheim-Skolem II)*** *Se uma proposição tem um modelo finito de cardinalidade arbitrariamente grande, então tem um modelo infinito.*

Demonstração. Seja a proposição ψ_k, para $k > 1$

$$\exists x_1 \ldots \exists x_k \bigwedge_{1 \leq i \leq j \leq k} x_i \neq x_j$$

ψ_k indica que existem pelo menos k objectos diferentes no domínio; não pode ser satisfeita por uma estrutura com menos de k elementos e todas as estruturas com mais elementos a satisfazem. Suponhamos, por contradição, que existe uma proposição φ que tem modelos arbitrariamente grandes, mas nenhum modelo infinito. Seja o conjunto

$$\Sigma = \{\varphi\} \cup \{\psi_k \mid k \in \mathbb{N} \setminus \{0, 1\}\}$$

Se Σ tem um modelo M, então M não pode ser finito (seja k, então ψ_{k+1} não era satisfeita), e não pode ser infinito (satisfaria φ). Então, Σ não tem modelo.
Pela compacidade, existe um conjunto finito $D \subset \Sigma$ que não tem modelo. D tem de conter φ (senão haveria um modelo suficientemente grande que continha todos os ψ_k de D). Seja k o maior inteiro tal que $\psi_k \in D$. Por hipótese, φ tem um modelo finito de cardinalidade maior que k. Então esse modelo satisfaz todas as proposições de D. Absurdo! □

Este teorema permite ilustrar mais limitações da expressividade da lógica de 1ª ordem. Em particular, temos:

Corolário 2.6. *Não existe nenhuma fórmula φ (com duas variáveis livres) tal que saber se existe uma estrutura $\mathcal{G}$ da linguagem $\mathcal{L}_G$, tal que $\mathcal{G} \models \varphi$ equivale a determinar se:*

> *Dado um grafo dirigido (finito) G e dois nós x e y de G, existe um caminho de x para y.*

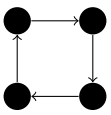
Demonstração. Suponhamos que existe uma tal fórmula φ. Seja ψ_0 a fórmula $\forall x \forall y \varphi$. A fórmula ψ_0 indica que G é fortemente conexo. Seja ainda ψ_1 a fórmula (que corresponde a *todos os nós têm grau de saída 1*)

$$\forall x \exists y G(x,y) \ \wedge \ \forall x \forall y \forall z ((G(x,y) \ \wedge \ G(x,z)) \rightarrow y = z)$$

e ψ_2 a fórmula (que corresponde a *todos os nós têm grau de entrada 1*)

$$\forall x \exists y G(y,x) \ \wedge \ \forall x \forall y \forall z ((G(y,x) \ \wedge \ G(z,x)) \rightarrow y = z)$$

E seja ψ a proposição $\psi_0 \ \wedge \ \psi_1 \ \wedge \ \psi_2$. Os grafos que satisfazem ψ dizem-se *ciclos*.



Obviamente, existem *ciclos* finitos com número arbitrário de nós. Pelo teorema de Löwenheim-Skolem, ψ tem um modelo infinito, seja G_∞. Mas ciclos infinitos (i.e com um número infinito de nós) não existem!: seja n_0 um nó de G_∞ e consideremos todos os nós atingíveis de n_0. Como é fortemente conexo esse conjunto incluí todos os nós do grafo. Mas como o grau de entrada de n_0 é 1, existe um nó, seja n_j tal que (n_j, n_0) é um arco de G_∞. Mas então o ciclo é finito. Absurdo! □

Do mesmo modo, se uma linguagem da lógica de $1^{\underline{a}}$ ordem tivesse um símbolo de predicado que pretendesse significar `é antepassado de`, então não haveria nenhum conjunto de proposições que "capturasse" este conceito pois existiria sempre um modelo que permitiria *antepassados infinitamente distantes....*

Teorema 2.2. ***(da compacidade da lógica proposicional)*** *Um conjunto Σ de fórmulas da lógica proposicional é satisfazível se e só se todo o seu subconjunto finito o for.*

Demonstração. Seja $\mathcal{L}_{Prop}$ uma linguagem de $1^{\underline{a}}$ ordem sem igualdade e um símbolo de predicado unário P, como único símbolo não-lógico. Seja $A = \{P(x_i) \mid i \in \mathbb{N}\}$ o conjunto de fórmulas atómicas de $\mathcal{L}_{Prop}$ e seja $\pi : A \longrightarrow \mathcal{V}_{Prop}$ tal que $\pi(P(x_i)) = p_i$, $i \in \mathbb{N}$. Podemos estender π a uma função bijectiva entre o conjunto das fórmulas de $\mathcal{L}_{Prop}$ sem quantificadores e as fórmulas proposicionais, obtendo a *forma booleana* das primeiras fórmulas. Em particular, um conjunto Σ de fórmulas de $\mathcal{L}_{Prop}$ sem quantificadores é *satisfazível* se e só se o conjunto $\pi(\Sigma)$ das respectivas formas booleanas o for.
($\Leftarrow$) Seja Σ um conjunto de fórmulas proposicionais, tal que todo o subconjunto finito $\Sigma_0 \subseteq \Sigma$ é satisfazível. Então, todo o subconjunto finito de $\pi^{-1}(\Sigma)$ é satisfazível. Pela compacidade, $\pi^{-1}(\Sigma)$ é satisfazível. Mas então também $\pi(\pi^{-1}(\Sigma)) = \Sigma$ é satisfazível
($\Rightarrow$) Trivial. □

Exercício 2.25. *(Aplicação do teorema da compacidade)*
Seja $\mathcal{L}$ uma linguagem de $1^{\underline{a}}$ ordem com igualdade.

1. *Para cada $n \geq 1$, construir uma proposição φ_n de $\mathcal{L}$ tal que $\mathcal{A} \models \varphi_n$ se e só se o domínio de $\mathcal{A}$ tem pelo menos n elementos.*

   Resolução

   Para cada i, φ_i é a fórmula

   $$\exists x_1 \ldots \exists x_n \bigwedge_{1 \leq i \leq j \leq n} x_i \neq x_j$$

2. *Seja $\Sigma = \cup_{i \geq 1} \{\varphi_i\}$. Quanto é que uma estrutura de $\mathcal{L}$ é um modelo de Σ?*

   Resolução

   Quando o seu domínio é infinito (pelo menos numerável)

3. *Suponhamos que φ é uma proposição tal que $\mathcal{A}$ é um modelo de φ se e só se o domínio de $\mathcal{A}$ é infinito. Mostra que $\Sigma \models \varphi$.*

   Resolução

   Se $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ satisfaz Σ então A é um conjunto infinito. Mas então satisfaz φ.

4. *Justifica que φ não pode existir.*

   Resolução

   Pela compacidade, $\Sigma \models \varphi$ se e só se existe $\Sigma_0 \subseteq \Sigma$ finito tal que $\Sigma_0 \models \varphi$. Seja n_0 o maior valor tal que $\varphi_{n_0} \in \Sigma_0$, então qualquer estrutura $\mathcal{A} = (A, \cdot^{\mathcal{A}})$, com $|A| = n_0$ satisfaz Σ_0. Logo,, $\mathcal{A}$ satisfaz φ. Absurdo!

◇

Leituras suplementares [BE00] (Cap. 19)

2.6 Axiomatizações e teorias

Normalmente, para além de fórmulas válidas, estamos interessados na satisfazibilidade de fórmulas numa dada estrutura (ou na sua validade numa dada estrutura). Mas para que possamos ter um sistema dedutivo associado, interessa ter um *conjunto de fórmulas* que sejam válidas nessa estrutura e a partir das quais se possam obter todas as consequências semânticas. Essas fórmulas são premissas nos sistemas dedutivos e são designadas de *axiomas não-lógicos*. Para além de válido numa dada estrutura $\mathcal{A}$ (e portanto consistente), um conjunto de axiomas Σ deve ser completo: se uma fórmula φ for válida em $\mathcal{A}$ então φ é consequência semântica de Σ. Dada a completude e integridade, isso equivale a que para toda a fórmula φ, ou φ ou $\neg\varphi$ seja dedutível de Σ.

Exemplo 2.14. *No* Mundo dos Blocos*, onde há objectos geométricos de três formas possíveis (cubos, tetraedros e dodecaedros), uma axiomatização da noção de* forma *é dada por:*

1. $\neg\exists x(Cubo(x) \wedge Tetra(x))$
2. $\neg\exists x(Cubo(x) \wedge Dodec(x))$
3. $\neg\exists x(Dodec(x) \wedge Tetra(x))$
4. $\forall x(Cubo(x) \vee Dodec(x) \vee Tetra(x))$
5. $\forall x\forall y((Cubo(x) \wedge Cubo(y)) \rightarrow MesmaF(x,y))$
6. $\forall x\forall y((Tetra(x) \wedge Tetra(y)) \rightarrow MesmaF(x,y))$
7. $\forall x\forall y((Dodec(x) \wedge Dodec(y)) \rightarrow MesmaF(x,y))$
8. $\forall x\forall y((Cubo(x) \wedge MesmaF(x,y)) \rightarrow Cubo(y))$
9. $\forall x\forall y((Tetra(x) \wedge MesmaF(x,y)) \rightarrow Tetra(y))$
10. $\forall x\forall y((Dodec(x) \wedge MesmaF(x,y)) \rightarrow Dodec(y))$

As primeiras 3 proposições indicam que um objecto não pode ter duas formas; a quarta que cada objecto tem uma forma; e as restantes indicam que dois objectos têm a mesma forma se e só se são os dois cubos, tetraedros ou dodecaedros.

Exemplo 2.15. ***Teoria dos grupos*** *Considera uma linguagem de* $1^{\underline{a}}$ *ordem (com* $=$*), com um símbolo funcional binário* $\circ$ *e uma constante* 1*. A noção de* Grupo *da teoria dos grupos é axiomatizada completamente pelos seguintes axiomas não-lógicos (dos quais se podem deduzir todas as propriedades de um grupo):*

1. $\forall x\forall y\forall z\ (x \circ y) \circ z = x \circ (y \circ z)$
2. $\forall x\ (x \circ 1) = x$
3. $\forall x\exists y\ (x \circ y) = 1$

Exemplos de outras axiomatizações:

- Axiomas da Geometria de Euclides
- Axiomas de Zermello-Frankel para a teoria dos conjuntos
- Axiomas de Peano para a teoria dos números naturais

2.6.1 Teoria *ingénua* dos conjuntos

Um conjunto é uma colecção de elementos. Seja uma linguagem de 1ª ordem com igualdade que tem um símbolo relacional binário $\in$. Para distinguir entre conjuntos e elementos, usaremos variáveis a, b, c, ... para conjuntos e x, y, z, ... para elementos. Consideremos então os seguintes axiomas:

Axioma da extensionalidade $\forall\, a \forall\, b\ (\forall x\ (x \in a \leftrightarrow x \in b) \rightarrow a = b)$

> Um conjunto é completamente determinado pelos seus elementos; assim, conjuntos com os mesmos elementos são iguais.

(Esquema de) Axioma da compreensão $\forall \exists a \forall x (x \in a \leftrightarrow \varphi(x))$

> Cada propriedade determina um conjunto, ie, cada fórmula determina um conjunto: o dos elementos que têm essa propriedade. A quantificação sem variáveis corresponde a quantificar universalmente todas as variáveis que ocorram em φ.

A partir destes axiomas, podem-se também definir (e deduzir) fórmulas correspondentes à relação $\subset$, às operações $\cap$ e $\cup$, conjunto potência, ... Por exemplo,

Inclusão $\forall x(x \in a \rightarrow x \in b)$, $(a \subseteq b)$

União $\forall a \forall b \forall z((z \in a \cup b) \leftrightarrow (z \in a\ \vee\ z \in b))$, $(a \cup b)$

Intersecção $\forall a \forall b \forall z((z \in a \cap b) \leftrightarrow (z \in a\ \wedge\ z \in b))$, $(a \cap b)$

Conjunto potência $\forall b \exists c \forall x(x \in c \leftrightarrow x \subseteq b)$, $(\mathcal{P}(b))$ (para qualquer conjunto existe um único conjunto cujos elementos são os subconjuntos desse conjunto)

e também se podem deduzir propriedades destas operações.

Proposição 2.11. *Para qualquer b, é falso que $P(b) \subseteq b$.*

Demonstração. Pretende-se provar que, para qualquer b, $P(b) \not\subseteq b$, isto é, que existe um subconjunto de b que não pertence a b. Seja

$$c = \{x \mid x \in b\ \wedge\ x \notin x\}$$

Pelo axioma da compreensão c é um conjunto e $c \subset b$, logo $c \in P(b)$. Vamos ver que $c \notin b$. Suponhamos que $c \in b$. Então ou $c \in c$ ou $c \notin c$. Mas facilmente se vê que ambas as hipóteses não se podem verificar. □

Proposição 2.12. *Existe um conjunto c, tal que $P(c) \subseteq c$.*

Demonstração. Pelo axioma da compreensão existe um conjunto (universal) que contém tudo:

$$c = \{x \mid x = x\}$$

Mas então qualquer subconjunto de c é um elemento de c, portanto $P(c) \in c$. □

Acabámos de ver que se podem deduzir as fórmulas $\forall b \neg(P(b) \subseteq b)$ e $\exists b P(b) \subseteq b$
Donde os axiomas são **inconsistentes**! (Logo não satisfazíveis...)
Esta contradição está relacionada com o paradoxo de Russell:

$$Z = \{x \mid x \notin x\} \text{ e considerar } Z \in Z$$

Podemos então concluir que o **Axioma da compreensão** não é válido: a propriedade *não pertencer a si próprio* não determina nenhum conjunto.

2.6.2 Teoria de conjuntos de Zermelo-Frankel

Para evitar a inconsistência da teoria ingénua dos conjuntos, os axiomas usualmente adoptados para a teoria dos conjuntos são os de Zermelo-Frankel. Mantém-se o axioma da extensionalidade mas o axioma da compreensão é substituído por vários outros, que evitam a possibilidade de referência a *conjuntos tão grandes* como o *conjunto de todos os conjuntos*.

Axioma da extensionalidade $\forall\, a \forall\, b\ (\forall x\ (x \in a \leftrightarrow x \in b) \rightarrow a = b$

Axioma da separação $\forall \forall a \exists b \forall x\ (x \in b \leftrightarrow (x \in a\ \wedge\ \varphi(x))$ Esta é a versão mais fraca do Axioma da compreensão. Só se podem formar subconjuntos de conjuntos já existentes. Mas este axioma, embora torne a teoria consistente, é muito restritivo. Nem sequer permite deduzir que a reunião de dois conjuntos é um conjunto. Assim foi necessário acrescentar mais axiomas.

Axioma dos pares $\forall u \forall v \exists a \forall x (x \in a \leftrightarrow (x = u\ \vee\ x = v))$ (para cada dois elementos existe um conjunto a que os dois pertencem)

Axioma da união $\forall a \exists b \forall x (x \in b \leftrightarrow \exists c (c \in a\ \wedge\ x \in c))$ (para qualquer conjunto a de conjuntos , a união de todos os elemntos de a também são um conjunto)

Axioma da potência $\forall b \exists c \forall x (x \in c \leftrightarrow x \subseteq b)$ (conjunto potência)

Axioma da infinitude Existe o conjunto de todos os números naturais.

Axioma da substituição Se $\forall a (\forall x (x \in a\ \wedge\ (\exists! y \varphi(x, y))) \rightarrow \exists b \forall x (x \in b \leftrightarrow \exists t\ (t \in a\ \wedge\ \varphi(t, x))))$

Axioma da regularidade $\forall z(z \neq \emptyset \rightarrow \exists y(y \in z \ \wedge \ \forall x(x \in z \rightarrow x \notin y)))$ (nenhum conjunto tem uma intersecção não vazia com cada um dos seus elementos)

Axioma da escolha Se f é uma função com um domínio a não vazio e para $x \in a$, $f(x)$ é um conjunto não vazio então existe uma função g de domínio a tal que para cada $x \in a$, $g(x) \in f(x)$. (g escolhe elementos de cada $f(x)$)

2.6.3 Axiomas para a teoria dos números (aritmética)

Seja uma linguagem de 1ª ordem com igualdade, $\mathcal{F}_0 = \{0, 1\}$ e $\mathcal{F}_2 = \{+, \times\}$. Podemos omitir o operador relacional $\leq$ uma vez que essa noção se pode definir a partir da igualdade:
$x \leq y$ é $\exists z x + z = y$
$x < y$ é $\exists z(x + z = y \ \wedge \ z \neq 0)$
Já vimos que a esta linguagem permite definir fórmulas que caracterizam propriedades dos números naturais, i.e, da estrutura $\mathcal{N} = (\mathbb{N}, .^{\mathcal{A}})$ onde as interpretações dos símbolos não-lógicos correspondem às operações aritméticas usuais.
Alguns desses conceitos podem-se exprimir pelas seguintes fórmulas:

Divisão inteira: $x = q \times y + r \ \wedge \ r < y =_{def} INTDIV(x, y, q, r)$

y **divide** x**:** $\exists q INTDIV(x, y, q, 0) =_{def} DIV(y, x)$

$2 : \ 1 + 1 =_{def} 2$

x **é par:** $DIV(2, x) =_{def} Par(x)$

x **é ímpar:** $\neg Par(x) =_{def} Impar(x)$

x **é primo:** $\neg x < 2 \ \wedge \ \forall y(DIV(y, x) \rightarrow (y = 1 \ \vee \ y = x)) =_{def} Primo(x)$

x **é potência de 2:** $\forall y((DIV(y, x) \ \wedge \ Primo(x)) \rightarrow y = 2) =_{def} P(2, x)$

y **é** 2^k **e o** k**-ésimo bit de** x **é 1:** $P(2, y) \ \wedge \ \forall q \forall r(INTDIV(x, y, q, r) \rightarrow Impar(q)) =_{def} BIT(x, y)$

Os axiomas de Peano (**PA**) são factos básicos dos números naturais:

1. $\forall x(x + 1 \neq 0)$ (0 não é o sucessor de nenhum natural)
2. $\forall x \forall y(x + 1 = y + 1 \rightarrow x = y)$ (o sucessor é injectivo)
3. $0 + 1 = 1$
4. $\forall x \ x + 0 = x$ (0 é identidade para $+$)

5. $\forall x \forall y\ x + (y + 1) = (x + y) + 1$ (+ é associativo)

6. $\forall x\ x \times 0 = 0$ (0 é absorvente de $\times$)

7. $\forall x \forall y\ x \times (y + 1) = (x \times y) + x$ (distributividade)

8. (princípio da indução) $(\varphi[0/x] \ \wedge\ (\forall x(\varphi \to \varphi[x + 1/x]) \to \forall x\varphi$

É fácil de verificar que os axiomas de Peano são válidos em $\mathcal{N}$. Assim, pela integridade da dedução natural, todas as proposições φ tal que $\mathbf{PA} \vdash \varphi$, são válidas em $\mathcal{N}$.

Exercício 2.26. *Mostrar que os axiomas de Peano são válidos em* $\mathcal{N}$. $\diamond$

Exercício 2.27. *Mostrar que* $\mathbf{PA} \vdash \forall x(x + 1 = 1 + x)$. $\diamond$

Resolução 2.27

1	$\forall x(x + 1 \neq 0)$	
2	$\forall x \forall y(x + 1 = y + 1 \to x = y)$	
3	$0 + 1 = 1$	
4	$\forall x\ x + 0 = x$	
5	$\forall x \forall y\ x + (y + 1) = (x + y) + 1$	
6	$\forall x\ x \times 0 = 0$	
7	$\forall x \forall y\ x \times (y + 1) = (x \times y) + x$	
8	$(\varphi[0/x] \ \wedge\ (\forall x(\varphi \to \varphi[x + 1/x]))) \to \forall x\varphi$	
9	$1 + 0 = 1$	$\forall$E, 4
10	$0 + 1 = 1 + 0$	$=$**I**, 3, 9
11	u | $u + 1 = 1 + u$	
12	$\forall y\ 1 + (y + 1) = (1 + y) + 1$	$\forall$E, 7
13	$1 + (u + 1) = (1 + u) + 1$	$\forall$E, 7, 12
14	$1 + u = u + 1$	$=$ simetria, 11
15	$1 + (u + 1) = (u + 1) + 1$	$=$**E**, 14, 13
16	$(u + 1) + 1 = 1 + (u + 1)$	$=$ simetria, 15
17	$u + 1 = 1 + u \to (u + 1) + 1 = 1 + (u + 1)$	$\to$I, 11–16
18	$\forall x(x + 1 = 1 + x \to (x + 1) + 1 = 1 + (x + 1))$	$\forall$I, 11–17
19	$0 + 1 = 1 \ \wedge\ \forall x(x + 1 = 1 + x \to (x + 1) + 1 = 1 + (x + 1))$	$\wedge$I, 10, 18
20	$\forall x(x + 1 = 1 + x)$	$\to$E, 8, 19

2.6.4 Teorias da lógica de 1ª ordem

Apresentamos em seguida uma formalização das noções de axiomatização e teoria.

Definição 2.15. *Uma teoria $\mathcal{T}$ é um conjunto de proposições de uma linguagem $\mathcal{L}$. Uma teoria duma linguagem $\mathcal{L}$ diz-se (formalmente)* completa *se para qualquer proposição φ de $\mathcal{L}$ ou φ ou $\neg\varphi$ é deduzível de $\mathcal{T}$.*

Definição 2.16. *(**Teoria duma estrutura**) Dada uma estrutura $\mathcal{A}$ duma linguagem $\mathcal{L}$, o conjunto de todas as proposições válidas em $\mathcal{A}$ denomina-se teoria da estrutura e denota-se por $Th(\mathcal{A})$.*

Definição 2.17. *(**Axiomatização**) Uma* axiomatização *de uma estrutura $\mathcal{A}$ é um conjunto de proposições Σ válido em $\mathcal{A}$, i.e, tal que $\mathcal{A} \models \Sigma$. Uma axiomatização de $\mathcal{A}$ é* completa *se para todo $\varphi \in Th(\mathcal{A})$, $\Sigma \models \varphi$ (e também $\Sigma \vdash \varphi$).*

Para a aritmética, p.e, pretendia-se uma axiomatização completa, isto é, em que fosse possível deduzir todas e só as proposições que eram verdadeiras em $\mathbb{N}$, isto é, na estrutura $\mathcal{N} = (\mathbb{N}, .^{\mathcal{A}})$. Já vimos que a axiomatização de Peano é íntegra.

Leituras suplementares [BE00] (Cap. 15,16.4)

2.7 Outros sistemas dedutivos

2.7.1 Sistemas dedutivos de Hilbert, H

Supondo apenas o conjunto completo de conectivas $\{\neg, \rightarrow\}$ e uma linguagem com igualdade:

Axiomas

- $\varphi \rightarrow (\psi \rightarrow \varphi)$
- $(\varphi \rightarrow (\psi \rightarrow \theta)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \theta))$
- $(\neg\psi \rightarrow \neg\varphi) \rightarrow ((\neg\psi \rightarrow \varphi) \rightarrow \psi)$
- $\forall x\varphi \rightarrow \varphi[t/x]$, onde x é substituível por t em φ
- $\forall x(\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \forall x\psi)$, onde x não ocorre livre em φ
- $x = x$
- $x = y \rightarrow (\varphi \rightarrow \varphi[y/x])$ e φ é atómica

Regras de inferência

- *Modus ponens*: de φ e de $\varphi \rightarrow \psi$, inferir ψ

- *generalização*: para $x \in Var$, inferir $\forall x\varphi$ a partir de φ

Proposição 2.13. *$\Sigma \vdash_{ND} \varphi$ se e só se $\Sigma \vdash_{H} \varphi$*

Demonstração. ($\Leftarrow$): Basta ver que os axiomas de H são teoremas de DN. A regra de inferência *modus ponens* corresponde à regra da eliminação de implicação de DN ($\rightarrow$ **E**) e a regra generalização à regra da introdução do quantificador universal de DN ($\forall$**I**)
($\Rightarrow$): é possível transformar uma dedução em DN, numa dedução em H. (Não o faremos neste curso...) □

2.7.2 Tableaux

Recordemos que as regras proposicionais de expansão de *tableaux* são:

$$\frac{\neg\neg\varphi}{\varphi} \qquad \frac{\neg \mathbf{F}}{\mathbf{V}} \qquad \frac{\neg \mathbf{V}}{\mathbf{F}} \qquad \frac{\alpha}{\alpha_1 \\ \alpha_2} \qquad \frac{\beta}{\beta_1 \mid \beta_2}$$

onde

α	α_1	α_2	β	β_1	β_2
$\varphi \wedge \psi$	φ	ψ	$\neg(\varphi \wedge \psi)$	$\neg\varphi$	$\neg\psi$
$\neg(\varphi \vee \psi)$	$\neg\psi$	$\neg\varphi$	$\varphi \vee \psi$	φ	ψ
$\neg(\varphi \rightarrow \psi)$	φ	$\neg\psi$	$\varphi \rightarrow \psi$	$\neg\varphi$	ψ

2.7.2.1 Regras de expansão para *tableaux*

A notação uniforme para as fórmulas pode ser extendida para os quantificadores (γ **universais** e δ **existenciais**):

γ	$\gamma(t)$	δ	$\delta(t)$
$\forall x\varphi$	$\varphi[x/t]$	$\exists x\varphi$	$\varphi[x/t]$
$\neg\exists x\varphi$	$\neg\varphi[x/t]$	$\neg\forall x\varphi$	$\neg\varphi[x/t]$

Nesta notação, as novas regras de expansão dos **tableaux** são:

$$\frac{\gamma}{\gamma(t)} \qquad \frac{\delta}{\delta(p)}$$

onde t é um termo fechado e p é uma constante nova.

Exemplo 2.16. *Considerar o* tableaux *para a negação duma fórmula válida:*

$$\neg(\forall x(P(x) \to Q(x)) \to (\forall x P(x) \to \forall x Q(x)))$$
$$\forall x(P(x) \to Q(x))$$
$$\forall x P(x)$$
$$\neg\forall x Q(x)$$
$$\neg Q(a)$$
$$P(a)$$
$$P(a) \to Q(a)$$

$$\neg P(a) \qquad\qquad Q(a)$$
$$\times \qquad\qquad \times$$

obtemos um tableaux *fechado, onde a é uma constante, que não aparece na fórmula.*

Para cada quantificador existencial é necessário usar uma constante nova. Por exemplo se considerarmos a negação de:

$$\forall x(P(x) \ \vee \ Q(x)) \to (\forall x P(x) \ \vee \ \forall x Q(x))$$

que é satisfazível mas não válida, se não tivermos esse cuidado obtemos um **tableaux** fechado.

Exemplo 2.17. *Considerar o* tableaux *para a negação da fórmula:*

$$\neg(\forall x(P(x) \ \vee \ Q(x)) \to (\forall x P(x) \ \vee \ \forall x Q(x))$$
$$(\forall x(P(x) \ \vee \ Q(x))$$
$$\neg(\forall x P(x) \ \vee \ \forall x Q(x))$$
$$\neg\forall x P(x)$$
$$\neg\forall x Q(x))$$
$$\neg Q(a)$$
$$\neg P(\mathbf{b})$$
$$P(a) \ \vee \ Q(a)$$
$$P(b) \ \vee \ Q(b)$$

$$P(b) \qquad\qquad Q(b)$$
$$Q(a) \qquad P(a) \qquad\qquad P(a) \qquad Q(a)$$
$$\times \qquad \times \qquad\qquad \times$$

Notar que um ramo não fechado de um tableaux *define uma estrutura em que a fórmula é válida.*

Exemplo 2.18. *Consideremos ainda um* ***tableaux*** *para φ igual a $\varphi_1 \ \wedge \ \varphi_2 \ \wedge \ \varphi_3$ onde:*

$$\begin{aligned} \varphi_1 &= \forall x \exists y P(x,y) \\ \varphi_2 &= \forall x \neg P(x,x) \\ \varphi_3 &= \forall x \forall y \forall z((P(x,y) \ \wedge \ P(y,z)) \to P(x,z)) \end{aligned}$$

Depois de aplicar as regras α obtemos

$$\forall x \exists y P(x,y))$$
$$\forall x \neg P(x,x))$$
$$\forall x \forall y \forall z((P(x,y) \ \wedge \ P(y,z)) \rightarrow P(x,z))$$

Agora não temos nenhuma constante (ou outro termo) para instanciar. Podemos escolher qualquer elemento a_1 *e obter* $\exists y P(a_1, y)$ *a partir de* φ_1*. E depois instanciar o* $\exists$ *com uma constante nova,* a_2*. E voltar a instanciar* φ_1 *com* a_2*, ficando* $\exists y P(a_2, y)$*. E voltar a instanciar este com uma nova constante* a_3*....e deste modo temos um processo que não termina...*
Pode-se mostrar que φ *não tem nenhum modelo* ***finito****!: Suponhamos que existe* $\mathcal{A}$ *com donínio finito mas não vazio. Por* φ_1 *existe uma sequência de* a_i *tal que* $\mathcal{A} \models_{s[x/a_i][y/a_j]} P(x,y)$*, para todos* i *e* $j = i+1$*. Por* φ_3*,* $j \neq i$*. Mas como o domínio é finito, existe* k *com* $a_k = a_i$*. O que contradiz* φ_2 *que obrigaria a* $\mathcal{A} \not\models_{s[x/a_i/]} [P(x,x)$*.*

Podemos concluir que os **tableaux** não são um processo de decisão para a validade das fórmulas de primeira ordem.

Capítulo 3

Indecidibilidade e Incompletude

3.1 Programa de David Hilbert

No início do século XX, o mátemático David Hilbert colocou diversas questões sobre os fundamentos da matemática:

1. A matemática é completa no sentido que cada afirmação pode ser demonstrada ou contraditada?
2. A matemática é consistente no sentido em que para nenhuma afirmação pode ser demonstrado que ela é verdadeira e falsa
3. A matemática é decidível no sentido que existe um método preciso que determine a verdade ou falsidade de qualquer afirmação matemática?

Respostas negativas:

Godël 1931: incompletude da aritmética

Church+Turing, 1936-7: Indecidibilidade da LPO

Para a aritmética, p.e, pretendia-se uma axiomatização (teoria) completa, isto é, em que fosse possível demonstrar (deduzir) todas e só as proposições que eram verdadeiras em $\mathbb{N}$.

Teorema 3.1. *(Teorema da incompletude de Gödel para PA) Os axiomas de Peano não constituem uma teoria completa.*

Teorema 3.2. *(Teorema da incompletude de Gödel) Não existe nenhum conjunto recursivamente enumerável de axiomas Σ, tal que para toda a fórmula φ se tem que $\Sigma \vdash \varphi$ se e só se $\mathcal{N} \models \varphi$.*

David Hilbert formulou com problema fundamental da matemática (*Entscheidungsproblem*) o seguinte:

> A matemática é decidível no sentido que existe um método preciso que determine a verdade ou falsidade de qualquer afirmação matemática?

Usando a lógica de primeira ordem para representar as teorias matemáticas, vem:
O problema fundamental da lógica matemática é:

> Dada uma fórmula φ duma linguagem de primeira ordem $\mathcal{L}$ existe um processo de computação que determine (decida) se φ é válida (ou equivalentemente se φ é um teorema dum sistema dedutivo)?

Resposta Não

Alan Turing *On computable numbers, with an application to the Entscheidungsproblem*, 1937

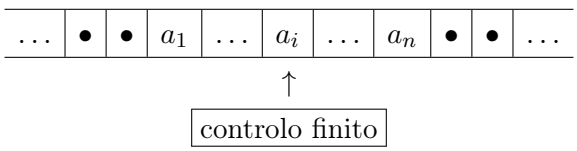
Alonzo Church *An unsolvable problem of elementary number theory*,1936

3.2 Indecidibilidade da Lógica de 1ª ordem

3.2.1 Revisões de Decidibilidade e Máquinas de Turing

Pela tese de Church-Turing, um problema é *efectivamente computável*, i.e. tem um método algorítmico para o resolver se e só se existe uma máquina de Turing que o resolve.

Máquinas de Turing Foi desenhada por Alan Turing para descrever o mínimos indispensável para se obter um método efectivo de computação.
Uma **máquina de Turing** (MT)



é constituída por

- um controlo finito (conjunto finito de estados)
- uma fita infinita dividida em células
- uma cabeça de leitura/escrita que actua de cada vez sobre uma célula da fita

No início os dados estão escritos na fita, um símbolo em cada célula, e as restantes células da fita contém um caracter especial da fita, designado por "branco", neste caso, $\bullet$. A cabeça está no símbolo mais à esquerda dos dados.
Num passo de computação (movimento), dependendo do símbolo lido na fita pela cabeça e do estado do controlo finito, a máquina

1. muda de estado
2. escreve um símbolo na célula que está debaixo da cabeça
3. move a cabeça para a esquerda ou para a direita

Formalmente uma máquina de Turing é

$$M = (S, A, \Gamma, \delta, s_0, \bullet, F)$$

onde

- S é um conjunto finito de estados
- Γ é o conjunto finito de *símbolos da fita*
- A é um subconjunto de Γ que não incluí $\bullet$, é o conjunto dos *símbolos de entrada*
- δ é a *função de transição*, função parcial de $S \times \Gamma$ em $S \times \Gamma \times \{\leftarrow, \rightarrow\}$
- s_0 é o estado inicial
- $\bullet$ é um símbolo de Γ, designado por *branco*
- $F \subseteq S$ é o conjunto de estados finais

Exemplo 3.1. $M = (\{s_0, s_1, s_2, s_3\}, \{0, 1\}, \{0, 1, \bullet, X\}, \bullet, s_0, \{s_4\})$
$\delta(s_0, 0) = (s_2, X, \rightarrow) \quad \delta(s_0, 1) = (s_1, X, \rightarrow) \quad \delta(s_0, X) = (s_0, X, \rightarrow)$
$\delta(s_0, \bullet) = (s_4, \bullet, \leftarrow) \quad \delta(s_2, 0) = (s_2, 0, \rightarrow) \quad \delta(s_2, 1) = (s_3, X, \leftarrow)$
$\delta(s_2, X) = (s_2, X, \rightarrow) \quad \delta(s_1, 1) = (s_1, 1, \rightarrow) \quad \delta(s_1, 0) = (s_3, X, \leftarrow)$
$\delta(s_1, X) = (s_1, X, \rightarrow) \quad \delta(s_3, 0) = (s_3, 0, \leftarrow) \quad \delta(s_3, 1) = (s_3, 1, \leftarrow)$
$\delta(s_3, X) = (s_3, X, \leftarrow) \quad \delta(s_3, \bullet) = (s_0, \bullet, \rightarrow).$
Que linguagem reconhece M?
$L = \{x \in \{0, 1\}^\star \mid x$ *tem igual número de* 1*'s e de* 0*'s*$\}$

Uma **configuração duma MT** representa o estado da fita, do controlo finito e da cabeça em cada instante. Embora a fita seja infinita, ao fim dum número finito de passos a cabeça só visitou um número finito de células.
Podemos representar a configuração ou a descrição instantânea (ID) por
$X_1 \cdots X_{i-1} s X_i \cdots X_n$
onde:

1. s é o estado da MT
2. a cabeça da fita está a reconhecer o i-ésimo símbolo a partir da esquerda
3. $X_1 \dots X_n$ é a sequência de caracteres da fita desde ou o caracter não-branco mais à esquerda ou o símbolo que está sobre a cabeça, conforme o que for mais à esquerda, e analogamente para o seu extremo direito.

A relação $\longrightarrow$, mudança de configuração num passo é definida da seguinte forma. Seja $X_1 \cdots X_{i-1} s X_i \cdots X_n$ uma configuração. Seja $\delta(s, X_i) = (s', Y, D)$. Suponhamos que $D = \leftarrow$. Então

$$X_1 \cdots X_{i-1} s X_i \cdots X_n \longrightarrow X_1 \cdots X_{i-2} s' X_{i-1} Y X_{i+1} \cdots X_n$$

Excepto se:

1. Se $i = 1$ então $s X_1 \cdots X_n \longrightarrow s' \bullet Y X_2 \cdots X_n$
2. Se $i = n$ e $Y = \bullet$ então $X_1 \cdots X_{n-1} s X_n \longrightarrow X_1 \cdots X_{n-2} s' X_{n-1}$

Analogamente, se $D = \rightarrow$ tem-se que

$$X_1 \cdots X_{i-1} s X_i \cdots X_n \longrightarrow X_1 \cdots X_{i-1} Y s' X_{i+1} \cdots X_n$$

Excepto se:

1. Se $i = n$ então $X_1 \cdots X_{n-1} s X_n \longrightarrow X_1 \cdots X_{n-1} Y s' \bullet$
2. Se $i = 1$ e $Y = \bullet$ então $s X_1 \cdots X_n \longrightarrow s' X_2 \cdots X_{n-1}$

Sejam as relações movimento em n passos $\stackrel{n}{\longrightarrow}$ e o fecho transitivo e reflexivo $\longrightarrow^\star$.
Uma palavra $x \in A^\star$ é **aceite** por uma máquina de Turing M se com palavra x na fita e no estado inicial s_0, M usando a função de transição δ, entra num **estado final**. Nota: mal a máquina atinja um estado final a computação pára, independentemente de ter lido ou não todos os símbolos de entrada.
A **linguagem aceite por** M, é

$$L(M) = \{x \mid \ x \in A^\star \text{ e } s_0 x \longrightarrow^\star x_1 s x_2 \text{ para algum } s \in F, \text{ e } x_1, x_2 \in \Gamma^\star\}$$

Se $x \in L(M)$ então M pára quando atinge um estado final. Caso contrário, M pode ou

- não parar
- parar num estado não final.

Linguagens recursivamente enumeráveis, recursivas e não recursivas Uma linguagem L diz-se **recursivamente enumerável** (r.e) ou **semi-decidível** se é aceite por uma máquina de Turing . Isto é existe uma MT M tal que $L = L(M)$.

Uma linguagem é **recursiva** se existe uma máquina de Turing que a aceita e que pára para todos os dados. Neste casos diz-se que as máquinas **reconhecem** as lingugem

Uma linguagem é **indecidível ou não recursiva** se não existe nenhuma máquina de Turing que a reconheça.

Pode ser:

- recursivamente enumeráveis (r.e): existe uma MT que pára se os dados pertencerem à linguagem, mas pode não parar caso contrário.

- não serem r.e

Podemos dar uma outra caracterização dos conjuntos recursivamente enumeráveis, e que justifica o seu nome...

Teorema 3.3. *Uma linguagem $L \subseteq A^\star$ é r.e. se e só se existe uma máquina de Turing M que enumera os seus elementos, isto é, $L = E(M)$.*

Demonstração. ($\Leftarrow$) Suponhamos que $L = E(M)$, para alguma MT M. Então, L é aceite por uma máquina de Turing M' que com dados x simula M com a fita inicialmente vazia: se durante a computação de M, x é enumerado, então M' pára e aceita x. Caso contrário M' não termina.

($\Rightarrow$) Suponhamos que L é r.e.. Então, existe uma máquina de Turing M que aceita L. Pretende-se obter uma máquina M' tal que $L = E(M')$, isto é, que enumere os elementos de L. M' não pode simular directamente M para cada um dos $x \in A^\star$, porque se $x \notin L$, M não pára. A técnica que será usada denomina-se *dovetailing*[1] – computações intercaladas. M' com a fita inicialmente vazia opera da seguinte forma: os elementos de $A^\star$, $x_0, x_1, \ldots$ são gerados por ordem lexicográfica e para cada x_n, M' simula M com dados x_n, $M(x_n)$, intercaladamente e para $n = 0, 1, 2, \ldots$ passos de computação: simula o $1^{\underline{o}}$ passo de computação de M para x_0; simula o $1^{\underline{o}}$ passo de computação de M para x_1; simula $2^{\underline{o}}$ passo de computação para x_0; simula o $1^{\underline{o}}$ passo de computação de M para x_2; ... Pode-se obter a seguinte tabela em que os números da tabela são a ordem das computações de M':

[1] A tradução literal de "dove tail" é "cauda de andorinha"

	1	2	3	4	... (computações em M)
$M(x_0)$	1	3	6	10	...
$M(x_1)$	2	5	9	...	
$M(x_2)$	4	8	...		
$M(x_3)$	7	...			
$\vdots$					

Se M pára para algum destes dados, por exemplo x_n, então M' escreve x_n na fita (seguido de um $\bullet$) e continua a simulação. Isto garante que M' só enumera elementos em L. Se $x_n \in L$, então M com dados x_n pára ao fim de um número finito de computações, logo x_n será enumerado por M'. □

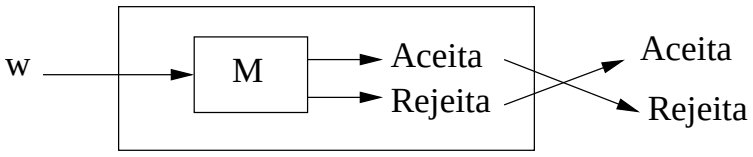
Linguagens e Complementos O estudo das linguagens complementares pode permitir distinguir se uma linguagem é r.e mas não recursiva!

Teorema 3.4. *Se L é recursiva, $\overline{L}$ é recursiva*

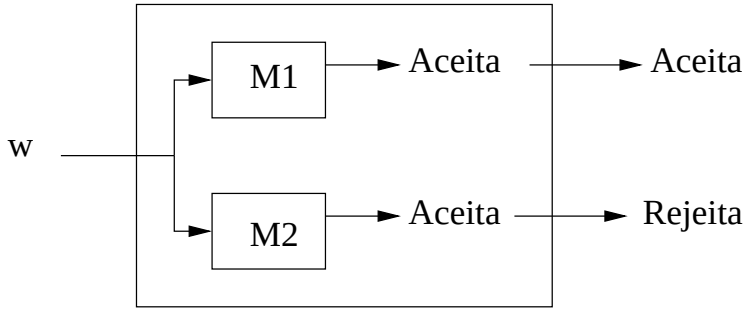
Demonstração. Suponhamos que $L = L(M)$ para uma MT M que pára sempre. Basta construir uma MT $\overline{M}$ igual a M excepto em que

- os estados finais de M passam a não o ser em $\overline{M}$
- tem um novo estado de final r, sem transições dele.
- para cada par (estado não final de M,símbolo da fita) sem transições em M, acrescenta-se uma transição para o estado r.

□



Teorema 3.5. *Se L e $\overline{L}$ são r.e., então L é recursiva (e portanto, também o é $\overline{L}$)*



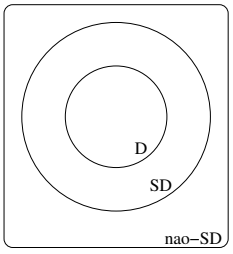
Demonstração. Seja $L = L(M_1)$ e $\overline{L} = L(M_2)$. Construímos uma MT M que simula em paralelo M_1 e M_2, usando duas fitas e estados cujas componentes são os de M_1 e M_2. Se com dados w, M_1 aceitar, M aceita. Senão $w \notin L$, mas $w \in \overline{L}$. Então, M_2 tem de aceitar w. Nessa altura M pára e rejeita. Portanto, com todos os dados M pára e $L(M) = L$. Logo L é recursiva. □

As linguagens podem dividir-se em:

D recursivas

SD r.e

não-SD não r.e



Há apenas 4 maneiras de uma linguagem L e $\overline{L}$ se situarem no diagrama:

- Ambas L e $\overline{L}$ são recursivas:estão em **D**
- Nem L nem $\overline{L}$ são r.e: estão e, **não-SD**
- L é r.e mas não recursiva e $\overline{L}$ não r.e: L em **SD\D** e $\overline{L}$ em **não-SD**
- $\overline{L}$ é r.e mas não recursiva e L não r.e: $\overline{L}$ em **SD\D** e L em **não-SD**

Uma linguagem não recursiva Considere-se a linguagem constituída pelos pares (M, x) tal que:

1. M é (a codificação em binário de) uma máquina de Turing cujo alfabeto de entrada é $\{0, 1\}$
2. $x \in \{0, 1\}^\star$
3. M aceita x

As máquinas U que aceitam esta linguagem denominam-se Máquinas de Universais de Turing, i.e,

$$L_u = L(U) = \{(M, x) \mid x \in L(M)\}$$

Com dados (M, x), U simula M com dados x.
L_u é recursivamente enumerável mas não é recursiva.

Redução entre linguagens Dadas $L_1 \subseteq \Sigma^\star$ e $L_2 \subseteq \Delta^\star$, uma redução de L_1 a L_2 é uma função total computável (algoritmo) $\sigma : \Sigma^\star \longrightarrow \Delta^\star$ tal que para todo $x \in \Sigma^\star$:

$$x \in L_1 \;\Leftrightarrow\; \sigma(x) \in L_2$$

Para σ existe uma máquina de Turing total (que pára sempre) e que com dados x pára com $\sigma(x)$ na fita.
Diz-se que L_1 é redutível a L_2, $L_1 \leq_m L_2$

Teorema 3.6.

a) Se $L_1 \leq_m L_2$ e L_2 é r.e então L_1 é r.e. Equivalentemente, se $L_1 \leq_m L_2$ e L_1 não é r.e então L_2 também não.

b) Se $L_1 \leq_m L_2$ e L_2 é recursiva então L_1 é recursiva. Equivalentemente, se $L_1 \leq_m L_2$ e L_1 não é recursiva então L_2 também não.

3.2.2 Linguagem L_∞

Seja L_∞ uma linguagem de 1ª ordem cujo conjunto de símbolos não-lógicos é infinito numerável: $\mathcal{F}_0 = \{c_0, c_1, \ldots\}$, e para cada $n > 0$, $\mathcal{F}_n = \{f_0^n, f_1^n, \ldots\}$ e $\mathcal{R}_n = \{R_0^n, R_1^n, \ldots\}$.
Para representar esta linguagem como uma linguagem formal temos que considerar um alfabeto finito.
Podemos escrever os termos, as fórmulas e as deduções de $\mathcal{L}_\infty$ como **palavras** de um **alfabeto** apropriado. Por exemplo:

$$A_\infty = \{x, c, f, R, \underline{0}, \underline{1}, \ldots, \underline{9}, \overline{0}, \overline{1}, \ldots, \overline{9}, =, \wedge, \vee, \rightarrow, \neg, \forall, \exists, (,), \mathbf{F}\}$$

onde os inteiros $\underline{n}$ serão usados para indentificar o símbolo funcional (ou relacional) e os inteiros $\overline{n}$ a aridade do símbolo. Assim,
O termo $f_{33}^2(x_{41}, f_5^1(c_{28}))$ pode ser representado por

$$f\overline{2}\underline{33}x\underline{41}f\overline{1}\underline{5}c\underline{28}$$

A fórmula $\forall x_{13}\exists x_{23}(R_{14}^1(x_{13}) \rightarrow R_{12}^2(x_{13}, x_{23}))$ pode ser representada por

$$\forall x\underline{13}\exists x\underline{23}(R\overline{1}\underline{14}x\underline{13} \rightarrow R\overline{2}\underline{12}x\underline{13}x\underline{23})$$

Uma dedução no sistema de dedução natural é uma sequência finita de fórmulas, onde cada fórmula é precedida de uma sequência de inteiros que indica qual o nível e qual o número de ordem da sub-dedução a que pertence (11 é a dedução de topo).
Nota que uma dedução pode então ser vista como uma palavra de A_∞.

Uma dedução num sistema dedutivo de Hilbert é a uma sequência finita de fórmulas $\alpha_1, \cdots, \alpha_k$, logo também uma palavra de $A_\infty^\star$.
Para uma sequência $\alpha_1, \ldots, \alpha_k$ ser uma dedução de α_k é necessário ainda que cada α_i seja um axioma ou um dos α_j, $j < i$, ou o resultado da aplicação de uma regra.

Proposição 3.1. *O conjunto dos termos de L_∞ é decidível.*

Demonstração. Pela tese de Church-Turing basta encontrar um algoritmo que verifique se $\alpha \in A_\infty^\star$ corresponde a um termo:

- se $\alpha = c\beta$ ou $\alpha = x\beta$, β tem de ser uma sequência não nula de $\underline{n}$s, com $n = 0, 1, 2, \ldots$
- se $\alpha = f\beta$, β é da forma $\overline{n_k} \ldots \overline{n_0}\underline{m_l} \ldots \underline{m_0}\gamma$, com $n_0, \ldots, n_k, m_0, \ldots, m_l \in \{0, \ldots, 9\}$, $n_k \neq 0$ se $k \neq 0$, $m_l \neq 0$ se $l \neq 0$ e γ começa com c, x ou f e pode-se decompor em termos $\gamma_1, \ldots, \gamma_p$, com $p = \sum_{i=0}^{k} n_i \times 10^i$. O processo termina pois $|\gamma| < |\alpha|$.

□

Proposição 3.2. *O conjunto das fórmulas atómicas de L_∞ é decidível.*

Proposição 3.3. *O conjunto das fórmulas de L_∞ é decidível.*

Exercício 3.1. *Demonstra as proposições 3.2 e 3.3.* ◇

Proposição 3.4. *O conjunto das proposições de L_∞ é decidível.*

Demonstração. Uma proposição é uma fórmula sem variáveis livres. Convém começar por determinar o conjunto das variáveis livres de uma fórmula φ, $VL(\varphi)$. Isso pode ser feito por indução na estrutura da fórmula:

$$\begin{aligned}
VL(t_1 = t_2) &= Var(t_1) \cup Var(t_2) \\
VL(R_j^n(t_1, \ldots, t_n)) &= \cup_{i=1}^{n} Var(t_i) \\
VL(\neg\varphi) &= VL(\varphi) \\
VL(\varphi \circ \psi) &= VL(\varphi) \cup VL(\psi) \quad \circ \in \wedge, \vee, \rightarrow \\
VL(\forall x\varphi) &= VL(\varphi) \setminus \{x\} \\
VL(\exists x\varphi) &= VL(\varphi) \setminus \{x\}
\end{aligned}$$

e $Var(t)$ é o conjunto de variáveis que ocorrem em t (determina!).
Bastará então determinar se $VL(\varphi) = \emptyset$. □

Começamos por ver que o conjunto das proposições válidas é recursivamente enumerável.

Proposição 3.5. *O conjunto das proposições de L_∞ que são teoremas é semi-decidível (ou recursivamente enumerável (r.e)).*

Demonstração. Qualquer que seja o sistema dedutivo é possível encontrar um algoritmo que verifica se uma palavra de $\alpha \in A_\infty^\star$ é da forma $\alpha_1 \dots \alpha_k$ e tal que $\alpha_1 \dots \alpha_k$ é uma dedução de α_k. Também é possível decidir se α_k é uma proposição. Então é possível construir uma MT M que dada a representação α de uma fórmula φ:

1. determina se α é uma proposição
2. gera as palavras de $A_\infty^\star$, lexicograficamente
3. para cada uma verifica se é uma dedução $\alpha_1, \dots, \alpha_k$ e se α_k é α. Se for, M aceita α. Senão continua no passo 2.

□

Nota que não é garantido que a máquina de Turing pare sempre: só é garantido que pare se α for uma representação dum teorema.

Corolário 3.1. *O conjunto das proposições válidas de L_∞ é semi-decidível (ou recursivamente enumerável (r.e)).*

Demonstração. Pelo teorema da completude basta mostrar que o conjunto de teoremas de L_∞ é recursivamente enumerável, o que foi feito na proposição anterior. □

Concluímos assim que o problema de determinar se uma proposição da lógica de primeira ordem é válida é pelo menos semi-decidível.
Seja L_{val} a linguagem das proposições válidas de L_∞. Por redução de L_u, vamos ver que L_{val} não é recursiva!
Seja $L_u = \{(M, x) \mid x \in L(M)\}$
Dado M e x vamos construir (com um algoritmo σ) uma fórmula φ tal que:

$$(M, x) \in L_u \text{ sse } \sigma((M, x)) = \varphi \in L_{val}$$

Isto é, φ é válida se e só se M aceita x. Pela completude, φ é um teorema se e só se M aceita x.
E se houvesse um algoritmo para determinar se $\varphi \in L_{val}$ então haveria um algoritmo para determinar se $(M, x) \in L_u$. Absurdo!
Vamos então especificar a redução $L_u \leq_m L_{val}$.
Suponhamos que uma máquina de Turing M é dada por:

$$(\{s_1, s_2, \dots, s_k\}, \{0, 1\}, \{X_1, X_2, \dots, X_m\}, \delta, s_1, \bullet, \{s_2\})$$

Vamos considerar alguns símbolos não lógicos de L_∞, a que para legibilidade iremos dar nomes sugestivos. Seja c_ϵ uma constante, f_{X_i} um símbolo funcional unário para cada símbolo da fita X_i de M e R_{s_i} um símbolo de predicado binário para cada estado s_i de M. As variáveis de L_∞ serão $v_1, v_2, \ldots$.
Sendo $x = a_1 a_2 \ldots a_n$ com $a_i \in \{0, 1\}$, podemos representá-lo pelo termo:

$$f_{a_1}(f_{a_2} \ldots (f_{a_{n-1}}(f_{a_n}(c_\epsilon))) \ldots)$$

Para simplificar iremos denotar por $\underline{x}$ este termo e analogamente os restantes termos que denotam palavras de $\{X_1, X_2, \ldots, X_m\}^\star$.
Seja a estrutura $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ de L_∞ tal que $A = \{X_1, X_2, \ldots, X_m\}^\star$, $c_\epsilon^{\mathcal{A}} = \epsilon$, $f_{X_i}^{\mathcal{A}}(w) = X_i w$ para $1 \leq i \leq m$ e $w \in A$ e para cada estado s_i,

$$R_{s_i}^{\mathcal{A}} \quad = \quad \{(w, y) \in A \times A \mid s_1 x \xrightarrow{M}{}^{\star} w s_i y\}$$

isto é, estão em $R_{s_i}^{\mathcal{A}}$ os pares (w, y) tal que existe uma configuração $w s_i y$ atingível da inicial.
Em $\mathcal{A}$ a proposição

$$R_{s_1}(c_\epsilon, \underline{x})$$

é verdadeira , dado que a configuração inicial (de M com dados x) é atingível. E

$$\exists v_1 \exists v_2 R_{s_2}(v_1, v_2)$$

é verdadeira em $\mathcal{A}$ se e só se M aceita x.
Para $\delta(X_i, s_k) = (X_j, s_l, D_1)$, a proposição seguinte é verdadeira em $\mathcal{A}$:

$$\forall v_1 \forall v_2 (R_{s_k}(v_1, X_i v_2) \rightarrow R_{s_l}(v_1 X_j, v_2))$$

Analogamente se $\delta(X_i, s_k) = (X_j, s_l, D_2)$, a proposição seguinte é verdadeira em $\mathcal{A}$:

$$\forall v_1 \forall v_2 (R_{s_k}(v_1 X_m, X_i v_2) \rightarrow R_{s_l}(v_1, X_m X_j v_2))$$

Excepto nos casos especiais em que a cabeça de M está sob símbolos extremos da configuração: nesses casos, para D_1 (direita)

$$\forall v_1 (R_{s_k}(v_1, X_i) \rightarrow R_{s_l}(v_1 X_j, c_\epsilon))$$
$$\forall v_2 (R_{s_k}(c_\epsilon, X_i v_2) \rightarrow R_{s_l}(c_\epsilon, v_2)), \text{ se } X_j = \bullet$$

e para D_2 (esquerda)

$$\forall v_2 (R_{s_k}(c_\epsilon, X_i v_2) \rightarrow R_{s_l}(c_\epsilon, \bullet X_j v_2))$$
$$\forall v_1 (R_{s_k}(v_1 X_m, X_i) \rightarrow R_{s_l}(v_1, X_m)), \text{ se } X_j = \bullet$$

Seja T a conjunção de todas as fórmulas anteriores. Nota que são em número finito! E seja φ a fórmula

$$(R_{s_1}(c_\epsilon, \underline{x}) \ \wedge \ T) \rightarrow \exists v_1 \exists v_2 R_{s_2}(v_1, v_2)$$

O processo descrito é um algoritmo (seja σ) e $\sigma((M,x)) = \varphi$.
Falta ver que $(M,x) \in L_u$ sse $\sigma((M,x)) = \varphi \in L_{val}$.
($\Rightarrow$) Se M aceita x então existe uma sequência de configurações (computação válida) que conduz da configuração inicial a um estado final. Cada passo corresponde a uma instância duma das fórmulas da conjunção T. Podemos então construir uma dedução para φ ($\vdash \varphi$):

- Supor $R_{s_1}(\epsilon, \underline{x}) \ \wedge \ T$
- Aplicar $\wedge$ **E** tantas vezes quantos os elementos de T
- aplicar $\forall$**E** para obter a instância adequada para simular cada um dos passos a partir de $R_{s_1}(\epsilon, \underline{x})$ e $\rightarrow$ **E**, para obter a *configuração* seguinte
- Quando se deduzir a fórmula correspondente a ter-se atingido o estado final, podemos deduzir $\exists w \exists y R_{s_2}(w,y)$ aplicando duas vezes a regra $\exists$**I**. E aplicando $\rightarrow$ **I** deduzimos φ.

Então φ é um teorema, logo $\varphi \in L_{val}$.
($\Leftarrow$) Se φ for válida em todas as estruturas, então é válida em $\mathcal{A}$. O que implica que M tem uma computação que leva ao estado final, começando com x na fita. Isto é $(M,x) \in L_u$.
Temos então demonstrado que:

Teorema 3.7. *Determinar se uma fórmula φ duma linguagem de $1^{\underline{a}}$ ordem é* ***válida*** *é um problema indecidível.*

Corolário 3.2. *Determinar se uma fórmula φ linguagem de $1^{\underline{a}}$ ordem é* ***satisfazível*** *é um problema indecidível.*

Exercício 3.2. *Demonstra o corolário anterior.* $\diamond$

Resolução 3.2
Uma fórmula φ é satisfazível se e só se $\neg\varphi$ é não válida. Mas o problema de determinar se uma fórmula é não válida, é o problema complementar de determinar se é válida, logo tem de ser indecidível (porquê?).

3.3 Subconjuntos decidíveis de L_∞

Se restringirmos o tipo de fórmulas podemos ter subconjuntos onde o problema de determinar se uma fórmula é válida é decidível. Por exemplo:

- A linguagem L_{Prop} correspondente à lógica proposicional
- A linguagem em que os símbolos de predicado são todos unários
- As fórmulas cuja forma prenexa é $\exists x_1 \ldots \exists x_n \forall y_1 \ldots \forall y_m \varphi$ e φ não tem termos com símbolos funcionais
- ...

3.4 Incompletude dos axiomas de Peano (PA)

Recordemos os axiomas da aritmética de Peano:
Seja uma linguagem de 1ª ordem com igualdade, $\mathcal{F}_0 = \{0, 1\}$ e $\mathcal{F}_2 = \{+, \times\}$
Os axiomas são factos básicos dos números naturais:

1. $\forall x(x + 1 \neq 0)$
2. $\forall x \forall y(x + 1 = y + 1 \rightarrow x = y)$
3. $0 + 1 = 1$
4. $\forall x\ x + 0 = x$
5. $\forall x \forall y\ x + (y + 1) = (x + y) + 1$
6. $\forall x\ x \times 0 = 0$
7. $\forall x \forall y\ x \times (y + 1) = (x \times y) + x$
8. (princípio da indução) $(\varphi[0/x] \ \wedge \ (\forall x(\varphi \rightarrow \varphi[x + 1/x]) \rightarrow \forall x \varphi$

Os axiomas de Peano são uma axiomatização de $\mathcal{N} = (\mathbb{N}, \cdot^{\mathcal{N}})$ (onde as operações aritméticas têm o valor habitual). Então se $PA \vdash \varphi$, $\mathcal{N} \models \varphi$, isto é, o sistema é íntegro. Mas, temos que

Teorema 3.8. *(Teorema da incompletude de Gödel para PA) Os axiomas de Peano não constituem uma teoria completa.*

Teorema 3.9. *(Teorema da incompletude de Gödel) Não existe nenhum conjunto recursivamente enumerável de axiomas Σ, tal que para toda a fórmula φ se tem que $\Sigma \vdash \varphi$ se e só se $\mathcal{N} \models \varphi$ (i.e Σ não é completo).*

A demonstração feita por Kurt Gödel é um pouco complexa: ele construiu uma fórmula G de $\mathcal{L}_N$ que afirmava que *esta proposição não é deduzível dos axiomas de Peano* (ou noutro conjunto r.e de axiomas).
A proposição G é verdadeira!
Se G fosse falsa então, G era deduzível em **PA**, mas então pela integridade seria verdadeira.

Para a construção de G é necessário codificar as fórmulas como números inteiros, assim como as deduções. E também é necessário poder exprimir que x *é uma dedução de* y:

$$Ded(x, y)$$

Se se representar por $\lceil\varphi\rceil$ o *código de* φ, temos:

$$\mathbf{PA} \vdash G \leftrightarrow \neg\exists x Ded(x, \lceil G\rceil)$$

Mas os pormenores, para esta construção são ainda bastante complicados. Vamos ver outra demonstração, proposta por Alan Turing.
Seja Σ, PA ou outro conjunto (r.e.) de axiomas para $\mathcal{N}$.

1. O conjunto de proposições $\{\varphi \mid \Sigma \vdash \varphi\}$ é recursivamente enumerável
2. O conjunto $Th(\mathcal{N})$ não é recursivamente enumerável.

então estes 2 conjuntos não podem ser iguais e portanto Σ, se for íntegro, não pode ser completo. Assim temos demonstrado o (Primeiro) Teorema da incompletude de Gödel.

Exercício 3.3. *Mostra que PA é um conjunto recursivamente enumerável.* $\diamond$

Exercício 3.4. *Mostra que $\{\varphi \mid PA \vdash \varphi\}$ é recursivamente enumerável.* $\diamond$

Proposição 3.6. *$Th(\mathcal{N})$ não é recursivamente enumerável.*

Demonstração. Por uma redução de $\overline{L_u} \leq_m Th(\mathcal{N})$, onde

$$L_u = \{(M, x) \mid x \in L(M)\}$$

Dado (M, x) constrói-se uma fórmula ψ de $\mathcal{L}_N$, tal que

$$(M, x) \in \overline{L_u} \Leftrightarrow \psi \in Th(\mathcal{N})$$

Isto é, ψ tem de exprimir que M *não aceita* x.
Usando fórmulas mais simples constrói-se uma fórmula $VALCOMP_{M,x}(y)$ tal que y representa uma *história de computação válida* de M com dados x.
Isto é, y representa uma sequência de configurações C_i tal que:

- C_0 é configuração inicial de M com dados x: s_0x
- C_N é uma configuração de aceitação
- C_{i+1} segue num passo de C_i, i.e., $C_i \longrightarrow C_{i+1}$, para $0 \leq 1 \leq N-1$

Se M não aceita x, não existe história de computação válida.
Então ψ é a fórmula $\neg\exists y VALCOMP_{M,x}(y)$.
Nota que $VALCOMP_{M,x}(y)$ tem de ser expressa só usando operações aritméticas e construída por um algoritmo (a partir de M e x).
Suponhamos que as configurações de M estão codificadas num alfabeto Δ de tamanho p, com p primo (em binário seria mais complicado).
Sendo $x = a_1 a_2 \ldots a_n$, a configuração inicial será codificada por $k_0 k_1 k_2 \ldots k_n$ dígitos p-ários:

$$\begin{array}{ccccccc} s0 & a_1 & a_2 & \ldots & a_n \\ k_0 & k_1 & k_2 & \ldots & k_n \end{array}$$

Suponhámos ainda que o símbolo $\bullet$ corresponde ao dígito k.
Duas configurações consecutivas C_i e C_{i+1} só podem diferir no máximo nos dígitos que correspondam à mudança de configuração (4 no máximo). Por exemplo, se:
$\delta(s_1, a) = (s_2, b, \rightarrow)$
então podemos ter, p.e

$a \quad s_1 \quad a \quad b \qquad a \quad b \quad s_2 \quad b$

Dado Δ ser finito só há um número finito destes uplos. Seja C o conjunto de todos os uplos (a, b, c, d, e, f, g, h) tal que (a, b, c, d) ocorrem consecutivamente numa configuração C_i, e se (e, f, g, h) ocorrem nas posições correspondentes em C_{i+1} então isso é consistente com as transições δ de M. Agora temos que codificar tudo isto em fórmulas de **PA**. Notar que a sequência de configurações (história) vai ter de ser representada por um número v (e não vai haver separadores entre elas). Se houver uma configuração de aceitação então pode-se calcular qual o "espaço gasto" no máximo por cada configuração. Existirá uma potência de p suficientemente grande (será designada por c) e que permita determinar quais os dígitos que correspodem a cada configuração.
Comecemos por recordar algumas fórmulas.

Divisão inteira: $x = q \times y + r \ \wedge \ r < y =_{def} INTDIV(x, y, q, r)$

y **divide** x**:** $\exists q INTDIV(x, y, q, 0) =_{def} DIV(x, y)$

$2 : \ 1 + 1 =_{def} 2$

x **é par:** $DIV(2, x) =_{def} Par(x)$

x **é ímpar:** $\neg Par(x) =_{def} Impar(x)$

x **é primo:**

$$\neg x < 2 \ \wedge \ \forall y(DIV(y, x) \rightarrow (y = 1 \ \vee \ y = x)) =_{def} Primo(x)$$

x é potência de 2:

$$\forall y((DIV(y,x) \ \wedge \ Primo(x)) \rightarrow y = 2) =_{def} P(2,x)$$

y é 2^k e o k-ésimo bit de x é 1:

$$P(2,y) \ \wedge \ \forall q \forall r(INTDIV(x,y,q,r) \rightarrow Impar(q)) =_{def} BIT(x,y)$$

Divide-se x por y em binário e o quociente tem de ter o último bit 1. Assim podem-se ver os números como palavras e extrair "bits".

Vamos definir mais algumas (omitindo o sinal $\times$):

o número y é uma potência de p

$$\forall z(DIV(z,y) \ \wedge \ Primo(z) \rightarrow z = p) =_{def} Pot_p(y)$$

O número d é uma potência de p e $d = |v|$, v como palavra de Δ $Pot_p(d) \ \wedge \ v \leq d$ $=_{def} Comp(v,d)$

Vamos supor y e z potências de p.

O dígito p-ário de v na posição y é b

$\exists u \exists a(v = a + by + upy \ \wedge \ a < y \ \wedge \ b < p) =_{def} DIGIT(v,y,b)$ (Dividindo por p o quociente de v dividido por y, temos b como resto)

Os 4 dígitos p-ários de v "na"posição y são b, c,d,e

$$\begin{aligned} \exists u \exists a(v = a + by + cpy + dppy + epppy + upppppy \\ \wedge a < y \ \wedge \ b < p \ \wedge \ c < p \ \wedge \ d < p \ \wedge \ e < p) \\ =_{def} 4DIGIT(v,y,b,c,d,e) \end{aligned}$$

Os 4 dígitos de v na posição y e os na posição z correspondem

$$\begin{aligned} \bigvee_{(a,b,c,d,e,f,g,h) \in C} (4DIGIT(v,y,a,b,c,d) \ \wedge \ 4DIGIT(v,z,e,f,g,h)) \\ =_{def} MATCH(v,y,z) \end{aligned}$$

A palavra v representa as configurações consecutivas de M de tamanho entre c e d

$$\begin{aligned} \forall y((Pot_p(y) \ \wedge \ yppc < d) \rightarrow MATCH(v,y,yc) \\ =_{def} MOVE(v,c,d)) \end{aligned}$$

Cada configuração tem no máximo tamanho c e duas configurações consecutivas estão de acordo com o δ de M. c e d são potências de p.

A palavra v começa com a configuração inicial de M com dados x, preenchida com brancos até ao comprimento c

$$\begin{aligned} &\bigwedge_{i=0}^{n} DIGIT(v, p^i, k_i) \ \wedge \ p^n < c \\ \wedge \forall y((Pot_p(y) \ \wedge \ p^n < y < c) &\rightarrow DIGIT(v, y, k)) \\ &=_{def} Inicio(v, c) \end{aligned}$$

c é uma potência de p;n e p^i, $0 \leq i \leq n$ representam constantes que dependem de x

A palavra v tem um estado final algures

$$\begin{aligned} \exists y(Pot_p(y) \ \wedge \ y < d \ \wedge \ &\bigvee_{w \in F} DIGIT(v, y, w) \\ &=_{def} Aceita(v, d) \end{aligned}$$

onde F é o conjunto dos dígitos p-ários que representam os símbolos de Δ que correspondem a estados finais.

A palavra v é uma história de computação válida de M com dados x

$$\begin{aligned} &\exists c \exists d(Pot_p(c) \ \wedge \ c < d \ \wedge \ Comp(v, d) \\ \wedge \ Inicio(v, c) \ \wedge \ &Move(v, c, d) \ \wedge \ Aceita(v, d)) \\ &=_{def} VALCOMP_{M,x}(v) \end{aligned}$$

A máquina M não aceita x

$$\neg \exists v VALCOMP_{M,x}(v)$$

□

Leituras suplementares [BE00] (Cap. 10.5,12.5,15,16.4,18.4) [Koz97] (Cap. 39 e 40) [Pap94] (Cap. 6)

Capítulo 4

Programação em Lógica

Pretende-se que um programa seja um conjunto de fórmulas lógicas e que a execução desse programa corresponda a uma demonstração de que uma fórmula é um teorema. Os sistemas que vamos ver baseiam-se em:

- considerar fórmulas em forma prenexa e em que a matriz está em normal conjuntiva
- como sistema dedutivo usar variantes da resolução
- o sistema dedutivo deve ser íntegro e completo, pelo menos para uma dada estrutura.
- ser computacionalemente "universal", i.e., equivalente a máquinas de Turing.

4.1 Cláusulas

Seja $\mathcal{L}$ uma linguagem de 1ª ordem com igualdade e pelo menos uma constante.

Definição 4.1. *Um* literal positivo *(ou átomo) é uma fórmula atómica.*
Um literal negativo *é a negação de uma fórmula atómica.*

Exemplo 4.1. *$P(x, y)$, $f(x) = a$ são literais positivos. E, $\neg P(x, y)$ é um literal negativo.*

Definição 4.2. *Uma* cláusula *é uma fórmula φ da forma:*

$$\forall x_1 \ldots \forall x_s(\alpha_1 \ \vee \ \ldots \ \vee \ \alpha_k \ \vee \ \neg\beta_1 \ \vee \ \ldots \ \vee \ \neg\beta_n)$$

onde α_i, β_i são átomos e $x_1, \ldots, x_s$ são todas as variáveis que ocorrem na fórmula.
Também se pode escrever φ como:

$$\forall x_1 \ldots \forall x_s((\beta_1 \wedge \ldots \wedge \beta_n) \rightarrow (\alpha_1 \ \vee \ \ldots \ \vee \ \alpha_k))$$

ou ainda iremos representá-la pela notação clausal *seguinte:*

$$\alpha_1, \ldots, \alpha_k \leftarrow \beta_1, \ldots, \beta_n$$

4.1.1 Conversão em forma clausal

Vamos ver que para qualquer fórmula φ da lógica de primeira ordem é possível encontrar uma conjunto de cláusulas que é satisfazível se e só se φ é satisfazível.
A proposição seguinte vai permitir *eliminar* quantificadores existênciais.

Proposição 4.1. *Seja $\forall y_1 \dots y_n \exists x \varphi$ uma proposição de uma linguagem de $1^{\underline{a}}$ ordem $\mathcal{L}$ e seja $\mathcal{L}'$ uma linguagem com os símbolos de $\mathcal{L}$ e com mais um símbolo n-ário f. Então, $\forall y_1 \dots y_n \exists x \varphi$ é satisfazível em $\mathcal{L}$ se e só se $\forall y_1 \dots y_n \varphi[f(y_1, \dots, y_n)/x]$ é satisfazível em $\mathcal{L}'$.*

Demonstração. Temos que mostrar que existe $\mathcal{A}$ (estrutura de $\mathcal{L}$) tal que $\mathcal{A} \models \forall y_1 \dots y_n \exists x \varphi$ sse existe $\mathcal{A}'$ (estrutura de $\mathcal{L}'$) tal que $\mathcal{A}' \models \forall y_1 \dots y_n \varphi[f(y_1, \dots, y_n)/x]$
($\Rightarrow$) Para todos os $a_1, \dots a_n \in A$, existe $b \in A$ tal que

$$\mathcal{A} \models_s [a_1/y_1] \dots [a_n/y_n][b/x]\varphi$$

Nota que b depende dos $a_1, \dots, a_n$.
Então a estrutura $\mathcal{A}'$ pode ser igual a $\mathcal{A}$ (mesmo domínio e interpretação dos símbolos comuns) e tal que o valor de $f^{\mathcal{A}'}$ seja dado pela seguinte função:

$$\forall a_1, \dots, a_n \in A, f^{\mathcal{A}'}(a_1, \dots, a_n) = b$$

sendo b cada um dos definidos anteriormente.
($\Leftarrow$)
Em $\mathcal{A}'$, consideremos os valores de $f^{\mathcal{A}'}(a_1, \dots, a_n)$. Estes são os valores de cada um dos b. □

Existe um algoritmo que converte uma proposição φ de uma linguagem de $1^{\underline{a}}$ ordem, numa conjunção (ou conjunto) de cláusulas $\Phi = \varphi_1 \wedge \dots \wedge \varphi_n$ (duma linguagem alargada $\mathcal{L}'$) tal que:

- cada φ_i é da forma $\forall x_1 \dots \forall x_n (\lambda_1 \ \vee \ \dots \ \vee \ \lambda_n)$, onde cada λ_i é um literal
- a fórmula φ é satisfazível se e só se Φ é satisfazível

Esse algoritmo denomina-se *Skolemização* e consiste em:

1. Converter φ em forma normal prenexa $Q_1 x_1 Q_2 x_2 \dots Q_n x_n \psi$ onde cada Q_i é ou $\forall$ ou $\exists$ e ψ não tem quantificadores (e denomina-se a *matriz*).:
2. Para $1 \leq i \leq n$ se Q_i é um quantificador existencial $\exists$ e $x_{i_1}, \dots x_{i_{m_i}}$ as variáveis quantificadas universalmente de índice menor que i então substitui-se:

   $$\exists x_i \theta \text{ por } \theta[f_i(x_{i_1}, \dots x_{i_{m_i}})/x_i]$$

   onde f_i é um novo símbolo funcional de aridade m_i.

3. Converter a matriz da fórmula resultante para forma normal conjuntiva aplicando sucessivamente as seguintes transformações:

$$\begin{array}{lcl} \neg\neg\varphi & \longrightarrow & \varphi \\ \neg(\varphi \wedge \psi) & \longrightarrow & \neg\varphi \vee \neg\psi \\ \neg(\varphi \vee \psi) & \longrightarrow & \neg\varphi \wedge \neg\psi \\ \varphi \vee (\psi \wedge \theta) & \longrightarrow & (\varphi \vee \psi) \wedge (\varphi \vee \theta) \end{array} :$$

4. Aplicar a seguinte transformação: $\forall x(\varphi \wedge \psi) \longrightarrow \forall x\varphi \wedge \forall x\psi$

Exercício 4.1. *Justifica a correção do algoritmo: isto é, que a fórmula resultante verifica as condições indicadas.* ◇

Exercício 4.2. *Aplica o algoritmo à seguinte fórmula:*

$$\forall x(\forall y(P(y) \to R(y,x)) \to Q(x))$$

◇

Resolução 4.2

Para forma normal prenexa temos de eliminar as implicações:

$$\forall x(\exists y(P(y) \wedge \neg R(y,x)) \vee Q(x))$$

e passando os quantificadores:

$$\forall x\exists y((P(y) \wedge \neg R(y,x)) \vee Q(x))$$

Para a Skolemização seja $f \in R_1$ um símbolo funcional novo e substituímos y por $f(x)$ eliminando o quantificador existêncial:

$$\forall x((P(f(x)) \wedge \neg R(f(x),x)) \vee Q(x))$$

Falta apenas converter a matriz para forma normal conjuntiva e distribuir o quantificador universal:

$$\forall x((P(f(x)) \vee Q(x)) \wedge (\neg R(f(x),x) \vee Q(x))) \tag{4.1}$$

$$\forall x((P(f(x)) \vee Q(x))) \wedge \forall x(\neg R(f(x),x) \vee Q(x)) \tag{4.2}$$

Em notação clausal vem:

$$\{P(f(x)), Q(x) \leftarrow, Q(x) \leftarrow R(f(x),x)\}$$

Definição 4.3 (Fórmulas de Horn)**.** *Uma cláusula é uma* fórmula de Horn *(ou uma cláusula de Horn) se tem no máximo um literal positivo. Uma fórmula de Horn é* positiva *se tem um literal positivo:*

$$\forall x_1 \ldots \forall x_s(\alpha_1 \ \vee \ \neg\beta_1 \ \vee \ \ldots \ \vee \ \neg\beta_n)$$

Ou em notação clausal:

$$\alpha_1 \leftarrow \beta_1, \ldots, \beta_n$$

onde α_1 é a diz-se a a cabeça *da cláusula e $\beta_1, \ldots, \beta_n$ o* corpo *da cláusula. Se o corpo é vazio a cláusula diz-se* unitária *(ou* facto*). Uma cláusula de Horn é* negativa *(objectivo) se não tem literal positivo:*

$$\forall x_1 \ldots \forall x_s(\neg\beta_1 \ \vee \ \ldots \ \vee \ \neg\beta_n)$$

que é equivalente a

$$\neg\exists x_1 \ldots \exists x_s(\beta_1 \wedge \ldots \wedge \beta_n)$$

Ou em notação clausal:

$$\leftarrow \beta_1, \ldots, \beta_n$$

A cláusula vazia *representa-se por ϵ corresponde a uma contradição (**F**) e é uma cláusula sem cabeça nem corpo.*

Exemplo 4.2. *Das seguintes fórmulas, indica quais são cláusulas de Horn, positivas ou negativas e escreve-as na notação clausal:*

- $\forall x \forall y \forall z(P(x,z) \ \vee \ Q(x,y) \ \vee \ \neg Q(x,z))$ *Não Horn*
  $P(x,z), Q(x,y) \leftarrow Q(x,z)$
- $\forall x \forall y \forall z(P(x,z) \ \vee \ \neg Q(x,y) \ \vee \ \neg Q(x,z))$ *positiva*
  $P(x,z) \leftarrow Q(x,y), Q(x,z)$
- $\forall x \forall z P(x,z)$ *unitária* $P(x,z) \leftarrow$
- $\forall x \forall y \forall z(\neg P(x,z) \ \vee \ \neg Q(x,y))$ *negativa* $\leftarrow P(x,z), Q(x,y)$

Proposição 4.2. *Seja $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ uma estrutura de $\mathcal{L}$ e φ a cláusula $\alpha_1, \ldots, \alpha_k \leftarrow \beta_1, \ldots, \beta_n$, com variáveis $x_1, \ldots x_s$. $\mathcal{A}$ satisfaz φ, $\mathcal{A} \models \varphi$, se e só se para todos os $a_1, \ldots, a_s \in A$ existe um literal $\lambda \in \{\alpha_1, \ldots, \alpha_k, \neg\beta_1, \ldots, \neg\beta_n\}$ tal que $A \models_s \lambda$ para $s(x_i) = a_i$, $1 \leq i \leq s$*

Demonstração. Resulta directamente da definição de cláusula e da relação $\models_s$. □

Definição 4.4. *Um* programa definido *é um conjunto finito de cláusulas de Horn positivas. Num programa o conjunto de cláusulas com cabeças com mesmo símbolo de predicado P, chama-se a* definição *de P.*

Exemplo 4.3. *Considera a linguagem de* $1^{\underline{a}}$ *ordem* $\mathcal{L}_{ss}$ *sem igualdade com* $\mathcal{F}_0 = \{0, \mathsf{nil}\}$, $\mathcal{F}_1 = \{\mathsf{s}\}$, $\mathcal{F}_2 = \{\mathsf{cons}\}$, $\mathcal{R}_1 = \{\mathsf{sorted}\}$, $\mathcal{R}_2 = \{\mathsf{slowsort}, \mathsf{perm}, \mathsf{less\_eq}\}$ *e* $\mathcal{R}_3 = \{\mathsf{delete}\}$. *O programa seguinte dada uma sequência de inteiros permuta os seus elementos até estarem ordenados:*

$$\mathsf{slowsort}(\mathsf{x},\mathsf{y}) \leftarrow \mathsf{perm}(\mathsf{x},\mathsf{y}), \mathsf{sorted}(\mathsf{y})$$
$$\mathsf{sorted}(\mathsf{nil}) \leftarrow$$
$$\mathsf{sorted}(\mathsf{cons}(\mathsf{x},\mathsf{nil})) \leftarrow$$
$$\mathsf{sorted}(\mathsf{cons}(\mathsf{x},\mathsf{cons}(\mathsf{y},\mathsf{z}))) \leftarrow \mathsf{less\_eq}(\mathsf{x},\mathsf{y}), \mathsf{sorted}(\mathsf{cons}(\mathsf{y},\mathsf{z}))$$
$$\mathsf{perm}(\mathsf{nil},\mathsf{nil}) \leftarrow$$
$$\mathsf{perm}(\mathsf{cons}(\mathsf{x},\mathsf{y}),\mathsf{cons}(\mathsf{u},\mathsf{v})) \leftarrow \mathsf{delete}(\mathsf{u},\mathsf{cons}(\mathsf{x},\mathsf{y}),\mathsf{z}), \mathsf{perm}(\mathsf{z},\mathsf{v})$$
$$\mathsf{delete}(\mathsf{x},\mathsf{cons}(\mathsf{x},\mathsf{y}),\mathsf{y}) \leftarrow$$
$$\mathsf{delete}(\mathsf{x},\mathsf{cons}(\mathsf{y},\mathsf{z}),\mathsf{cons}(\mathsf{y},\mathsf{w})) \leftarrow \mathsf{delete}(\mathsf{x},\mathsf{z},\mathsf{w})$$
$$\mathsf{less\_eq}(0,\mathsf{x})$$
$$\mathsf{less\_eq}(\mathsf{s}(\mathsf{x}),\mathsf{s}(\mathsf{y})) \leftarrow \mathsf{less\_eq}(\mathsf{x},\mathsf{y})$$

onde:

- *um inteiro* n *é representado pelo termo* $\mathsf{s}(\mathsf{s}(...\mathsf{s}(0)))$ *com* n *símbolos funcionais* s. *Ex:* $\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))$ *representa 3*
- *uma lista de inteiros é representada por termos* $\mathsf{cons}(\mathsf{x},\mathsf{y})$ *onde* x *é um inteiro e* y *é uma lista. A lista vazia é representada por* nil *Ex:* $\mathsf{cons}(\mathsf{s}(\mathsf{s}(0)),\mathsf{cons}(\mathsf{s}(0),\mathsf{cons}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))),\mathsf{nil})$ *representa a lista* $[2,1,3]$
- *slowsort dada uma lista* x*, verifica se está ordenada (sorted), senão permuta (perm)...*

Proposição 4.3. *Seja* $\mathcal{P}$ *um programa,* $\leftarrow \beta_1, \ldots, \beta_n$ *um objectivo e* $y_1, \ldots, y_r$ *as variáveis que ocorrem nos* β_i, $1 \leq i \leq n$. *Então*

$$\mathcal{P} \models \exists y_1 \ldots \exists y_r \beta_1 \wedge \ldots \wedge \beta_n$$

se e só se $\mathcal{P} \cup \{\leftarrow \beta_1, \ldots, \beta_n\}$ *é não satisfazível.*

Demonstração. Directamente das definições. □

Exemplo 4.4. *Verifica que*

$$P \cup \{\leftarrow \mathsf{slowsort}(\mathsf{cons}(\mathsf{s}(\mathsf{s}(0)),\mathsf{cons}(\mathsf{s}(0),\mathsf{cons}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))),\mathsf{nil}),\mathsf{y})\}$$

não é satisfazível.Ou por outras palavras executa P *com o objectivo*

$$\leftarrow \mathsf{slowsort}(\mathsf{cons}(\mathsf{s}(\mathsf{s}(0)),\mathsf{cons}(\mathsf{s}(0),\mathsf{cons}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0))),\mathsf{nil}),\mathsf{y})$$

e obtém em y *a resposta.*

Exercício 4.3. *Considera o seguinte programa definido P:*

$$Add(x, 0, x) \leftarrow$$
$$Add(x, s(y), s(z)) \leftarrow Add(x, y, z)$$

e o objectivo G:

$$\leftarrow Add(s(0), s(s(0)), x)$$

1. *Escreve cada fórmula do programa e do objectivo sem ser na* notação clausal *(i.e com os quantificadores e operações lógicas usuais).*

2. *Mostra que $P \cup \{G\} \vdash \boldsymbol{F}$ usando o sistema de dedução natural (e a notação de Fitch) e as fórmulas na forma obtida em 1.*

◇

Exemplo 4.5. *Justifica a validade ou a falsidade das seguintes afirmações, para uma linguagem de $1^{\underline{a}}$ ordem:*

1. *todo o conjunto de fórmulas de Horn positivas é satisfazível*

2. *todo o conjunto de fórmulas de Horn negativas é satisfazível*

3. *todo o conjunto de fórmulas de Horn é satisfazível*

4.1.2 Satisfazibilidade de Cláusulas

Para determinar que um conjunto de fórmulas Φ não é satisfazível é necessário que nenhuma estrutura de $\mathcal{L}$ seja modelo de Φ. Mas se as fórmulas forem cláusulas basta mostrar que não têm um modelo de um dado tipo: um *modelo de Herbrand*. Mais ainda se as claúsulas forem de Horn, são satisfazíveis se o forem no seu modelo mínimo de Herbrand. Dado um programa definido P e um objectivo G, $P \cup G$ é não satisfazível se e só se G não é satisfazível no modelo mínimo de P.
Começamos por definir estrutura de Herbrand.

Definição 4.5. *Uma estrutura $\mathcal{A} = (A, \cdot^{\mathcal{A}})$ de $\mathcal{L}$ é* estrutura de Herbrand *se:*

- $A = \mathcal{T}_0$, *o conjunto de termos fechados de $\mathcal{L}$*
- $c^{\mathcal{A}} = c$, *para $c \in \mathcal{F}_0$*
- $f^{\mathcal{A}}(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$, *para $f \in \mathcal{F}_n$, $n > 0$ e $t_1, \ldots, t_n \in \mathcal{T}_0$*

Todas as estruturas de Herbrand têm o mesmo domínio (ou universo) e coincidem no valor dos símbolos funcionais: apenas diferem no valor dos símbolos de predicado.

Exemplo 4.6. *Para a linguagem $\mathcal{L}_{ss}$ (do slowsort) o domínio (ou universo) de qualquer estrutura de Herbrand $\mathcal{A}$ é:*

$$A = \{0,nil,s(0),s(nil),cons(0,0),cons(0,nil),cons(nil,0),cons(nil,nil),\dots\}$$

e $0^{\mathcal{A}} = 0$, $\mathsf{nil}^{\mathcal{A}} = \mathsf{nil}$, $\mathsf{s}^{\mathcal{A}}(0) = \mathsf{s}(0)$, *etc*

Exemplo 4.7. *Seja a linguagem $\mathcal{L}$ tal que $\mathcal{F}_1 = \{\mathsf{f},\mathsf{g}\}$, $\mathcal{R}_1 = \{\mathsf{p},\mathsf{r}\}$ e $\mathcal{R}_2 = \{\mathsf{q}\}$. Como $\mathcal{L}$ não tem constantes adicionamos uma constante c a $\mathcal{T}_0$ de $\mathcal{L}$. O universo de Herbrand é:*

$$A = \{c,f(c),g(c),f(f(c)),f(g(c)),g(f(c)),g(g(c)),\dots\}$$

e $\mathsf{c}^{\mathcal{A}} = \mathsf{c}$, $\mathsf{f}^{\mathcal{A}}(\mathsf{c}) = \mathsf{f}(\mathsf{c})$, $\mathsf{f}^{\mathcal{A}}(\mathsf{g}(\mathsf{c})) = \mathsf{f}(\mathsf{g}(\mathsf{c}))$, *etc*

Teorema 4.1. *Seja Φ um conjunto de cláusulas. Então Φ é satisfazível se e só se tiver um modelo de Herbrand.*

Demonstração. ($\Leftarrow$) É óbvio que se φ tiver um modelo de Herbrand é satisfazível
($\Rightarrow$) Seja $\mathcal{A}$ um modelo de Φ. É necessário mostrar que Φ tem um modelo de Herbrand, $\mathcal{A}_{\mathcal{H}}$: associamos a cada símbolo relacional R a relação:

$$R^{\mathcal{A}_{\mathcal{H}}} = \{(t_1,\dots,t_n) \in \mathcal{T}_0^n \mid (t_1^{\mathcal{A}},\dots,t_n^{\mathcal{A}}) \in R^{\mathcal{A}}\}$$

Vejamos que $\mathcal{A}_{\mathcal{H}}$ é um modelo de Φ: seja $\forall(\lambda_1 \vee \dots \vee \lambda_m) \in \Phi$.

$$\mathcal{A}_{\mathcal{H}} \models \forall(\lambda_1 \vee \dots \vee \lambda_m)$$

sse para toda a interpretação das variáveis $s_{\mathcal{H}}$ em $\mathcal{A}_{\mathcal{H}}$ se tem

$$\mathcal{A}_{\mathcal{H}} \models_{s_{\mathcal{H}}} (\lambda_1 \vee \dots \vee \lambda_m)$$

Se $s_{\mathcal{H}}(x) = t \in \mathcal{T}_0$ seja $s(x) = t^{\mathcal{A}}$ em $\mathcal{A}$. Como $\mathcal{A} \models \forall(\lambda_1 \vee \dots \vee \lambda_m)$, então $\mathcal{A} \models_s (\lambda_1 \vee \dots \vee \lambda_m)$, isto é, existe pelo menos um k tal que $\mathcal{A} \models_s \lambda_k$. Mas, por construção de $\mathcal{A}_{\mathcal{H}}$ e como cada λ_i é um literal, $\mathcal{A} \models_s \lambda_i$ sse $\mathcal{A}_{\mathcal{H}} \models_{s_{\mathcal{H}}} \lambda_i$, $1 \leq i \leq m$ (verifica!). Donde

$$\mathcal{A}_{\mathcal{H}} \models_{s_{\mathcal{H}}} (\lambda_1 \vee \dots \vee \lambda_m) \text{ sse } \mathcal{A} \models_s (\lambda_1 \vee \dots \vee \lambda_m)$$

. □

Nota que se φ não fosse um conjunto de cláusulas o teorema não se verificava.

Exemplo 4.8. *O conjunto de proposições* $\{\mathsf{p}(\mathsf{c}), \exists x \neg \mathsf{p}(x)\}$ *da linguagem com* $\mathcal{F}_0 = \{\mathsf{c}\}$ *e* $\mathcal{R}_1 = \{\mathsf{p}\}$ *é satisfazível mas não tem modelo de Herbrand:* $\mathcal{A} = (\{0,1\}, \cdot^{\mathcal{A}})$ *com* $\mathsf{c}^{\mathcal{A}} = 0$ *e* $\mathsf{p}^{\mathcal{A}} = \{0\}$ *satisfaz o conjunto. Mas* $\mathcal{T}_0 = \{\mathsf{c}\}$ *e as únicas estruturas de Herbrand são tais que* $\mathsf{p}^{\mathcal{H}_1} = \emptyset$ *e* $\mathsf{p}^{\mathcal{H}_2} = \{\mathsf{c}\}$*, e nenhuma satisfaz o conjunto.*

Corolário 4.1. *Um conjunto de cláusulas de Horn é satisfazível sse tiver um modelo de Herbrand.*

Definição 4.6. *Dado um conjunto de cláusulas de Horn* Φ *o* modelo mínimo de Herbrand *de* Φ *é a estrutura de Herbrand* $\mathcal{A}_\Phi$ *tal que, para* $R \in \mathcal{R}_n$*:*

$$R^{\mathcal{A}_\Phi} = \{(t_1, \ldots, t_n) \in \mathcal{T}_0^n \mid \Phi \vdash R(t_1, \ldots, t_n)\}$$

O modelo da definição anterior é *mínimo* porque:

Teorema 4.2. *Seja* Φ *um conjunto satisfazível de cláusulas de Horn. Então:*

- $\mathcal{A}_\Phi \models \Phi$
- *se* $\mathcal{A}_\mathcal{H}$ *é um modelo de Herbrand de* Φ*, então para todo o símbolo relacional n-ário* R*,* $R^{\mathcal{A}_\Phi} \subseteq R^{\mathcal{A}_\mathcal{H}}$

Demonstração. Para toda a fórmula atómica α, com variáveis $x_1, \ldots, x_k$ e para toda a interpretação de variáveis s, se $s(x_i) = t_i$, $1 \leq i \leq k$, então, pela definição de $\mathcal{A}_\Phi$, tem-se:

$$\mathcal{A}_\Phi \models_s \alpha \text{ sse } \Phi \vdash \alpha[t^k/x^k] \tag{4.3}$$

onde $\alpha[t^k/x^k]$ é a fórmula que resulta de α substituindo simultaneamente todas as ocorrências de $x_1, \ldots, x_k$ por $t_1, \ldots, t_k$.
Seja $\forall x_1 \ldots \forall x_m \theta \in \Phi$, e θ não tem quantificadores. Vamos mostrar que se tem $\mathcal{A}_\Phi \models_s \theta$ qualquer que seja a interpretação s.
Temos dois casos:

$\theta \equiv (\neg\alpha_1 \vee \ldots \vee \neg\alpha_n \vee \alpha)$**, com** $n \geq 0$ Temos que demonstrar que $\mathcal{A}_\Phi \models_s (\alpha_1 \wedge \ldots \wedge \alpha_n) \rightarrow \alpha$, para toda interpretação s. Se $n = 0$, então θ é uma fórmula atómica e o resultado segue de (4.3). Se $n > 0$, se $\mathcal{A}_\Phi \models_s \alpha_1 \wedge \ldots \wedge \alpha_n$, então $\mathcal{A}_\Phi \models_s \alpha_i$, $1 \leq i \leq n$. Mas, de (4.3), $\Phi \vdash \alpha_1[t^m/x^m]$, $\ldots$, $\Phi \vdash \alpha_m[t^m/x^m]$. E então, $\Phi \vdash \alpha[t^m/x^m]$, porque $\Phi \vdash ((\alpha_0 \wedge \ldots \wedge \alpha_n) \rightarrow \alpha)[t^m/x^m]$. Temos então que $\mathcal{A}_\Phi \models_s \theta$.

$\theta \equiv (\neg\alpha_0 \vee \ldots \vee \neg\alpha_n)$ De $\forall x_1 \ldots \forall x_m \theta \in \Phi$, vem que $\Phi \vdash \theta[t^m/x^m]$ (aplicando m vezes a regra $\forall\mathbf{E}$). Logo, $\Phi \vdash \neg(\alpha_0 \wedge \ldots \wedge \alpha_n)[t^m/x^m]$. Se $\mathcal{A}_\Phi \not\models_s (\neg\alpha_0 \vee \ldots \vee \neg\alpha_n)$, então $\mathcal{A}_\Phi \models_s \alpha_0$, $\ldots \mathcal{A}_\Phi \models_s \alpha_n$. Mas, por (4.3), $\Phi \vdash \alpha_0[t^m/x^m]$,$\ldots$, $\Phi \vdash \alpha_n[t^m/x^m]$, donde $\Phi \vdash (\alpha_0 \wedge \ldots \wedge \alpha_n)[t^m/x^m]$. Absurdo! Então, $\mathcal{A}_\Phi \models_s \theta$.

Seja $\mathcal{A}_{\mathcal{H}}$ um modelo de Herbrand de Φ e R um símbolo relacional. Se $(t_1, \ldots, t_n) \in R^{\mathcal{A}_\Phi}$, então $\Phi \vdash R(t_1, \ldots, t_n)$, e pela integridade, $\mathcal{A}_{\mathcal{H}} \models R(t_1, \ldots, t_n)$, i.e., $(t_1, \ldots, t_n) \in R^{\mathcal{A}_{\mathcal{H}}}$ □

O teorema seguinte garante que dado um programa definido P (conjunto de cláusulas de Horn positivas) e um objectivo G (uma cláusula de Horn negativa) , $P \cup \{G\}$ é não satisfazível se e só se G não é satisfeito em $\mathcal{A}_P$.

Teorema 4.3. *Seja Φ um conjunto de cláusulas de Horn. Para qualquer fórmula fechada da forma $\exists x_1 \ldots \exists x_n(\alpha_0 \wedge \ldots \wedge \alpha_l)$, onde cada α_i é uma fórmula atómica, as seguintes afirmações são equivalentes:*

1. $\Phi \vdash \exists x_1 \ldots \exists x_n(\alpha_0 \wedge \ldots \wedge \alpha_l)$
2. $\mathcal{A}_\Phi \models \exists x_1 \ldots \exists x_n(\alpha_0 \wedge \ldots \wedge \alpha_l)$
3. *existem* $t_1, \ldots, t_n \in \mathcal{T}_0$, *tal que* $\Phi \vdash (\alpha_0 \wedge \ldots \wedge \alpha_l)[t^n/x^n]$

Demonstração.
(1)⇒(2) pela integridade da LPO.
(2)⇒(3) Se $\mathcal{A}_\Phi \models \exists x_1 \ldots \exists x_n(\alpha_0 \wedge \ldots \wedge \alpha_l)$ existe uma interpretação das variáveis s, com $s(x_i) = t_i$, $1 \leq i \leq n$, tal que $\mathcal{A}_\Phi \models_s (\alpha_0 \wedge \ldots \wedge \alpha_l)$. Logo $\mathcal{A}_\Phi \models_s \alpha_0$, $\ldots$,$\mathcal{A}_\Phi \models_s \alpha_l$, e, por (4.3), $\Phi \vdash \alpha_0[t^n/x^n], \ldots \Phi \vdash \alpha_l[t^n/x^n]$. E aplicando a regra $\wedge\mathbf{I}$ l vezes, $\Phi \vdash (\alpha_0 \wedge \ldots \wedge \alpha_l)[t^n/x^n]$.
(3)⇒ (1) Basta aplicar n vezes a regra $\exists\mathbf{I}$. □

Exercício 4.4. *Seja $\mathcal{L}$ uma linguagem de $1^{\underline{a}}$ ordem sem igualdade e Φ um conjunto de proposições.*

1. *Mostra que $\Phi \vdash \exists x\psi$ não implica necessariamente que existe um termo $t \in \mathcal{T}$ tal que $\Phi \vdash \psi[t/x]$*
2. *Supõe que Φ é um conjunto de cláusulas de Horn e ψ da forma $\alpha_0 \wedge \ldots \wedge \alpha_l$ onde α_i são fórmulas atómicas. Mostra que se $\Phi \vdash \exists x\psi$ existe $t \in \mathcal{T}_0$ tal que $\Phi \vdash \psi[t/x]$.*

◇

Resolução 4.4

1. Basta considerar $\mathcal{R}_1 = \{R\}$, $\Phi = \{\exists x R(x)\}$ e ψ a fórmula $R(x)$.
2. Segue do teorema anterior.

Seja P um programa definido. Então:

L_P linguagem de primeira ordem constituída por todos os símbolos não lógicos de P (eventualmente com mais uma constante c). Ex: Seja P

$$\mathsf{p(x)} \leftarrow \mathsf{q(f(x), g(x))}$$
$$\mathsf{r(x)} \leftarrow$$

$\mathcal{L}_P$ é dada por $\mathcal{F}_0 = \{\mathsf{c}\}$, $\mathcal{F}_1 = \{\mathsf{f}, \mathsf{g}\}$, $\mathcal{R}_1 = \{\mathsf{p}, \mathsf{r}\}$ e $\mathcal{R}_2 = \{\mathsf{q}\}$

U_P O *universo de Herbrand de* P, isto é, o conjunto dos termos fechados da linguagem $\mathcal{L}_P$. Ex:

$U_P = \{\mathsf{c},\mathsf{f(c)},\mathsf{g(c)},\mathsf{f(f(c))},\mathsf{f(g(c))},\mathsf{g(f(c))},\mathsf{g(g(c))},\dots\}$

B_P A *base de Herbrand* de $\mathcal{L}_P$, é o conjunto de todas as fórmulas atómicas fechadas da linguagem $\mathcal{L}_P$. Ex:

$B_P = \{\mathsf{p(c)}, \mathsf{q(c,c)}, \mathsf{r(c)}, \mathsf{p(f(c))}, \mathsf{p(g(c))}, \mathsf{q(f(c),c)}, \dots\}$

$I_{\mathcal{A}}$ Cada estrutura de Herbrand de $\mathcal{L}_P$ fica identificada por um subconjunto de B_P (dos átomos que são verdadeiros nessa estrutura)...Isto é, dada uma estrutura de Herbrand $\mathcal{A}$, existe um e um só $I_{\mathcal{A}} \subseteq B_P$ tal que, para cada símbolo relacional n-ário R de $\mathcal{L}_P$ e para $t_1, \dots, t_n \in U_P$:

$$R(t_1, \dots, t_n) \in I_{\mathcal{A}} \text{ sse } (t_1, \dots, t_n) \in R^{\mathcal{A}}$$

Ex: Seja $\mathcal{A}$ tal que $\mathsf{p}^{\mathcal{A}} = \mathsf{r}^{\mathcal{A}} = \{\mathsf{c}, \mathsf{f(c)}, \mathsf{g(c)}\}$ e $\mathsf{q}^{\mathcal{A}} = \emptyset$. Então,

$$I_A = \{\mathsf{p(c)}, \mathsf{p(f(c))}, \mathsf{p(g(c))}, \mathsf{r(c)}, \mathsf{r(f(c))}, \mathsf{r(g(c))}\}$$

. Nota que $\mathcal{A} \not\models P$ (porquê?).

M_P $M_P = I_{\mathcal{A}_P}$, onde $\mathcal{A}_P$ é modelo mínimo de Herbrand. Ex: $M_P = \{\mathsf{r(t)} \mid \mathsf{t} \in U_P\}$

Corolário 4.2. *Seja P um programa definido. Então*

$$M_P = \{\varphi \in B_P \mid P \models \varphi\}$$

Exercício 4.5. *Mostra o corolário anterior.* $\diamond$

Exercício 4.6. *Mostra que $M_P \subseteq I_{\mathcal{A}}$ para qualquer modelo de Herbrand $\mathcal{A}$ de P.* $\diamond$

Exercício 4.7. *Para cada um dos programas definidos P e objectivos G descritos abaixo determina:*

- $\mathcal{L}_P$

- *cada fórmula do programa e do objectivo sem ser na* notação clausal *(i.e com os quantificadores e operações lógicas* $\vee$, $\wedge$ *e* $\neg$*)*
- *um modelo de Herbrand* $\mathcal{A}$ *de* P
- U_P
- B_P
- $I_{\mathcal{A}}$ *para o modelo de Herbrand* $\mathcal{A}$ *que escolheste*
- *para* $G = \leftarrow \alpha_1, \ldots, \alpha_k$ *determina os termos fechados* $t_1, \ldots, t_n$ *tais que* $P \models (\alpha_1 \wedge \ldots \wedge \alpha_k)[t_1/x_1, \ldots, t_n/x_n]$*, onde* $x_1, \ldots, x_n$ *são as variáveis que ocorrem em* G *ou mostra que* $P \cup \{G\}$ *é satisfazível.*

a) P : $Nesimo(s(0), cons(x, y), x) \leftarrow$
$Nesimo(s(w), cons(y, z), x) \leftarrow Nesimo(w, z, x)$

G : $\leftarrow Nesimo(x, cons(y, cons(z, nil)), z)$

b) P : $Delete(x, cons(x, y), y) \leftarrow$
$Delete(x, cons(y, z), cons(y, w)) \leftarrow Delete(x, z, w)$

G : $\leftarrow Delete(y, cons(x, cons(y, nil), z)$

c) P : $Max(x, y, x) \leftarrow Less\_eq(y, x)$
$Max(x, y, y) \leftarrow Less\_eq(x, y)$
$Less\_eq(0, x) \leftarrow$
$Less\_eq(s(x), s(y)) \leftarrow Less\_eq(x, y)$

G : $\leftarrow Max(s(0), x, s(s(0)))$

$\diamond$

4.2 Unificação

Definição 4.7. *Uma* substituição *é uma função* $\sigma : \mathcal{V}ar \longrightarrow \mathcal{T}$ *tal que o conjunto dos* $x_i \in \mathcal{V}ar$ *com* $\sigma(x_i) = t_i \neq x_i$ *é finito. E escreve-se*

$$\sigma = [t_1/x_1, \ldots, t_n/x_n]$$

A substituição identidade *é a substituição* ι *tal que* $\iota(x) = x$*, para todo o* $x \in \mathcal{V}ar$*.*
Uma substituição $[t_1/x_1, \ldots, t_n/x_n]$ *é* fechada *se todos os* t_i *são termos fechados.*

Por exemplo $[f(a)/x, b/y, g(b,b)/z]$ é uma substituição fechada, mas não $[f(y)/x, x/y, a/z]$

Definição 4.8. *Seja uma* expressão *E um termo, um literal, uma conjunção ou uma disjunção de literais. Uma expressão* simples *é um termo ou um átomo.*
Seja $\sigma = [t_1/x_1, \ldots, t_n/x_n]$ e E uma expressão, $E\sigma$ (ou $\sigma(E)$), uma instância *de E é a expressão que resulta de E substituindo simultaneamente todas as ocorrências de $x_1, \ldots, x_n$ por $t_1, \ldots, t_n$. $E\sigma$ é uma* instância fechada *se não contiver variáveis.*
Se $S = \{E_1, \ldots, E_m\}$, é um conjunto finito de expressões E_i, e σ uma substituição, então $S\sigma = \{E_1\sigma, \ldots, E_m\sigma\}$

Exemplo 4.9. *Seja*
$E = P(x, y, f(a))$ *e* $\sigma = [b/x, x/y]$, $E\sigma$ *é* $P(b, x, f(a))$,
$E = f(x, g(y, z), z)$ *e* $\sigma = [g(y,z)/x, a/z, z/y, f(f(b))/u]$, $E\sigma$ *é* $f(g(y,z), g(z,a), a)$,
$E = P(f(x), x)$ *e* $\sigma = [a/y]$, $E\sigma$ *é* $P(f(x), x)$.

Definição 4.9. *Sejam $\theta = [s_1/x_1, \ldots, s_n/x_n]$ e $\sigma = [t_1/y_1, \ldots, t_n/y_n]$ duas substituições, a* substituição composta *$\theta\sigma : \mathcal{V}ar \longrightarrow \mathcal{T}$, é obtida de*

$$[s_1\sigma/x_1, \ldots, s_n\sigma/x_n, t_1/y_1, \ldots, t_n/y_n]$$

retirando os $s_i\sigma = x_i$ e todos t_j/y_j tal que $y_j \in \{x_1, \ldots, x_n\}$.

Sendo $\theta = [f(z)/x, a/y, y/z]$ e $\sigma = [a/x, z/y, b/z, d/u]$, a composta é

$$\theta\sigma = [f(b)/x, a/y, d/u]$$

Sendo $\theta = [c/x, f(z)/y, u/v]$ e $\sigma = [v/u, x/z]$, a composta é

$$\theta\sigma = [c/x, f(x)/y, v/u, x/z]$$

Proposição 4.4. *Sendo θ, σ e γ substituições e E uma expressão tem-se que:*

a) $\theta\iota = \iota\theta = \theta$

b) $E(\theta\sigma) = (E\theta)\sigma$

c) $(\theta\sigma)\gamma = \theta(\sigma\gamma)$

Exercício 4.8. *Mostra a proposição anterior.* $\diamond$

Definição 4.10. *Uma substituição σ é* idempotente *se $\sigma\sigma = \sigma$.*

Definição 4.11. *Dada uma expressão E, seja V o conjunto das variáveis que ocorrem em E. Uma substituição $\theta = [y_1/x_1, \ldots, y_n/x_n]$ é uma* renomeação de variáveis *para E se o conjunto $\{x_1, \ldots, x_n\}$ está contido em V, todos y_i são distintos e $(V \backslash \{x_1, \ldots, x_n\}) \cap \{y_1, \ldots, y_n\} = \emptyset$.*

Definição 4.12. *Duas expressões E e F são* variantes *se existirem substituições σ e τ tais que $E = F\sigma$ e $F = E\tau$.*

Exemplo 4.10. *$P(f(x,y), g(z), a)$ é uma variante de $P(f(y,x), g(u), a)$. $P(x,x)$ não é uma variante de $P(x,y)$.*

Proposição 4.5. *Sejam E e F duas variantes. Então existem substituições θ e σ tais que $E = F\theta$ e $F = E\sigma$ e θ e σ são renomeações de variáveis, respectivamente para F e E.*

Demonstração. Sejam θ_1 e σ_1 substituições tais que $E = F\theta_1$ e $F = E\sigma_1$. Seja V o conjunto das variáveis que ocorrem em E e seja σ a obtida de σ_1 retirando os elementos t/x, tais que $x \notin V$. Então $F = E\sigma$ e $E = F\theta_1 = E\sigma\theta_1$. Logo, σ tem de ser uma renomeação de variáveis. □

Seja S um conjunto finito de expressões simples.

Definição 4.13. *Um* unificador *de S é uma substituição θ tal que $S\theta$ tem exactamente um elemento. E diz-se que S é* unificável. *Um unificador θ de S é um* unificador mais geral *(*umg*) de S, se para qualquer unificador σ de S existir uma substituição γ tal que $\sigma = \theta\gamma$.*

Exemplo 4.11. *$\{P(f(x), z), P(y, a)\}$ é unificável: $[f(a)/y, a/x, a/z]$ é um unificador e $[f(x)/y, a/z]$ é um unificador mais geral. E $\sigma = \theta[a/x]$.*
$\{P(y, g(u,z)), P(x,x)\}$ é unificável: $[g(u,z)/y, g(u,z)/x]$ é um unificador mais geral
$\{P(f(x), a), P(y, f(w))\}$ não é unificável: pois a e $f(w)$ não são unificáveis.

Se θ e σ forem dois *umg*s de S os elementos de $S\theta$ e $S\sigma$ são variantes (porquê?). Então, pela Proposição 4.5, os *umg*s de S são únicos a menos de renomeação de variáveis.

Definição 4.14. *O* conjunto de diferenças *de um conjunto finito S de expressões simples, é dado por: localizar a posição mais à esquerda, na qual nem todas as expressões de S têm o mesmo símbolo, e extrair de cada uma a subexpressão que começa nessa posição.*

Exemplo 4.12. *Para $\{R(f(x), g(h(z)), a), R(f(x), g((f(u)), b), R(f(x), g((w), a)\}$ o conjunto de diferenças é $\{h(z), f(u), w\}$. Para $\{R(f(a), g(x)), R(y,y)\}$ o conjunto de diferenças é $\{f(a), y\}$.*

4.2.1 Algoritmo da Unificação (de Robinson)

Dado um conjunto finito S de expressões simples, vamos descrever um algoritmo que retorna um *umg* de S, se e só se, S for unificável:

1. Seja $k = 0$ e $\sigma_0 = \iota$

2. Se $S\sigma_k$ contém exactamente um elemento, então parar e retornar σ_k como *umg* de S. Caso contrário, determinar o conjunto de diferenças D_k de $S\sigma_k$

3. Se existirem v e t em D_k tais que v é uma variável que não ocorre em t, então $\sigma_{k+1} = \sigma_k[t/v]$, incrementar k e voltar para 2. Caso contrário, parar e indicar que S não é unificável.

Exemplo 4.13. $S = \{R(f(x), g(h(z)), a), R(y, g(f(u)), b), R(f(x), g(w), a)\}$

- $\sigma_0 = \iota$
- $D_0 = \{f(x), y\}$, $\sigma_1 = [f(x)/y]$ *e*
  $S\sigma_1 = \{R(f(x), g(h(z)), a), R(f(x), g(f(u)), b), R(f(x), g(w), a)\}$
- $D_1 = \{h(z), f(u), w\}$, $\sigma_2 = [f(x)/y][h(z)/w] = [f(x)/y, h(z)/w]$ *e*
  $S\sigma_2 = \{R(f(x), g(h(z)), a), R(f(x), g(f(u)), b)\}$
- $D_2 = \{h(z), f(u)\}$.
  Logo, S não é unificável.

Exemplo 4.14. *Seja* $S = \{P(a, x, h(g(z))), P(z, h(y), h(y))\}$. *Então,*

- $\sigma_0 = \iota$
- $D_0 = \{a, z\}$, $\sigma_1 = [a/z]$ *e* $S\sigma_1 = \{P(a, x, h(g(a))), P(a, h(y), h(y))\}$
- $D_1 = \{x, h(y)\}$, $\sigma_2 = [a/z, h(y)/x]$ *e*
  $S\sigma_2 = \{P(a, h(y), h(g(a))), P(a, h(y), h(y))\}$
- $D_2 = \{y, g(a)\}$,$\sigma_3 = [a/z, h(y)/x, g(a)/y]$ *e*
  $S\sigma_2 = \{P(a, h(g(a)), h(g(a)))\}$.
  Logo, S é o unificável e σ_3 *um* umg *de* S.

Exemplo 4.15. $S = \{P(x, x), P(y, f(y))\}$.

- $\sigma_0 = \iota$
- $D_0 = \{x, y\}$, $\sigma_1 = [y/x]$ *e* $S\sigma_1 = \{P(y, y), P(y, f(y))\}$
- $D_1 = \{f(y), y\}$. *Como y ocorre em* $f(y)$, *S não é unificável.*

Teorema 4.4. ***(do algoritmo da unificação)*** *Seja um conjunto finito S de expressões simples. Se S é unificável, então o algoritmo da unificação termina e retorna um* umg *de S. Se S não é unificável então o algoritmo da unificação termina e reporta esse facto.*

Demonstração. O algoritmo termina sempre: S só tem um número finito de variáveis e no passo 3 uma variável é eliminada. Se S não é unificável, pára no passo 3 e reporta o facto. Se S é unificável, seja θ um unificador de S. Então o algoritmo têm de parar no passo 2. Basta provar que se σ_k é a substituição da $k+1$-ésima iteração, então existe γ_k tal que $\theta = \sigma_k\gamma_k$ (e quanto parar, σ_k é um unificador de S). Por indução em k:

Base. Para k=0, basta $\gamma_0 = \theta$

Indução. Suponhamos que para k existe γ_k tal que $\theta = \sigma_k\gamma_k$, e $|S\sigma_k| > 1$. Seja D_k o conjunto de diferenças de $S\sigma_k$. Como $\theta = \sigma_k\gamma_k$ e θ unifica S, γ_k unifica D_k. Podemos concluir que D_k tem pelo menos uma variável v' (sendo um conjunto de diferenças, não podia ter termos todos com o mesmo símbolo funcional, mas para unificar não podem ser distintos, logo tem de haver uma variável). Seja t' outro termo de D_k. Então $v'\gamma_k = t'\gamma_k$. Logo v' não ocorre em t'. Seja então, $\sigma_{k+1} = \sigma_k[t'/v']$. E definimos $\gamma_{k+1} = \gamma_k \setminus [v'\gamma_k/v']$. Se $v'\gamma_k/v' \in \gamma_k$ então

$$\begin{aligned} \gamma_k &= [v'\gamma_k/v'] \cup \gamma_{k+1} && (4.4)\\ &\quad [t'\gamma_k/v'] \cup \gamma_{k+1} && (4.5)\\ &\quad [t'\gamma_{k+1}/v'] \cup \gamma_{k+1} \text{ não ocorre em } t' && (4.6)\\ &\quad [t'/v']\gamma_{k+1} \text{ definição de composta} && (4.7) \end{aligned}$$

Senão, $\gamma_{k+1} = \gamma_k$, todos os elementos de D_k são variáveis e $\gamma_k = [t'/v']\gamma_{k+1}$.

Então, $\theta = \sigma_k\gamma_k = \sigma_k[t'/v']\gamma_{k+1} = \sigma_{k+1}\gamma_{k+1}$.

□

Exemplo 4.16. ***(Regra de Resolução)*** *A regra da resolução para cláusulas de primeira ordem pode ser definida usando a noção de unificação.*
Dado $L = \{l_1, \ldots, l_n\}$ um conjunto de literais, seja $\overline{L} = \{\tilde{l_1}, \ldots, \tilde{l_n}\}$. Sejam C_1 e C_2 duas claúsulas sem variáveis em comum. Seja $L_1 \subseteq C_1$ e $L_2 \subseteq C_2$, tal que L_1 e $\overline{L_2}$ unificam com umg σ.
*A **resolvente** C de C_1 e C_2 é:*

$$(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

Sendo $\{p(f(x), g(y)), q(x, y)\}$ e $\{\neg p(f(f(a)), g(z)), q(f(a), g(z))\}$. Um umg de $\{p(f(x), g(y))\}$ e $\{p(f(f(a)), g(z))\}$ é $[f(a)/x, y/z]$. Então, uma resolvente é

$$\{q(f(a), z), q(f(a), g(z))\}$$

4.3 O operador T_P

Definição 4.15. *Dado um programa definido P e um conjunto fórmulas atómicas fechadas $I \in B_P$,*

$$\begin{aligned} T_p(I) &= \{\alpha \in B_P \mid \alpha \leftarrow \beta_1, \ldots, \beta_n \text{ é uma instância fechada} \\ &\quad \text{duma cláusula de } P \text{ e } \{\beta_1, \ldots, \beta_n\} \subseteq I\} \end{aligned}$$

Para $n \in (N)$, definimos $T_P \uparrow n$ indutivamente por:

$$\begin{aligned} T_P \uparrow 0 &= \emptyset \\ T_P \uparrow n &= T_P(T_P \uparrow (n-1)) \end{aligned}$$

e ainda

$$T_P \uparrow \omega = \cup_{n \in \mathbb{N}} T_P \uparrow n$$

Nota que

$$\begin{aligned} T_P \uparrow 1 &= T_P(\emptyset) \\ &= \{\alpha \in B_P \mid \alpha \leftarrow \text{ é uma instância fechada} \\ &\quad \text{duma cláusula de } P\} \end{aligned}$$

Exemplo 4.17. *Seja P:*

$$\begin{aligned} &\mathsf{S(f(x))} \leftarrow \mathsf{P(x,y)}, \mathsf{Q(y)} \\ &\mathsf{P(a, g(a))} \leftarrow \\ &\mathsf{P(b, g(b))} \leftarrow \\ &\mathsf{P(g(x), x)} \leftarrow \mathsf{Q(x)} \\ &\mathsf{Q(a)} \leftarrow \\ &\mathsf{Q(c)} \leftarrow \end{aligned}$$

e $I = \{\mathsf{P(a,b)}, \mathsf{Q(b)}\}$

$$T_P(I) = \{\mathsf{P(g(b),b)}, \mathsf{S(f(a))}, \mathsf{P(a,g(a))}, \mathsf{P(b,g(b))}, \mathsf{Q(a)}, \mathsf{Q(c)}\}$$

Exemplo 4.18. *Para P do exemplo 4.17:*

$$\begin{aligned}
T_P \uparrow 0 &= \emptyset \\
T_P \uparrow 1 &= \{\mathsf{Q(a)}, \mathsf{Q(c)}, \mathsf{P(a, g(a))}, \mathsf{P(b, g(b))}\} \\
T_P \uparrow 2 &= \{\mathsf{P(g(a), a)}, \mathsf{P(g(c), c)}, \mathsf{P(b, g(b))}, \mathsf{P(a, g(a))}, \mathsf{Q(a)}, \mathsf{Q(c)}\} \\
T_P \uparrow 3 &= \{\mathsf{P(g(a), a)}, \mathsf{P(g(c), c)}, \mathsf{P(b, g(b))}, \mathsf{P(a, g(a))}, \mathsf{Q(a)}, \mathsf{Q(c)}, \\
&\quad \mathsf{S(f(g(a)))}, \mathsf{S(f(g(c)))}\} \\
T_P \uparrow 4 &= \{\mathsf{P(g(a), a)}, \mathsf{P(g(c), c)}, \mathsf{P(b, g(b))}, \mathsf{P(a, g(a))}, \mathsf{Q(a)}, \mathsf{Q(c)}, \\
&\quad \mathsf{S(f(g(a)))}, \mathsf{S(f(g(c)))}\} \\
&\vdots \\
T_P \uparrow \omega &= \{\mathsf{P(g(a), a)}, \mathsf{P(g(c), c)}, \mathsf{P(b, g(b))}, \mathsf{P(a, g(a))}, \mathsf{Q(a)}, \mathsf{Q(c)}, \\
&\quad \mathsf{S(f(g(a)))}, \mathsf{S(f(g(c)))}\}
\end{aligned}$$

Exemplo 4.19. *Seja P*

$$\begin{aligned}
&\mathsf{Double(0, 0)} \leftarrow \\
&\mathsf{Double(s(x), s(s(y)))} \leftarrow \mathsf{Double(x, y)}
\end{aligned}$$

$$\begin{aligned}
T_P \uparrow 0 &= \emptyset \\
T_P \uparrow 1 &= \{\mathsf{Double(0, 0)}\} \\
T_P \uparrow 2 &= \{\mathsf{Double(0, 0)}, \mathsf{Double(s(0), s(s(0)))}\} \\
T_P \uparrow 3 &= \{\mathsf{Double(0, 0)}, \mathsf{Double(s(0), s(s(0)))}, \\
&\quad \mathsf{Double(s(s(0)), s(s(s(s(0)))))}\} \\
&\vdots \\
T_P \uparrow \omega &= \{\mathsf{Double}(\underbrace{s(\ldots s}_{\mathsf{n}}(0)\ldots), \underbrace{s(\ldots s}_{\mathsf{2n}}(0))\ldots)) \mid n \in \mathbb{N}\}
\end{aligned}$$

Proposição 4.6.

1. *$T_P \uparrow k \subseteq T_P \uparrow (k+1)$, para $k \geq 0$*
2. *$T_P(T_P \uparrow \omega) = T_P \uparrow \omega$, isto é, $= T_P \uparrow \omega$ é um ponto fixo de T_P*

Exercício 4.9. *Mostra a Proposição 4.6.* $\diamond$

A proposição seguinte caracteriza os modelos de Herbrand de um programa definido P:

Proposição 4.7. *Seja P uma programa definido e $\mathcal{A}$ uma estrutura de Herbrand de $\mathcal{L}_P$. $\mathcal{A}$ é um modelo de P (i.e $\mathcal{A} \models P$) se e só se $T_P(I_\mathcal{A}) \subseteq I_\mathcal{A}$.*

Demonstração. $\mathcal{A} \models P$ sse para cada instância fechada $\alpha \leftarrow \beta_1, \ldots, \beta_n$ de qualquer cláusula de P, se $\mathcal{A} \models \beta_1$, $\ldots$, $\mathcal{A} \models \beta_n sa$, então $\mathcal{A} \models \alpha$. Mas então, por definição de $I_{\mathcal{A}}$, se $\beta_1, \ldots \beta_n \in I_{\mathcal{A}}$ então $\alpha \in I_{\mathcal{A}}$. Isto é, se $\alpha \in T_P(I_{\mathcal{A}})$ então $\alpha \in I_{\mathcal{A}}$. □

Em particular, para qualquer programa P, a estrutura de Herbrand $\mathcal{A}$ tal que $I_{\mathcal{A}} = B_P$ é um modelo de P. (porquê?)

Teorema 4.5. *Seja P um programa definido, $M_P = T_P \uparrow \omega$*

Demonstração. $\subseteq$: A estrutura de Herbrand $\mathcal{A}$ tal que $I_{\mathcal{A}} = T_P \uparrow \omega$ é um modelo de P, porque $T_P(T_P \uparrow \omega) = T_P \uparrow \omega$. Então $M_P \subseteq T_P \uparrow \omega$.
$\supseteq$: mostra-se por indução sobre n que se $\alpha \in T_P \uparrow n$ então $P \models \alpha$. (mostra!) Então $T \uparrow n \subseteq M_P$, para todo $n \in \mathbb{N}$, logo $T \uparrow \omega \subseteq M_P$. □

Exemplo 4.20. *Para o programa P do exemplo 4.19,*

$$M_P = \{\mathsf{Double}(\underbrace{s(\ldots s}_{\mathsf{n}}(0)\ldots), \underbrace{s(\ldots s}_{\mathsf{2n}}(0))\ldots)) \mid n \in \mathbb{N}\}$$

Exercício 4.10. *Para os programas definidos do exercício 4.7 determina o $T \uparrow \omega (= M_P)$.* ⋄

4.4 Resposta correcta

Definição 4.16. *Seja P um programa definido e G um objectivo. Uma* resposta *para $P \cup \{G\}$ é uma substituição para variáveis que ocorrem em G.*

Para P do exemplo 4.19, $P \cup \{\leftarrow \mathit{Double}(x, s(y)\}$, $[s(0)/x]$ é uma resposta.

Definição 4.17. *Seja P um programa definido e G um objectivo $\leftarrow \beta_1, \ldots, \beta_k$, $k \geq 1$ e θ uma resposta para $P \cup \{G\}$. θ é uma* resposta correcta *para $P \cup \{G\}$ sse*

$$P \models \forall x_1 \ldots \forall x_s((\beta_1 \wedge \ldots \wedge \beta_k)\theta)$$

onde $x_1, \ldots, x_s$ são as variáveis que ocorrem em $(\beta_1 \wedge \ldots \wedge \beta_k)\theta$.

Então, θ é uma resposta correcta sse $P \cup \{\neg\forall x_1 \ldots \forall x_s((\beta_1 \wedge \ldots \wedge \beta_k)\theta)\}$ é não satisfazível.
Para o exemplo acima:
$[s(0)/x, s(0)/y]$ é uma resposta correcta.
$[s(z)/x, s(z)/y]$ não é uma resposta correcta.
Se $P \cup \{G\}$ não é satisfazível então $P \models \exists(\beta_1 \wedge \ldots \wedge \beta_n)$. Isto quer dizer que existe uma substituição fechada σ tal que $P \models (\beta_1 \wedge \ldots \wedge \beta_n)\sigma$. Suponhamos que podemos escrever

$\sigma = \theta\gamma$ tal que em que $P \models (\beta_1 \wedge \ldots \wedge \beta_n)\theta\gamma$ para qualquer substituição fechada γ. Então temos que $P \models \forall(\beta_1 \wedge \ldots \wedge \beta_k)\theta$, i.e., θ é uma resposta correcta para G.
Podemos relacionar as respostas correctas com o modelo mínimo de Herbrand de um programa definido:

Teorema 4.6. *Seja P um programa definido, G um objectivo $\leftarrow \beta_1, \ldots, \beta_k$, $k \geq 1$ e θ uma resposta para $P \cup \{G\}$, tal que $(\beta_1 \wedge \ldots \wedge \beta_k)\theta$ é fechada. Então são equivalentes:*

1. *θ é uma resposta correcta*
2. *se $\mathcal{A}$ é um modelo de Herbrand para P, então $\mathcal{A} \models (\beta_1 \wedge \ldots \wedge \beta_k)\theta$*
3. *$\mathcal{A}_P \models (\beta_1 \wedge \ldots \wedge \beta_k)\theta$*

Demonstração. Obviamente 1⇒2 e 2⇒3. Basta ver que 3⇒1:

$\mathcal{A}_P \models (\beta_1 \wedge \ldots \wedge \beta_k)\theta$ ⇒ $\mathcal{A} \models (\beta_1 \wedge \ldots \wedge \beta_k)\theta$, para qualquer $\mathcal{A}$ modelo de Herbrand para P

⇒ $\mathcal{A} \not\models \neg(\beta_1 \wedge \ldots \wedge \beta_k)\theta$, para qualquer $\mathcal{A}$ modelo de Herbrand para P

⇒ $P \cup \{\neg(\beta_1 \wedge \ldots \wedge \beta_k)\theta\}$ não tem modelos de Herbrand (e como $\neg(\beta_1 \wedge \ldots \wedge \beta_k)\theta$ é uma cláusula!)

⇒ $P \cup \{\neg(\beta_1 \wedge \ldots \wedge \beta_k)\theta\}$ não é satisfazível.

□

4.5 Resolução-SLD

Vamos definir um sistema dedutivo que permita determinar se $P \cup \{G\}$ não é satisfazível e, nesse caso, calcular uma resposta.

Definição 4.18. *Seja G um objectivo $\leftarrow \alpha_1, \ldots, \alpha_m, \ldots, \alpha_k$ e $\alpha \leftarrow \beta_1, \ldots, \beta_q$, uma cláusula C. G' é derivado por resolução de G e C, com unificador mais geral θ se:*

1. *α_m é o átomo seleccionado em G*
2. *θ é um* umg *de α_m e α*
3. *G' é $\leftarrow (\alpha_1, \ldots, \beta_1, \ldots, \beta_q, \ldots, \alpha_k)\theta$*

Isto é,

$$\frac{\leftarrow \alpha_1, \ldots, \alpha_m, \ldots, \alpha_k \quad \alpha \leftarrow \beta_1, \ldots, \beta_q}{\leftarrow (\alpha_1, \ldots, \beta_1, \ldots, \beta_q, \ldots, \alpha_k)\theta}$$

G' diz-se a a resolvente de G e C.

Definição 4.19. *Seja P um programa definido e G um objectivo. Uma* derivação SLD *de $P \cup \{G\}$ é constituída por:*

- *uma sequência (finita ou infinita) $G_0 = G, G_1, G_2, \ldots$ de objectivos*
- *uma sequência $C_1, C_2, \ldots$ de variantes de cláusulas em P, de modo a que nenhuma variável de C_i tenha ocorrido antes na derivação*
- *uma sequência de substituições θ_1, θ_2, ... tal que cada G_{i+1} foi derivado (por resolução) de G_i e C_i com* umg θ_{i+1}

A sigla SLD *significa* Resolução **L**inear com função de **S**eleção para Cláusulas **D**efinidas.

$$\begin{array}{cc} G_0 = G & C_1, \theta_1 \\ \downarrow & \swarrow \\ G_1 & C_2, \theta_2 \\ \downarrow & \swarrow \\ G_2 & \\ \vdots & \\ G_{n-1} & C_n, \theta_n \\ \downarrow & \swarrow \\ G_n & \\ \downarrow & \\ \vdots & \end{array}$$

Definição 4.20. *Uma* refutação-*SLD de $P \cup \{G\}$ é uma derivação* SLD *finita de $P \cup \{G\}$ cujo último objectivo é a cláusula vazia (ϵ). Se $G_n = \epsilon$ dizemos que a refutação tem comprimento n.*

Exemplo 4.21. *Seja P:* $\mathsf{Max(x,y,x)} \leftarrow \mathsf{Less\_eq(y,x)}$
$\mathsf{Max(x,y,y)} \leftarrow \mathsf{Less\_eq(x,y)}$
$\mathsf{Less\_eq(0,x)} \leftarrow$
$\mathsf{Less\_eq(s(x),s(y))} \leftarrow \mathsf{Less\_eq(x,y)}$

e $G : \leftarrow \mathsf{Max(s(0),x,s(s(0)))}$

Uma refutação para $P \cup \{G\}$ de comprimento $n = 3$ é:

$\leftarrow \mathsf{Max(s(0),x,s(s(0)))}$	$(\mathsf{Max(x_1,y_1,y_1)} \leftarrow \mathsf{Less\_eq(x_1,y_1)}$
	$\theta_1 = [\mathsf{s(0)/x_1, s(s(0))/y_1, s(s(0))/x}])$
$\leftarrow \mathsf{Less\_eq(s(0),s(s(0)))}$	$(\mathsf{Less\_eq(s(x_2),s(y_2))} \leftarrow \mathsf{Less\_eq(x_2,y_2)})$
	$\theta_2 = [\mathsf{0/x_2, s(0)/y_2}])$
$\leftarrow \mathsf{Less\_eq(0,s(0))}$	$(\mathsf{Less\_eq(0,x_3)} \leftarrow, \theta_3 = [\mathsf{s(0)/x_3}])$
ϵ	

Definição 4.21. *Seja P um programa definido e G um objectivo. Se $\theta_1, \ldots, \theta_n$ for a sequência de* umg*s usada numa refutação-*SLD *de $P \cup \{G\}$, a* resposta calculada *para $P \cup \{G\}$ é a restrição da substituição composta $\theta_1 \cdots \theta_n$ às variáveis de G.*

Exemplo 4.22. *Para a refutação do exemplo 4.21, a resposta calculada é a restrição de*

$$\begin{aligned}\theta_1 \cdots \theta_3 &= [\mathsf{s(0)/x_1, s(s(0))/y_1, s(s(0))/x}][\mathsf{0/x_2, s(0)/y_2}][\mathsf{s(0)/x_3}] \\ &= [\mathsf{s(0)/x_1, s(s(0))/y_1, s(s(0))/x, 0/x_2, s(0)/y_2, s(0)/x_3}]\end{aligned}$$

à variável x, i.e., $\theta = [\mathsf{s(s(0))/x}]$

Exercício 4.11. *Sendo $G = \leftarrow \mathsf{Max(x, y, y), Max(y, s(0), s(0))}$, determina uma resposta calculada para $P \cup \{G\}$.* $\diamond$

4.5.1 Integridade da resolução *SLD*

Teorema 4.7. ***(Integridade da resolução* SLD*)*** *Seja P um programa definido e G um objectivo. Toda a resposta calculada para $P \cup \{G\}$ é uma resposta correcta para $P \cup \{G\}$.*

Demonstração. Seja G um objectivo $\leftarrow \alpha_1, \ldots, \alpha_k$ e seja uma refutação de comprimento n para $P \cup \{G\}$, constituída por $G_0 = G, G_1, \ldots, G_n$ objectivos, $C_1, \ldots, C_n$ variantes de cláusulas de P e $\theta_1, \ldots, \theta_n$ *umgs.* Mostramos por indução sobre n que $P \models \forall((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta_1 \cdots \theta_n)$:

Base. Para $\mathbf{n{=}1}$, tem-se $G = \leftarrow \alpha_1$, $C = \alpha \leftarrow$ e $\alpha_1\theta_1 = \alpha\theta_1$. E, como C é uma variante duma cláusula de P, $P \models \forall(\alpha_1\theta_1)$

Indução. Suponhamos que o resultado é válido para refutações de comprimento menor que n. Seja $C_1 = \alpha \leftarrow \beta_1, \ldots, \beta_q$ $(q \geq 0)$ e seja α_m o átomo seleccionado em G. Por hipótese indução,

$$P \models \forall((\alpha_1 \wedge \ldots \wedge \alpha_{m-1} \wedge \beta_1 \wedge \ldots \wedge \beta_q \wedge \alpha_{m+1} \wedge \ldots \wedge \alpha_k)\theta_1 \ldots \theta_n)$$

Para $q > 0$, resulta em particular que $P \models \forall((\beta_1 \wedge \ldots \wedge \beta_q)\theta_1 \cdots \theta_n)$.

Mas $\forall(\alpha_m\theta_1 \cdots \theta_n)$ é igual a $\forall(\alpha\theta_1 \cdots \theta_n)$, donde $P \models \forall(\alpha_m\theta_1 \cdots \theta_n)$, logo

$$P \models \forall((\alpha_1 \wedge \ldots \wedge \alpha_k)\theta_1 \cdots \theta_n)$$

.

□

Corolário 4.3. *Seja P um programa definido e G um objectivo. Se existir uma refutação-*SLD *de $P \cup \{G\}$, então $P \cup \{G\}$ não é satisfazível.*

Demonstração. Seja G um objectivo $\leftarrow \alpha_1, \ldots, \alpha_k$ e θ uma resposta calculada para $P \cup \{G\}$. Pelo teorema 4.7, θ é uma resposta correcta para $P \cup \{G\}$, i.e., $P \models \forall(\alpha_1 \wedge \ldots \wedge \alpha_k)\theta$, logo $P \cup \{G\}$ não é satisfazível. □

Definição 4.22. *Dado um programa definido P o* conjunto sucesso *de P é*

$$\mathsf{Suc}(P) = \{\alpha \in B_P \mid \textit{existe uma refutação de } P \cup \{\leftarrow \alpha\}\}.$$

Podemos considerar $\mathsf{Suc}(P)$ uma *versão sintáctica* de M_P...na realidade:

Lema 4.1. *Dado um programa definido P, $\mathsf{Suc}(P) \subseteq M_P$.*

Demonstração. Se $\alpha \in \mathsf{Suc}(P)$, então existe uma refutação *SLD* de $P \cup \{\leftarrow \alpha\}$, donde pela integridade, $P \models \alpha$. Logo, $\alpha \in M_P$ □

Vamos ver que também $M_P \subseteq \mathsf{Suc}(P)$.

Definição 4.23. *Uma refutação-*SLD *não restringida é uma refutação-*SLD *em que se usam substituições $\theta_1, \theta_2, \ldots$ que são unificadores, mas não necessariamente unificadores mais gerais.*

Vamos ver que se há uma refutação-*SLD* não restringida então também existe uma refutação-*SLD*...

Lema 4.2. *Seja P um programa definido e G um objectivo tal que existe uma refutação-*SLD *não restringida de $P \cup \{G\}$ de comprimento n com unificadores $\theta_1, \ldots, \theta_n$. Então existe uma substituição γ e uma refutação-*SLD *de $P \cup \{G\}$ de comprimento n e com* umgs *$\theta'_1 \ldots \theta'_n$ tal que $\theta_1 \ldots \theta_n = \theta'_1 \ldots \theta'_n\gamma$*

Demonstração. Por indução sobre n.

Base. Para $n = 1$. Tem-se que $G_0 = \leftarrow \alpha$, $G_1 = \epsilon$, $C_1 = \alpha_1$ e $\alpha_1\theta_1 = \alpha\theta_1$. Mas então existe um unificador mais geral θ'_1 de $\{\alpha, \alpha_1\}$, donde existe γ tal que $\theta_1 = \theta'_1\gamma$. E então tem-se uma refutação de comprimento 1 e *umg* θ'_1.

Indução. Suponhamos que o resultado é válido para refutações de comprimento $< n$. Seja $G_0 = G, G_1, G_2, \ldots, G_n = \epsilon$ uma refutação não restringida, com variantes de cláusulas $C_1, \ldots, C_n$ e unificadores $\theta_1, \ldots, \theta_n$. Então existe um *umg* θ'_1 para o átomo seleccionado em G e a cabeça de C_1, tal que $\theta_1 = \theta'_1\rho$ para alguma substituição ρ. Logo existe uma refutação não restringida para $P \cup \{G\}$, $G_0 = G, G'_1, G_2, \ldots, G_n = \epsilon$, com com variantes de cláusulas $C_1, \ldots, C_n$ e unificadores $\theta'_1, \rho\theta_2, \ldots, \theta_n$ tal que $G_1 = G'_1\rho$. Por hipótese de indução existe para $P \cup \{G'_1\}$, uma refutação-*SLD* $G'_1, G_2, \ldots, G_n = \epsilon$ com *umgs* $\theta'_2 \cdots \theta'_n$ tal que $\rho\theta_2 \cdots \theta_n = \theta'_2 \cdots \theta'_n\gamma$, para alguma substituição γ. Logo existe $P \cup \{G\}$, uma refutação-*SLD* $G_0 = G, G'_1, G_2, \ldots, G_n = \epsilon$ com *umgs* $\theta', \theta'_2 \ldots \theta'_n$ tal que $\theta_1 \cdots \theta_n = \theta'\rho\theta_2 \cdots \theta_n = \theta'_1\theta'_2 \cdots \theta'_n\gamma$.

□

Lema 4.3. *Seja P um programa definido, G um objectivo e θ uma substituição. Suponhamos que existe uma refutação-*SLD *para $P \cup \{G\theta\}$ de comprimento n e com umgs $\theta_1, \ldots, \theta_n$. Então existe uma substituição γ e uma refutação-*SLD *para $P \cup \{G\}$ de comprimento n e com umgs $\theta'_1, \ldots, \theta'_n$ tal que $\theta\theta_1 \cdots \theta_n = \theta'_1 \cdots \theta'_n\gamma$.*

Demonstração. Suponhamos que, no primeiro passo, C_1 é uma variante duma cláusula de P tal que $C_1\theta = C_1$. Então $\theta\theta_1$ é um unificador da cabeça de C_1 e do átomo em G que corresponde ao átomo seleccionado em $G\theta$. A resolvente de G e C_1 com unificador $\theta\theta_1$ é precisamente G_1. Então obtemos uma refutação não restringida de $P \cup \{G\}$ que é idêntica à refutação original de $P \cup \{G\theta\}$ a menos do primeiro passo, pois começa com G e utiliza $\theta\theta_1$. Basta agora aplicar o lema 4.2. □

Teorema 4.8. *Seja P um programa definido, $\mathsf{Suc}(P) = M_P$.*

Demonstração. $\mathsf{Suc}(P) \subseteq M_P$: pelo lema 4.1.
$M_P \subseteq \mathsf{Suc}(P)$: Seja $\alpha \in M_P$. Então $\alpha \in T_P \uparrow \omega$. Logo, existe $n \in \mathbb{N}$ tal que $\alpha \in T_P \uparrow n$. Mostramos por indução sobre n que existe uma refutação de $P \cup \{\leftarrow \alpha\}$, i.e., $\alpha \in \mathsf{Suc}(P)$.

Base. Para $n = 0$ é trivial.

Indução. Suponhamos que o resultado se verifica para $\beta \in T_P \uparrow i$, $i < n$, e seja $\alpha \in T_P \uparrow n$. Então existe uma instância fechada $\alpha_1\theta \leftarrow \beta_1\theta, \ldots, \beta_k\theta$ de uma cláusula de P, tal que $\beta_1\theta, \ldots, \beta_k\theta \in T_P \uparrow (n-1)$ e $\alpha = \alpha_1\theta$. Por hipótese de indução, existem refutações de $P \cup \{\leftarrow \beta_j\theta\}$, para $j = 1, \ldots, k$ e como cada $\beta_j\theta$ é um átomo fechado é possível combinar essas refutações para uma refutação de $P \cup \{\leftarrow (\beta_1, \ldots, \beta_n)\theta\}$. Logo, existe uma refutação-*SLD* não restringida para $P \cup \{\leftarrow \alpha\}$ e pelo lema 4.2 uma refutação-*SLD*.

□

4.5.2 Completude da resolução-*SLD*

Teorema 4.9. *Seja P um programa definido e G um objectivo tal que $P \cup \{G\}$ não é satisfazível. Então existe uma refutação-*SLD *de $P \cup \{G\}$.*

Demonstração. Seja G o objectivo $\leftarrow \beta_1, \ldots, \beta_k$. Como $P \cup \{G\}$ não é satisfazível, então $P \vdash \exists(\beta_1 \wedge \ldots \wedge \beta_k)$. E também, existe $P \vdash (\beta_1 \wedge \ldots \wedge \beta_k)\theta$, e tal que $\beta_i\theta$ são átomos fechados. Logo $\{\beta_1\theta, \ldots, \beta_k\theta\} \subseteq M_P$ e pelo Teorema 4.8 existe uma refutação de $P \cup \{\leftarrow \beta_i\theta\}$, para $i = 1, \ldots, k$. Podemos então obter uma refutação para $P \cup \{G\theta\}$ e aplicar o lema 4.3. □

Lema 4.4. *Seja P um programa definido e α um átomo tal que $P \models \forall x_1 \ldots \forall x_s \alpha$, onde $x_1, \ldots, x_s$ são as variáveis em α. Então existe uma refutação-*SLD *de $P \cup \{\leftarrow \alpha\}$ cuja resposta é uma substituição identidade.*

Demonstração. Consideremos temporariamente, P como um conjunto de cláusulas da linguagem alargada $\mathcal{L}' = \mathcal{L}_P \cup \{a_1, \ldots, a_s\}$ onde a_i são constantes novas e distintas. Então, $P \models \alpha\theta$ para $\theta = [a_1/x_1, \ldots, a_s/x_s]$. Mas $\alpha\theta$ é fechado, pelo Teorema 4.8, existe uma refutação para $P \cup \{\leftarrow \alpha\theta\}$. Substituindo nessa refutação as ocorrências de a_i por x_i, para $i = 1, \ldots, s$, obtemos uma refutação para $P \cup \{\leftarrow \alpha\}$ cuja resposta é uma substituição identidade. □

Não vamos ter exactamente inverso do teorema da integridade, dado que as respostas calculadas são todas com *umg*s.

Teorema 4.10. ***(Completude da resolução* SLD*)*** *Seja P um programa definido e G um objectivo. Para toda a resposta correcta θ para $P \cup \{G\}$, existe uma resposta calculada σ de $P \cup \{G\}$ e uma substituição γ, tal que $\theta = \sigma\gamma$*

Demonstração. Seja G o objectivo $\leftarrow \beta_1, \ldots, \beta_k$. Como θ é uma resposta correcta para $P \cup \{G\}$, tem-se que $P \models \forall x_1 \ldots \forall x_s (\beta_1 \wedge \ldots \wedge \beta_k)\theta$
Pelo lema 4.4, para $i = 1, \ldots, k$ existe uma refutação de $P \cup \{\leftarrow \beta_i\theta\}$ que tem uma substituição identidade, como resposta calculada. Combinando estas refutações é possível obter uma refutação de $P \cup \{G\theta\}$, cuja resposta calculada é a identidade.
Seja então $\theta_1, \ldots, \theta_n$ a sequência de *umg*s utilizada na refutação de $P \cup \{G\theta\}$. Então $G\theta\theta_1 \cdots \theta_n = G\theta$. Aplicando o lema 4.3, existe uma refutação de $P \cup \{G\}$ com *umg*s $\theta'_1, \ldots, \theta'_n$ tal que $\theta\theta_1 \cdots \theta_n = \theta'_1 \cdots \theta'_n\gamma'$, para algum γ'. Tomando as restrições às variáveis de G, σ de $\theta'_1 \cdots \theta'_n$ e γ de γ', vem $\theta = \sigma\gamma$. □

Leituras suplementares [Llo87] [BA01] (Cap. 7,8)

Apêndice A

Dedução natural para a lógica proposicional

	Introdução		**Eliminação**
$\wedge$	$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\wedge\mathbf{I}$		$\dfrac{\varphi \wedge \psi}{\varphi}\wedge\mathbf{E}_1$ $\quad$ $\dfrac{\varphi \wedge \psi}{\psi}\wedge\mathbf{E}_2$
$\vee$	$\dfrac{\varphi}{\varphi \vee \psi}\vee\mathbf{I}_1$	$\dfrac{\psi}{\varphi \vee \psi}\vee\mathbf{I}_2$	$\dfrac{\varphi \vee \psi \qquad \begin{array}{c}[\varphi]\\ \vdots \\ \gamma\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots \\ \gamma\end{array}}{\gamma}\vee\mathbf{E}$
$\neg$	$\begin{array}{c}[\varphi]\\ \vdots \\ \mathbf{F}\end{array}$ $\dfrac{}{\neg\varphi}\neg\mathbf{I}$		$\dfrac{\neg\neg\varphi}{\varphi}\neg\mathbf{E}$
$\mathbf{F}$	$\begin{array}{c}\varphi\\ \vdots \\ \neg\varphi\end{array}$ $\dfrac{}{\mathbf{F}}\mathbf{F}\mathbf{I}(*)$		$\dfrac{\mathbf{F}}{\varphi}\mathbf{F}\mathbf{E}$
$\rightarrow$	$\begin{array}{c}[\varphi]\\ \vdots \\ \psi\end{array}$ $\dfrac{}{\varphi\rightarrow\psi}\rightarrow\mathbf{I}$		$\dfrac{\varphi \quad \varphi\rightarrow\psi}{\psi}\rightarrow\mathbf{E}$

Regra da Repetição

$$\frac{\varphi}{\varphi}\mathbf{R}$$

Algumas regras derivadas:

$$\frac{\varphi \to \psi \qquad \neg\psi}{\neg\varphi}\mathbf{MT}$$

$$\frac{\varphi}{\neg\neg\varphi}\neg\neg\mathbf{I}$$

$$\begin{array}{c} [\neg\varphi] \\ \vdots \\ \mathbf{F} \end{array}\over{\varphi}\mathbf{RA}$$

$$\frac{}{\varphi \vee \neg\varphi}\mathbf{TE}$$

(Fitch)

	Introdução	**Eliminação**
$\wedge$	φ $\vdots$ ψ $\vdots$ $\varphi \wedge \psi$ $\wedge$I	$\vdots$ $\varphi \wedge \psi$ $\vdots$ φ $\wedge$E $\vdots$ $\varphi \wedge \psi$ $\vdots$ ψ $\wedge$E
$\vee$	$\vdots$ φ $\vdots$ $\varphi \vee \psi$ $\vee$I $\vdots$ φ $\vdots$ $\psi \vee \varphi$ $\vee$I	$\varphi \vee \psi$ $\vdots$ φ / $\vdots$ / γ ψ / $\vdots$ / γ γ $\vee$E
$\neg$	φ / $\vdots$ / **F** $\neg\varphi$ $\neg$I	$\vdots$ $\neg\neg\varphi$ $\vdots$ φ $\neg$E
F	φ $\vdots$ $\neg\varphi$ $\vdots$ **F** **FI**	$\vdots$ **F** $\vdots$ φ **FE**
$\rightarrow$	φ / $\vdots$ / ψ $\varphi \rightarrow \psi$ $\rightarrow$I	φ $\vdots$ $\varphi \rightarrow \psi$ $\vdots$ ψ $\rightarrow$E

Apêndice B

Dedução natural para a lógica de 1ª ordem

	Introdução	**Eliminação**
$\wedge$	$\dfrac{\varphi \quad \psi}{\varphi \wedge \psi}\wedge\mathbf{I}$	$\dfrac{\varphi \wedge \psi}{\varphi}\wedge\mathbf{E}_1$ $\quad$ $\dfrac{\varphi \wedge \psi}{\psi}\wedge\mathbf{E}_2$
$\vee$	$\dfrac{\varphi}{\varphi \vee \psi}\vee\mathbf{I}_1$ $\quad$ $\dfrac{\psi}{\varphi \vee \psi}\vee\mathbf{I}_2$	$\dfrac{\varphi \vee \psi \qquad \begin{array}{c}[\varphi]\\ \vdots\\ \gamma\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots\\ \gamma\end{array}}{\gamma}\vee\mathbf{E}$
$\neg$	$\dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \mathbf{F}\end{array}}{\neg\varphi}\neg\mathbf{I}$	$\dfrac{\neg\neg\varphi}{\varphi}\neg\mathbf{E}$
$\mathbf{F}$	$\dfrac{\begin{array}{c}\varphi\\ \vdots\\ \neg\varphi\end{array}}{\mathbf{F}}\mathbf{F}\mathbf{I}(*)$	$\dfrac{\mathbf{F}}{\varphi}\mathbf{F}\mathbf{E}$
$\rightarrow$	$\dfrac{\begin{array}{c}[\varphi]\\ \vdots\\ \psi\end{array}}{\varphi\rightarrow\psi}\rightarrow\mathbf{I}$	$\dfrac{\varphi \quad \varphi\rightarrow\psi}{\psi}\rightarrow\mathbf{E}$
$=$	$\dfrac{}{t=t}=\mathbf{I}$	$\dfrac{t_1=t_2 \quad \varphi[t_1/x]}{\varphi[t_2/x]}=\mathbf{E}$ e x é substituível por t_1 e por t_2 em φ
$\forall$	$\dfrac{\begin{array}{c}[v]\\ \vdots\\ \varphi[v/x]\end{array}}{\forall x\,\varphi}\forall\mathbf{I}$ onde v é uma variável nova	$\dfrac{\forall x\,\varphi}{\varphi[t/x]}\forall\mathbf{E}$ onde x é substituível por t em φ

Regra da Repetição

$$\frac{\varphi}{\varphi}\,\mathbf{R}$$

Algumas regras derivadas:

$$\frac{\varphi \to \psi \qquad \neg\psi}{\neg\varphi}\,\mathbf{MT} \qquad \frac{\varphi}{\neg\neg\varphi}\,\neg\neg\mathbf{I}$$

$$\frac{\begin{array}{c}[\neg\varphi]\\ \vdots \\ \mathbf{F}\end{array}}{\varphi}\,\mathbf{RA} \qquad \frac{}{\varphi \vee \neg\varphi}\,\mathbf{TE}$$

	Introdução	**Eliminação**
$\wedge$	φ $\vdots$ ψ $\vdots$ $\varphi \wedge \psi$ $\wedge$I	$\vdots$ $\varphi \wedge \psi$ $\vdots$ φ $\wedge$E $\vdots$ $\varphi \wedge \psi$ $\vdots$ ψ $\wedge$E
$\vee$	$\vdots$ φ $\vdots$ $\varphi \vee \psi$ $\vee$I $\vdots$ φ $\vdots$ $\psi \vee \varphi$ $\vee$I	$\varphi \vee \psi$ $\vdots$ φ / $\vdots$ / γ ψ / $\vdots$ / γ γ $\vee$E
$\neg$	φ / $\vdots$ / **F** $\neg\varphi$ $\neg$I	$\vdots$ $\neg\neg\varphi$ $\vdots$ φ $\neg$E
F	φ $\vdots$ $\neg\varphi$ $\vdots$ **F** **FI**	$\vdots$ **F** $\vdots$ φ **FE**
$\rightarrow$	φ / $\vdots$ / ψ $\varphi \rightarrow \psi$ $\rightarrow$I	φ $\vdots$ $\varphi \rightarrow \psi$ $\vdots$ ψ $\rightarrow$E

	Introdução	**Eliminação**
$=$	$\begin{array}{l} \vdots \\ t = t \qquad =\mathbf{I} \end{array}$	$\begin{array}{l} t_1 = t_2 \\ \vdots \\ \varphi[t_1/x] \\ \vdots \\ \varphi[t_2/x] \qquad =\mathbf{E} \end{array}$ e x é substituível por t_1 e por t_2 em φ
$\forall$	$\begin{array}{l} v \mid \vdots \\ \quad \mid \varphi[v/x] \\ \forall x\, \varphi \qquad \forall\mathbf{I} \end{array}$ onde v é uma variável nova (não ocorre antes)	$\begin{array}{l} \forall x\, \varphi \\ \vdots \\ \varphi[t/x] \qquad \forall\mathbf{E} \end{array}$ onde x é substituível por t em φ
$\exists$	$\begin{array}{l} \varphi[t/x] \\ \vdots \\ \exists x\, \varphi \qquad \exists\mathbf{I} \end{array}$ onde x é substituível por t em φ	$\begin{array}{l} \exists x\, \varphi \\ v \mid \varphi[v/x] \\ \quad \mid \vdots \\ \quad \mid \psi \\ \psi \qquad \exists\mathbf{E} \end{array}$ onde v é uma variável nova que não ocorre antes nem em ψ

Bibliografia

[AFPMdS11] José Bacelar Almeida, Maria João Frade, Jorge Sousa Pinto, and Simão Melo de Sousa. *Rigorous Software Development: An Introduction to Program Verification*. Springer, 2011.

[And86] Peter B. Andrews. *An Introduction to Mathematical Logic and Type Theory:To Truth Through Proof*. Academic Press, Orlando, Florida, 1986.

[BA01] Mordechai Ben-Ari. *Mathematical Logic for Computer Science*. SV, 2nd edition, 2001.

[BE00] Jon Barwise and John Etchmendy. *Language, Proof, and Logic*. CSLI, 2000.

[Bro00] Sabine Broda. Apontamentos de lógica computacional. Technical report, Departamento de Ciência de Computadores, FCUP, 2000.

[DLL62] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5(7):394–397, July 1962.

[DP60] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215, July 1960.

[Fit90] Melvin Fitting. *First-Order Logic and Automated Theorem Proving*. SV, 1990.

[GLM97] Jean Goubault-Larrecq and Ian Mackie. *Proof Theory and Automated Deduction*. Kluwer Academic Press, 1997.

[HR00] Michael Huth and Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. CUP, 2000. 430 DCCBIB.

[Koz97] Dexter C. Kozen. *Automata and Computability*. Undergraduate texts in computer science. Springer, 1997.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming*. SV, 1987.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[Smu95] Raymond M. Smullyan. *Fisrt-order Logic*. Dover Publications, 1995.

Índice