Capítulo 10

A Lógica de Primeira Ordem

A Lógica de Primeira Ordem: A necessidade de uma linguagem mais expressiva

O cálculo proposicional possui limitações com respeito a codificação de sentenças declarativas. De fato, o cálculo proposicional manipula de forma satisfatória componentes das sentenças como *não*, *e* , *ou*, *se . . . então*, mas certos aspectos lógicos que aparecem em linguagens naturais ou artificiais são muito mais ricos. Por exemplo, como expressar coisas do tipo: "*Existe ...*" e "*Para todo . . .*" na lógica proposicional?

Exemplo 7. *Considere a seguinte sentença declarativa:*

Todo estudante é mais jovem do que algum instrutor.

Na lógica proposicional podemos identificar esta sentença com uma variável proposicional p. No entanto, esta codificação não reflete os detalhes da estrutura lógica desta sentença. De que trata esta sentença?

- Ser um estudante.
- Ser um instrutor.
- Ser mais jovem do que alguém.

Para expressar estas propriedades utilizaremos predicados. *Por exemplo, podemos escrever estudante(ana) para denotar que Ana é uma estudante. Da mesma forma podemos escrever instrutor(marcos) para denotar que Marcos é um instrutor. Por fim, podemos escrever jovem(ana,marcos) para denotar que Ana é mais jovem do que Marcos. Nestes exemplos, estudante, instrutor e jovem são exemplos de predicados.*

Ainda precisamos codificar as noções de "*todo*" e "*algum*". Para isto introduziremos o conceito de variável. Variáveis serão denotadas por letras latinas minúsculas do final do alfabeto: u, v, w, x, y, z (possivelmente acrescidas de sub-índices $x_1, x_2, \ldots$). Variáveis devem ser pensadas como "lugares vazios" que podem ser preenchidos (ou instanciados) por elementos concretos, como joão, maria, etc. Utilizando variáveis podemos especificar o significado dos predicados estudante, instrutor e jovem de uma maneira mais formal:

estudante(x): x é um estudante.
instrutor(x): x é um instrutor.
jovem(x,y): x é mais jovem do que y.

Note que o nome das variáveis não é importante: estudante(x): x é um estudante. é equivalente a estudante(y): y é um estudante. Para que possamos finalmente expressar em detalhes a sentença apresentada no exemplo precisamos codificar o significado de **Todo** e **algum** em **Todo** estudante é mais jovem do que **algum** instrutor. Os quantificadores $\forall$ e $\exists$ fazem este trabalho:

$\forall$: significa *para todo*;

$\exists$: significa *existe*.

Os quantificadores $\forall$ e $\exists$ estão sempre ligados a alguma variável:

$\forall_x$: para todo x;

$\exists_x$: existe um x (ou existe algum x).

Agora podemos finalmente codificar a sentença:

Todo estudante é mais jovem do que algum instrutor.

da seguinte forma:

$$\forall_x(estudante(x) \rightarrow (\exists_y(instrutor(y) \wedge jovem(x, y))))$$

Note que predicados diferentes podem ter um número distinto de argumentos: os predicados estudante e instrutor admitem apenas um argumento e por isto são chamados de predicados unários, enquanto que o predicado jovem admite dois argumentos, e portanto é um predicado binário.

O número de argumentos de um predicado é chamado sua *aridade*. Assim, os predicados unários têm aridade 1, enquanto que os predicados binários têm aridade 2, etc. No cálculo de predicados são permitidos predicados com qualquer aridade finita.

Exemplo 8. *Considere a sentença:*

Nem todos os pássaros podem voar.

Escolhemos os seguintes predicados para expressar esta sentença:

passaro(x): x *é um pássaro.*
voar(x): x *pode voar.*

Esta sentença pode ser codificada da seguinte forma:

$$\neg(\forall_x(passaro(x) \rightarrow voar(x)))$$

Exemplo 9. *Uma outra maneira de expressar a mesma idéia da sentença anterior é dizer que:*

Existem alguns pássaros que não podem voar.

Esta última sentença pode ser codificada da seguinte maneira:

$$\exists_x(passaro(x) \wedge \neg voar(x))$$

Posteriormente veremos que as duas codificações dadas são semanticamente equivalentes. De fato, existem transformações que convertem uma na outra.

A lógica de primeira ordem: uma linguagem formal

O vocabulário da lógica de primeira ordem consiste de três conjuntos:

1. Um conjunto $\mathcal{P}$ de símbolos de predicado;
2. Um conjunto $\mathcal{F}$ de símbolos de função;
3. Um conjunto $\mathcal{C}$ de constantes.

onde cada símbolo de predicado e de função vem com sua aridade bem definida. Os predicados são casos especiais de função: enquanto as funções possuem contra-domínio qualquer, os predicados têm contra-domínio sempre igual a $\{V, F\}$. As constantes são funções de aridade 0.

Definição 19. *Termos são definidos da seguinte forma:*

1. *Qualquer variável é um termo;*
2. *Se $c \in \mathcal{F}$ é uma função de aridade 0 então c é um termo;*
3. *Se $t_1, \ldots, t_n$ são termos e $f \in \mathcal{F}$ é uma função de aridade $n > 0$ então $f(t_1, \ldots, t_n)$ é um termo.*
4. *Nada mais é termo.*

Em BNF (Backus Naur form) temos:

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

onde x percorre o conjunto de variáveis $\mathcal{V}$, c percorre os símbolos de função de aridade 0 de $\mathcal{F}$ e f percorre os elementos de aridade maior do que 0 de $\mathcal{F}$.

Exemplo 10. *Suponha que n, f e g são símbolos de função de aridade respetivamente igual a 0, 1 e 2. Então $g(f(n), n)$ e $f(g(n, f(n)))$ são termos, mas $g(n)$ e $f(f(n), n)$ não são termos por violarem as aridades dos símbolos.*

A escolha dos conjuntos $\mathcal{P}$ e $\mathcal{F}$ para símbolos de predicado e de função é definida a partir do que se pretende descrever.

Definição 20. *Definimos o conjunto de fórmulas sobre o conjunto $S = (\mathcal{F}, \mathcal{P})$ indutivamente da seguinte forma:*

1. *Se $p \in \mathcal{P}$ é um símbolo de predicado de aridade $n > 0$, e se $t_1, \ldots, t_n$ são termos sobre $\mathcal{F}$ então $p(t_1, \ldots, t_n)$ é uma fórmula.*
2. *Se ϕ é uma fórmula então $(\neg\phi)$ é também uma fórmula.*
3. *Se ϕ e ψ são fórmulas então $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$ e $(\phi \leftrightarrow \psi)$ são fórmulas.*
4. *Se ϕ é uma fórmula e x é uma variável então $(\forall_x \phi)$ e $(\exists_x \phi)$ também são fórmulas.*
5. *Nada mais é fórmula.*

Em BNF temos:

$$\phi ::= p(t_1,\ldots,t_n) \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \rightarrow \phi) \mid (\phi \leftrightarrow \psi) \mid (\forall_x \phi) \mid (\exists_x \phi)$$

onde p é um símbolo de predicado de aridade $n > 0$, t_i são termos sobre $\mathcal{F}$ e x é uma variável.

Convenção 1. *Adotaremos a seguinte prioridade de operadores:*

1. $\neg$, $\forall$, $\exists$;
2. $\wedge, \vee$;
3. $\rightarrow, \leftrightarrow$.

Exemplo 11. *Considere a seguinte sentença:*

Todo filho de meu pai é meu irmão.

Podemos codificar esta fórmula de pelo menos duas formas distintas:

1. *Representando a noção de "pai" como predicado: Neste caso escolhemos três predicados: filho, pai e irmao com os seguintes significados e aridades:*

$$\begin{aligned} filho(x,y)&: \quad x \text{ é filho de } y \\ pai(x,y)&: \quad x \text{ é pai de } y \\ irmao(x,y)&: \quad x \text{ é irmão de } y. \end{aligned}$$

Uma possível codificação para a sentença dada utilizando estes predicados é:

$$\forall_x \forall_y (pai(x, joao) \wedge filho(y, x) \rightarrow irmao(y, joao))$$

dizendo que: "para todo x e todo y, se x é o pai de joao e se y é um filho de x então y é um irmão de joao".

Representando a noção de "pai" como função, que chamaremos de f: Neste caso, $f(x)$ retorna o pai de x. Note que isto funciona apenas porque o pai de uma dado x é único e está sempre definido, e portanto f é realmente uma função. Uma possível codificação para esta sentença é dada por:

$$\forall_x (filho(x, f(joao)) \rightarrow irmao(x, joao))$$

significando que "para todo x, se x é um filho do pai de joao então x é um irmao de joao.

Esta codificação é menos complexa que a anterior porque envolve apenas um quantificador.

Especificações formais em geral exigem um domínio de conhecimento. Muitas vezes este conhecimento não está explicitado no domínio. Sendo assim, um especificador pode desconsiderar restrições importantes para um modelo ou implementação. Por exemplo, as codificações dadas no exemplo anterior podem parecer corretas, mas e se x for igual a joao? Se o domínio de relações de parentesco não é um conhecimento comum o especificador pode não notar que uma pessoa não pode ser irmão dela mesma.

Definição 21. *A* **abrangência** *de $\forall_x$ (respectivamente, $\exists_x$) em $\forall_x \phi$ (respectivamente, $\exists_x \phi$) é ϕ. Uma ocorrência de uma* **variável ligada** *numa fórmula ψ, é uma ocorrência de uma variável x, dentro do campo de abrangência de um quantificador $\forall_x$ ou $\exists_x$. Uma ocorrência de uma* **variável livre** *é uma ocorrência de uma variável x não ligada.*

Exemplo 12. *Na fórmula $\exists_x(p(f(x), y) \to q(x))$, as duas ocorrências da variável x são ligadas, enquanto a ocorrência da variável y é livre. Na fórmula $\exists_x p(f(x), y) \to q(x)$ a primeira ocorrência da variável x é ligada, no entanto a segunda é livre.*

Definição 22. *Dada uma variável x, um termo t e uma fórmula ϕ, definimos $\phi[x/t]$ como sendo a fórmula obtida após substituir cada ocorrência livre de x em ϕ por t.*

Exemplo 13. *Considere novamente a fórmula $\forall_x((p(x) \to q(x)) \wedge s(x, y))$, que chamaremos simplesmente de ϕ. Temos que $\phi[x/f(x, y)] = \phi$. De fato, todas as ocorrências de x em ϕ são ligadas, e portanto a substituição $[x/f(x, y)]$ não tem nenhum efeito sobre esta fórmula.*

Exemplo 14. *Agora considere a fórmula*

$$(\forall_x(p(x) \wedge q(x))) \to (\neg p(x) \vee q(y))$$

que chamaremos simplesmente de ψ. Neste caso temos uma ocorrência livre de x e, portanto $\psi[x/f(x, y)]$ é igual a

$$(\forall_x(p(x) \wedge q(x))) \to (\neg p(f(x, y)) \vee q(y))$$

As substituições podem produzir efeitos colaterais indesejados:

Considere o termo $f(x, y)$ e a fórmula $\forall_y(p(x, y))$. Então $(\forall_y(p(x, y)))[x/f(x, y)]$ resulta na fórmula $(\forall_y(p(f(x, y), y)))$ se fizermos uma substituição "ingênua". Observe que o termo resultante possui uma semântica diferente da esperada porque a variável y do termo $f(x, y)$ não corresponde a variável y quantificada universalmente na fórmula dada. Como resolver este problema?

Definição 23. *Dados um termo t, uma variável x e uma fórmula ϕ, dizemos que t é livre para x em ϕ se nenhuma ocorrência livre de x em ϕ está no escopo de $\forall_y$ ou $\exists_y$ para qualquer variável y que ocorra em t.*

Exemplo 15. *Considere a fórmula $s(x) \wedge \forall_y(p(x) \to q(y))$, que possui duas ocorrências livres de x. A ocorrência de x mais a esquerda poderia, por exemplo, ser substituída pelo termo $f(y, y)$, no entanto a outra ocorrência não poderia ser substituída por este termo porque tal substituição acarretaria captura da variável y.*

Quando precisamos realizar uma substituição de um termo t que não está livre para uma variável x em uma fórmula ϕ, o que fazemos é renomear as variáveis ligadas para evitar capturas:

Exemplo 16. *No caso do exemplo anterior, a substituição de x por $f(y, y)$ em $s(x) \wedge \forall_y(p(x) \to q(y))$ pode ser resolvida renomeando a variável ligada y da fórmula para algum nome novo, por exemplo w: $s(x) \wedge \forall_w(p(x) \to q(w))$. Agora a substituição pode ser realizada sem provocar captura de variáveis.*