Lógica Computacional 1 (2025-1)

Flávio L. C. de Moura*

25 de março de 2025

A Lógica Proposicional

O Fragmento Implicacional da Lógica Proposicional

Partindo de um conjunto enumerável de variáveis proposicionais, iniciaremos nosso estudo pelo conectivo binário chamado *implicação*. Podemos representar as fórmulas que podem ser construídas com este conectivo pela seguinte gramática:

$$\varphi ::= p \mid (\varphi \to \varphi) \tag{1}$$

onde p denota um elemento de um conjunto enumerável de variáveis proposicionais, *i.e.* temos número finito, mas arbitrário, de variáveis proposicionais para construir uma fórmula. O construtor $\varphi \to \varphi$ diz que uma fórmula implicacional é construída a partir de duas fórmulas já construídas anteriormente. Por exemplo, se p e q denotam variáveis proposicionais então podemos concluir que $p \to q$ é uma fórmula, e neste caso, chamamos p de *antecedente*, e q de *sucedente* da implicação. Utilizando esta nova fórmula, podemos construir uma nova implicação a partir dela, e por exemplo, p, obtendo $(p \to q) \to p$ ou $p \to (p \to q)$, e assim por diante. Como indicado pela gramática (1), utilizaremos letras gregas minúsculas para representar fórmulas do fragmento implicacional da lógica proposicional.

Nosso objetivo agora é raciocinar sobre as estas fórmulas. Mais especificamente, queremos obter (ou derivar) novas informações a partir de informações conhecidas. Tudo isto em um contexto abstrato onde os símbolos proposicionais utilizados podem representar qualquer informação que corresponda a uma proposição. Utilizaremos a notação de *sequentes* para separar as informações (fórmulas) dadas da nova informação (fórmula) que queremos concluir. Chamaremos as fórmulas dadas de *premissas*, e a fórmula a ser derivada de *conclusão*, assim um sequente é formado por duas partes: um conjunto finito de fórmulas (que são as premissas), digamos Γ, e uma fórmula que é a conclusão, digamos φ, que escrevemos como $\Gamma \vdash \varphi$. Assim, se $\varphi_1, \varphi_2, \ldots, \varphi_n$ são as premissas de um sequente, e se ψ é a sua conclusão, então escrevemos $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ para representar o sequente que tem ψ como conclusão, e o conjunto $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$ de premissas. O conjunto $\{\varphi_1, \varphi_2, \ldots, \varphi_n\}$, isto é, a primeira componente do sequente $\varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ também pode ser chamado de *contexto* ao longo do texto, e normalmente será escrito sem as chaves que usualmente

*flaviomoura@unb.br

são usadas para representar conjuntos. Este é um abuso de linguagem usado para deixar a notação mais leve. Assim, se Γ denota um conjunto finito de fórmulas, ao invés de $\Gamma \cup \{\varphi\} \vdash \psi$, escreveremos simplesmente $\Gamma, \varphi \vdash \psi$, onde Γ, φ deve então ser lido como a união Γ com o conjunto unitário $\{\varphi\}$.

O conceito de prova agora será definido de forma mais precisa. Concretamente, uma prova (ou uma derivação) de um sequente da forma $\Gamma \vdash \psi$ é uma sequência de passos dedutivos, e um passo dedutivo consiste na aplicação de uma *regra de inferência* que possui a seguinte forma:

$$\frac{\Gamma_1 \vdash \gamma_1 \ \Gamma_2 \vdash \gamma_2 \ldots \Gamma_k \vdash \gamma_k}{\Gamma \vdash \psi}$$

onde $k \geq 0$. Quando $k = 0$ e $\Gamma = \{\psi\}$ a regra corresponde a um *axioma*:

$$\frac{}{\{\psi\} \vdash \psi}\ (Ax)$$

Uma prova (*i.e.* uma sequência de passos dedutivos) pode ser representada por meio de uma estrutura de árvore, onde os nós são anotados com sequentes. A raiz da árvore é anotada com o sequente que queremos provar, digamos, $\Gamma \vdash \psi$, e as folhas são axiomas. Quais são as regras de inferência que podem ser utilizadas no fragmento implicacional da lógica proposicional? Além do axioma apresentado acima, temos duas regras para a implicação. Antes de apresentá-las devemos lembrar que o sistema dedutivo que utilizaremos se chama *dedução natural*, e as regras deste sistema são divididas em dois tipos: *introdução* e *eliminação*. A regra de eliminação da implicação é conhecida pelo nome *modus ponens* e tem a seguinte estrutura:

$$\frac{\Gamma_1 \vdash \varphi \to \psi \qquad \Gamma_2 \vdash \varphi}{\Gamma_1 \cup \Gamma_2 \vdash \psi}\ (\to_e)$$

ou seja, para construirmos uma prova de um sequente com a forma $\Gamma \vdash \psi$ utilizando esta regra, precisamos construir duas outras provas: uma do sequente $\Gamma \vdash \varphi \to \psi$, e outra do sequente $\Gamma \vdash \psi$. Ou seja, na leitura desta regra de baixo para cima (*i.e.* da conclusão para as premissas) reduzimos o problema de provar $\Gamma \vdash \psi$ a dois outros problemas (potencialmente) mais simples. Esta regra também pode ser lida de cima para baixo (*i.e.* das premissas para a conclusão), e neste caso precisamos de uma prova de uma implicação, a saber $\Gamma \vdash \varphi \to \psi$, e de uma prova do antecedente desta implicação, a saber $\Gamma \vdash \psi$ para construirmos uma prova da conclusão da implicação, ou seja, uma prova de $\Gamma \vdash \psi$.

A regra de introdução é bastante intuitiva e, em certo sentido, nos fornece uma definição da implicação:

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \to \psi} \; (\to_i)$$

ou seja, na leitura de baixo para cima, para construirmos a prova de uma implicação precisamos construir uma prova do sucedente assumindo que temos uma prova do antecedente. Na leitura de cima para baixo, precisamos transformar uma prova do antecedente em uma prova do sucedente.

O interesse computacional do fragmento implicacional está diretamente relacionado ao algoritmo de inferência de tipos em linguagens funcionais[Hin97]. O fundamento teórico destas linguagens é o cálculo λ[**?**] desenvolvido por Alonzo Church em 1936 [Chu36, Chu40]. Para mais detalhes veja o Capítulo 1 de [AdM17]. Vejamos um exemplo.

Exemplo 1. *Considere o sequente* $\vdash (p \to q) \to (q \to r) \to p \to r$. *A primeira observação a ser feita aqui é que a implicação é associativa à direita, ou seja,* $\varphi \to \psi \to \gamma$ *deve ser lido como* $\varphi \to (\psi \to \gamma)$, *e não como* $(\varphi \to \psi) \to \gamma$. *Portanto, o sequente que queremos provar deve ser lido como* $\vdash (p \to q) \to ((q \to r) \to (p \to r))$. *Utilizando inicialmente a regra* $(\to_i)$, *temos a seguinte situação:*

$$\dfrac{p \to q \vdash (q \to r) \to (p \to r)}{\vdash (p \to q) \to ((q \to r) \to (p \to r))}\,(\to_i)$$

Podemos aplicar a regra $(\to_i)$ *mais duas vezes:*

$$\dfrac{\dfrac{\dfrac{p \to q, q \to r, p \vdash r}{p \to q, q \to r \vdash p \to r}\,(\to_i)}{p \to q \vdash (q \to r) \to (p \to r)}\,(\to_i)}{\vdash (p \to q) \to (q \to r) \to (p \to r)}\,(\to_i)$$

Agora não é mais possível utilizar a regra $(\to_i)$ *porque a conclusão* r *não é uma implicação, mas podemos utilizar a hipótese* $q \to r$ *para obter* r, *desde que tenhamos uma prova de* q *para utilizarmos* $(\to_e)$. *Neste ponto, a árvore é bifurcada em dois ramos e precisamos dividir o contexto de forma adequada em cada um dos ramos.*

$$\dfrac{\dfrac{\dfrac{\dfrac{p \to q, p \vdash q \qquad \dfrac{}{q \to r \vdash q \to r}\,(Ax)}{p \to q, q \to r, p \vdash r}\,(\to_e)}{p \to q, q \to r \vdash p \to r}\,(\to_i)}{p \to q \vdash (q \to r) \to (p \to r)}\,(\to_i)}{\vdash (p \to q) \to (q \to r) \to (p \to r)}\,(\to_i)$$

Observe que o ramo da direita consiste em um axioma já que a fórmula $q \to r$ *pertence ao conjunto de hipóteses. No ramo da esquerda podemos obter* q *por meio da regra* $(\to_e)$ *com as hipóteses* $p \to q$ *e* p. *A prova completa é dada a seguir:*

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash p}\,(Ax) \quad \dfrac{}{p \to q \vdash p \to q}\,(Ax)}{p \to q, p \vdash q}\,(\to_e) \quad \dfrac{}{q \to r \vdash q \to r}\,(Ax)}{p \to q, q \to r, p \vdash r}\,(\to_e)}{p \to q, q \to r \vdash p \to r}\,(\to_i)}{p \to q \vdash (q \to r) \to (p \to r)}\,(\to_i)}{\vdash (p \to q) \to (q \to r) \to (p \to r)}\,(\to_i)$$

Este primeiro exemplo possui uma característica importante: a escolha da regra a ser aplicada em cada passo é única, e portanto não temos outra opção para a construção desta prova! Se todo sequente tivesse esta característica, poderíamos construir um programa de computador que fizesse este trabalho para nós. Mesmo programas que não foram desenvolvidos especificamente para a construção destas provas podem resolver facilmente estas situações particulares.

Exemplo 2. *Considere o sequente* $\vdash (p \to q) \to (p \to p \to q)$. *Como no exemplo anterior, iniciaremos com uma aplicação da regra* $(\to_i)$:

$$\dfrac{p \to q \vdash p \to p \to q}{\vdash (p \to q) \to (p \to p \to q)}\,(\to_i)$$

Podemos aplicar a regra $(\to_i)$ *mais duas vezes:*

$$\dfrac{\dfrac{\dfrac{p \to q, p \vdash q}{p \to q, p \vdash p \to q}\,(\to_i)}{p \to q \vdash p \to p \to q}\,(\to_i)}{\vdash (p \to q) \to (p \to p \to q)}\,(\to_i)$$

Como o contexto é um conjunto de fórmulas, a nova introdução de p *não adiciona nenhuma fórmula no contexto porque os conjuntos* $\{p \to q, p\}$ *e* $\{p \to q, p, p\}$ *são idênticos. E agora, podemos concluir esta prova com uma aplicação de* $(\to_e)$:

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{p \vdash p}\,(Ax) \quad \dfrac{}{p \to q \vdash p \to q}\,(Ax)}{p \to q, p \vdash q}\,(\to_e)}{p \to q, p \vdash p \to q}\,(\to_i)}{p \to q \vdash p \to p \to q}\,(\to_i)}{\vdash (p \to q) \to (p \to p \to q)}\,(\to_i)$$

Exercício 3. *Prove o sequente* $\vdash (p \to p \to q) \to p \to q$.

Exercício 4. *Prove o sequente* $\vdash (q \to r \to t) \to (p \to q) \to p \to r \to t$.

Exercício 5. *Prove o sequente* $\vdash (p \to q \to r) \to (p \to q) \to p \to r$.

Exercício 6. *Prove o sequente* $\vdash (p \to q \to r) \to (q \to p \to r)$.

Exercício 7. *Prove o sequente* $\vdash (p \to q) \to (p \to r) \to (q \to r \to t) \to p \to t$.

Referências

[AdM17] M. Ayala-Rincón and F. L. C. de Moura. *Applied Logic for Computer Scientists - Computational Deduction and Formal Proofs*. UTCS. Springer, 2017.

[Chu36] A. Church. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics*, 58(2):345–363, 1936.

[Chu40] A. Church. A Formulation of the Simple Theory of Types. *journal of Symbolic Logic*, 5:56–68, 1940.

[Hin97] J. Roger Hindley. *Basic Simple Type Theory*. Number 42 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1997.