Lógica Proposicional

João Paulo Lazzarini Cyrino

14/09/2020

A lógica proposicional (ou cálculo proposicional) é um sistema formal dos mais simples, com poucas regras sintáticas e semânticas (de interpretação). Ela é especialmente útil para lidar com argumentos construidos a partir de conectivos, como *e*, *ou* e *se*.

Existem outros sistemas lógicos mais avançados, como a lógica de predicados (útil na semântica formal).

A sintaxe

A sintaxe de um sistema formal envolve as regras combinatórias dos símbolos que o compõe (vocabulário). Abaixo seguem as regras para a lógica proposicional.

Vocabulário: conjunto infinito de proposições atômicas (indivisíveis), representadas pelos símbolos *p*, *q*, *r*, *i*, com eventuais diacríticos etc, por exemplo *p'*.

Regras sintáticas: o que são fórmulas bem definidas (fbd)?

- Qualquer proposição atômica é uma fbd.
- Qualquer fórmula precedida por um símbolo de negação ($\neg$) é fbd.
- Duas fbd's podem se tornar uma nova fbd quando um dos símbolos abaixo ocorre entre elas e elas estão entre parênteses:
  - $\wedge$ conjunção
  - $\vee$ disjunção
  - $\rightarrow$ condicional
  - $\leftrightarrow$ bicondicional

Abaixo alguns exemplos de *fbds* na lógica proposicional:

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

Abaixo alguns exemplos de fórmulas que não fazem parte da lógica proposicional:

- pq
- $p \wedge r$ (sem parênteses)
- $m \vee s \neg t$
- $p \wedge m \vee q$
- (p) (não pode haver parênteses em torno de proposições atômicas)

A semântica

A semântica da lógica proposicional envolve apenas dois valores: Verdadeiro e Falso, ou *0* e *1*, *V* e *F* ou, para R, `TRUE` e `FALSE`.

Assume-se que toda proposição atômica possui um valor de verdade e toda fbd também possui um valor que depende de:

- O valor de verdade de seus componentes (proposições atômicas)
- O arranjo dos componentes com os conectivos

Nesse sentido, cada conectivo corresponde a uma função que toma dois valores e retorna um novo. Sabemos os resultados das operações com conectivos a partir das tabelas-verdade. Abaixo temos as tabelas verdade para cada um dos conectivos:

Negação

A negação simplesmente inverte o valor da proposição à qual ela se adjunge:

p	$\neg p$
1	0
0	1

Conjunção

A conjunção corresponde ao conectivo *e*, e só retorna verdadeiro se os dois valores conectados são verdadeiros:

p	q	$(p \wedge q)$
1	1	1
1	0	0
0	1	0
0	0	0

Disjunção

A disjunção corresponde ao conectivo *ou* com valor inclusivo. Ela só retorna falso se os dois valores conectados forem falsos. Caso contrário, ela sempre retornará verdadeiro:

p	q	$(p \vee q)$
1	1	1
1	0	1
0	1	1
0	0	0

Condicional

Condicinal corresponde ao *se*. Uma fórmula como $p \to q$ é lida como *se p então q*. Ela retornará falso somente quando o segundo elemento for falso:

p	q	$(p \to q)$
1	1	1
1	0	0
0	1	1
0	0	1

Bicondicional

A bicondicional corresponde ao *se e somente se*. Ela só é verdadeira se ambos os elementos forem verdadeiros ou falsos

p	q	$(p \leftrightarrow q)$
1	1	1
1	0	0
0	1	0
0	0	1

Calculando valores de verdade

Podemos calcular os valores de verdade de quaisquer formulas complexas a partir das tabelas. Tomemos um exemplo: $((p \wedge q) \rightarrow \neg(p \vee r))$.

Para calcular precisamos primeiro pensar nas combinações possíveis dos valores de verdade para cada uma das proposições atômicas que temos. Quando temos 2, como p e q, teremos 4 combinações: (1,1), (1,0), (0,1) e (1,1). Quando temos 3, como no caso do exemplo (p,q,r), temos $2 * *3$ combinações, ou seja, 8: (1,1,1),(0,1,1),(1,0,1),(0,0,1),(1,1,0),(0,1,0),(1,0,0),(0,0,0).

Para calcular, construimos, então, uma tabela com 8 linhas, cada uma correspondendo às possíveis combinações de valores de verdade de cada proposição atômica. Dessa forma:

p	q	r
1	1	1
0	1	1
1	0	1
0	0	1
1	1	0
0	1	0
1	0	0
0	0	0

Em seguida, adicionamos como colunas o resultado de cada expressão, de dentro para fora. Primeiramente, então, precisamos resolver $(p \wedge q)$ e $(p \vee r)$:

p	q	r	$(p \wedge q)$	$(p \vee r)$
1	1	1	1	1
0	1	1	0	1
1	0	1	0	1
0	0	1	0	1
1	1	0	1	1
0	1	0	0	1
1	0	0	0	0
0	0	0	0	0

Agora solucionamos a negação de $p \vee r$, ou $\neg(p \vee r)$, que é simplesmente inverter os valores da coluna correspondente:

p	q	r	$(p \land q)$	$(p \lor r)$	$\neg(p \lor r)$
1	1	1	1	1	0
0	1	1	0	1	0
1	0	1	0	1	0
0	0	1	0	1	0
1	1	0	1	1	0
0	1	0	0	1	0
1	0	0	0	0	1
0	0	0	0	0	1

Por fim, podemos solucionar toda a expressão $((p \land q) \to \neg(p \lor r))$

p	q	r	$(p \land q)$	$(p \lor r)$	$\neg(p \lor r)$	$((p \land q) \to \neg(p \lor r))$
1	1	1	1	1	0	0
0	1	1	0	1	0	1
1	0	1	0	1	0	1
0	0	1	0	1	0	1
1	1	0	1	1	0	0
0	1	0	0	1	0	1
1	0	0	0	0	1	1
0	0	0	0	0	1	1

É muito fácil de errar esse tipo de cálculo, principalmente quando temos mais que 3 proposições atômicas. Mas existe um jeito de fazer essa cálculo na linguagem R.

Primeiramente, precisamos criar um conjunto de valores lógicos possíveis, ou seja, um conjunto com `TRUE` e `FALSE`:

```
val <- c(TRUE, FALSE)
```

Agora, fazemos o produto cartesiano com a função `expand.grid`. Repetimos a variável que armazenda nosso conjunto (*val*) de acordo com o número de proposições atômicas e salvamos a tabela resultante em uma variável.

```
# Criar valores de verdade para uma expressão com três proposições atômicas:
tab <- expand.grid(val,val,val)
# Ver como ficou:
tab
```

```
##    Var1  Var2  Var3
## 1  TRUE  TRUE  TRUE
## 2 FALSE  TRUE  TRUE
## 3  TRUE FALSE  TRUE
## 4 FALSE FALSE  TRUE
## 5  TRUE  TRUE FALSE
## 6 FALSE  TRUE FALSE
## 7  TRUE FALSE FALSE
## 8 FALSE FALSE FALSE
```

Para facilitar, vamos converter a tabela *tab* em um data frame e também trocar os nomes das colunas para p,q e r:

```
# transformar tab e dataframe:
tab <- data.frame(tab)
```

```r
# trocar os nomes das colunas:
colnames(tab) <- c("p","q","r")
# ver como ficou:
tab
```

```
##       p     q     r
## 1  TRUE  TRUE  TRUE
## 2 FALSE  TRUE  TRUE
## 3  TRUE FALSE  TRUE
## 4 FALSE FALSE  TRUE
## 5  TRUE  TRUE FALSE
## 6 FALSE  TRUE FALSE
## 7  TRUE FALSE FALSE
## 8 FALSE FALSE FALSE
```

Agora, podemos calcular a expressão $((p \wedge q) \rightarrow \neg(p \vee r))$ como uma nova coluna de *tab*. Para isso, no entanto, precisamos entender como funcionam os cálculos lógicos em R. Temos apenas três operadores: ! para negação, & para conjunção e | para disjunção. Ou seja, não temos um operador para condicional nem para bicondicional. Porém, em lógica temos algumas expressões equivalentes. Concretamente, temos que: $(\neg p \vee q) = (p \rightarrow q)$, podemos, então, criar uma função `cond` que nos retorne o valor da condicional:

```r
# Criando a função:
cond <- function(p,q) !p | q
# Vamos testar em uma tabela verdade:
teste <- expand.grid(val,val)
teste[,3] <- cond(teste[,1],teste[,2])
# Agora ver se os resultados batem com os da tabela verdade condicional:
teste
```

```
##    Var1  Var2    V3
## 1  TRUE  TRUE  TRUE
## 2 FALSE  TRUE  TRUE
## 3  TRUE FALSE FALSE
## 4 FALSE FALSE  TRUE
```

Como podemos ver, como desejamos, nossa função `cond` apenas retorna falso quando o primeiro elemento é verdadeiro e o segundo, falso.

Dessa forma, para obter $((p \wedge q) \rightarrow \neg(p \vee r))$, podemos criar uma nova coluna em `tab` da seguinte forma:

```r
# Calcular e colocar o resultado na quarta coluna de tab:
tab[,4] <- cond((tab$p & tab $q), !(tab$p | tab$r))
# Ver o resultado:
tab
```

```
##       p     q     r    V4
## 1  TRUE  TRUE  TRUE FALSE
## 2 FALSE  TRUE  TRUE  TRUE
## 3  TRUE FALSE  TRUE  TRUE
## 4 FALSE FALSE  TRUE  TRUE
## 5  TRUE  TRUE FALSE FALSE
## 6 FALSE  TRUE FALSE  TRUE
## 7  TRUE FALSE FALSE  TRUE
## 8 FALSE FALSE FALSE  TRUE
```

Tautologias, Contradições e Contingências

Uma expressão lógica que sempre retorne valores verdadeiros é uma tautologia. Uma que sempre retorne valores falsos é uma contradição. As demais expressões são contingências: seus valores de verdade dependem dos valores de verdade das proposições atômicas.

O que é interessante sobre tautologias e contradições é que elas são propriedades das expressões e não dependem dos valores das proposições atômicas.

Exemplo de tautologia: $(p \vee \neg p)$

p	$\neg p$	$(p \vee \neg p)$
1	0	1
0	1	1

Exemplo de contradição: $(p \wedge \neg p)$

p	$\neg p$	$(p \wedge \neg p)$
1	0	0
0	1	0

Tautologias são importantes para sabermos se duas expressões são equivalentes. Ou seja, se $(p \leftrightarrow q)$ é uma tautologia, $p \Leftrightarrow q$.

Equivalências

Existe um conjunto de equivalências em lógica proposicional. Algumas delas seguem abaixo:

- Leis idempotentes:
  - $(p \wedge p) \Leftrightarrow p$
  - $(p \vee p) \Leftrightarrow p$
- Leis comutativas:
  - $(p \wedge q) \Leftrightarrow (q \wedge p)$
  - $(p \vee q) \Leftrightarrow (q \vee p)$
- Leis de identidade:
  - $(p \wedge 1) \Leftrightarrow p$
  - $(p \wedge 0) \Leftrightarrow 0$
  - $(p \vee 1) \Leftrightarrow 1$
  - $(p \vee 0) \Leftrightarrow p$
- Leis de DeMorgan:
  - $\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$
  - $\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$
- Leis condicionais:
  - $(p \rightarrow q) \Leftrightarrow (\neg p \vee q)$
  - $(p \rightarrow q) \Leftrightarrow (\neg q \rightarrow \neg p)$ (contraposição)

Dedução

Aprendemos uma série de cálculos dentro do sistema lógico proposicional, mas a pergunta que fica é: para que tudo isso serve?

Uma vez que estabelecemos uma sintaxe e semântica para um sistema lógico, podemos testar se certas conclusões derivam de certas premissas, contanto que ambas estejam na mesma linguagem. Com uma linguagem lógica, portanto, conseguimos provar uma conclusão a partir de premissas.

A lógica proposicional é muito simples e, portanto, não conseguimos com ela provar o famoso silogismo *todo homem é mortal, sócrates é homem, logo sócrates é mortal*. Mas conseguimos provar coisas mais básicas como:

- Se Maria mora em Salvador ela mora na Bahia.
- Se Maria mora na Bahia ela mora no Brasil.
- Logo:
  - Se Maria mora em Salvador, ela mora no Brasil.

Perceba que o importante não é o seu conhecimento sobre a geografia política do Brasil, mas sim que, se as duas primeiras afirmações (premissas) são verdadeiras, a terceira é necessariamente verdadeira por dedução lógica.

Escrevemos em linguagem lógica da seguinte forma:

- $(s \to b)$ se s então b
- $(b \to r)$ se b então r
- $\therefore (s \to r)$ logo, se s então r

Podemos checar a validade desse silogismo com uma tabela verdade para a expressão $((s \to b) \wedge (b \to r)) \to (s \to r)$, ou seja: se é verdade que $(s \to b)$ e $(b \to r)$ então é verdade que $(s \to r)$. Vamos calcular isso em R:

```r
# 3 proposições atômicas: s, b e r
val <- c(TRUE, FALSE)
tabela <- data.frame(expand.grid(val,val,val))
colnames(tabela) <- c("s","b","r")
# Vamos usar nossa função cond para calcular e apresentar o cálculo na coluna 4 da tabela:
tabela[,4] <- cond(cond(tabela$s,tabela$b) & cond(tabela$b,tabela$r),
                   cond(tabela$s,tabela$r))
# Vamos olhar o resultado
tabela
```

```
##       s     b     r   V4
## 1  TRUE  TRUE  TRUE TRUE
## 2 FALSE  TRUE  TRUE TRUE
## 3  TRUE FALSE  TRUE TRUE
## 4 FALSE FALSE  TRUE TRUE
## 5  TRUE  TRUE FALSE TRUE
## 6 FALSE  TRUE FALSE TRUE
## 7  TRUE FALSE FALSE TRUE
## 8 FALSE FALSE FALSE TRUE
```

Como vemos, o resultado apresentado na coluna 4 da tabela mostra verdadeiro para todas as situações. Ou seja, o silogismo é uma tautologia e isso prova que a conclusão decorre das premissas.

Abaixo provamos o silogismo: $(p \vee q), (q \to r), \neg r \therefore p$

```r
tabela <- data.frame(expand.grid(val,val,val))
colnames(tabela) <- c("p","q","r")
tabela[,4] <- cond(cond(tabela$q,tabela$r) & (tabela$p | tabela $q) & !tabela$r, tabela$p)
tabela
```

```
##       p     q     r   V4
## 1  TRUE  TRUE  TRUE TRUE
## 2 FALSE  TRUE  TRUE TRUE
## 3  TRUE FALSE  TRUE TRUE
## 4 FALSE FALSE  TRUE TRUE
## 5  TRUE  TRUE FALSE TRUE
## 6 FALSE  TRUE FALSE TRUE
```

```
## 7  TRUE FALSE FALSE TRUE
## 8 FALSE FALSE FALSE TRUE
```

Fazer provas dessa forma é bastante ágil e pode ser útil para que consigamos ter consistência em nosso raciocínio. Se quisermos saber se uma afirmação decorre de algumas premissas, basta traduzir tudo para essa linguagem da lógica proposicional e calcular.

Existem técnicas para obter provas manualmente e recomendo fortemente a leitura do livro *Mathematical Methods for Linguistics* (Partee, Meulen, Wall) para se aprofundar nesse assunto.