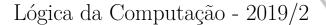
Insper



Aula 27/T13 - 13/Nov/2019

Raul Ikeda - rauligs@insper.edu.br

Objetivos

1. Lógica de Predicados

Observe as afirmações abaixo:

Javascript é derivada do C. Toda linguagem derivada do C é imperativa. Logo, Javascript é imperativa.

Você conseguiria modelar essas afirmações usando Lógica Proposicional?

Para não mentir sozinho, podemos testar isso no Prolog:

- https://swish.swi-prolog.org/
- Criar um novo programa
- Rules (quadro esquerdo):

```
\begin{aligned} &\operatorname{deriv\_c(javascript)}. \\ &\operatorname{imperativa}(X) :- &\operatorname{deriv\_c}(X). \end{aligned}
```

• Query (quadro direito inferior):

imperativa(javascript).

Apenas para ver o poder do Prolog, abrir o exemplo da resolução do Sudoku no interpretador acima.

Lógica de Predicados:

Como dito na aula anterior A Lógica de Primeira Ordem, ou Lógica de Predicados, é uma extensão da Lógica Proposicional, utilizando dos quantificadores \forall e \exists . Uma outra característica é possuir sentenças no formato de Predicados, ou P(X) como visto acima, ao passo que a Lp possui apenas premissas ou sentenças sem estruturas. Pode ser **monádica**, quando as funções possuem apenas um argumento, ou **poliádica**, quando possuem vários argumentos.

Exercício: Transcrever em lógica

Todos os sushis são crus. Nenhum cru é de porco. Então, nenhum sushi é de porco. A Linguagem Proposicional é **decidível** (Boolos et al Pag. 246), assim como a Linguagem de Primeira Ordem Monádica. Já a Poliádica é indecidível (Boolos et al Pag. 285).

Questão: Pesquisar sobre Lógica de Segunda Ordem.

Algumas Equivalências:

$$\neg \forall x P(x) \Leftrightarrow \exists x \neg P(x)$$

$$\neg \exists x P(x) \Leftrightarrow \forall x \neg P(x)$$

$$\forall x \forall y P(x, y) \Leftrightarrow \forall y \forall x P(x, y)$$

$$\exists x \exists y P(x, y) \Leftrightarrow \exists y \exists x P(x, y)$$

$$\forall x P(x) \land \forall x Q(x) \Leftrightarrow \forall x (P(x) \land Q(x))$$

$$\exists x P(x) \lor \exists x Q(x) \Leftrightarrow \exists x (P(x) \lor Q(x))$$

Algumas Inferências:

$$\exists x \, \forall y \, P(x,y) \Rightarrow \forall y \, \exists x \, P(x,y)$$
$$\forall x \, P(x) \vee \forall x \, Q(x) \Rightarrow \forall x \, (P(x) \vee Q(x))$$
$$\exists x \, (P(x) \wedge Q(x)) \Rightarrow \exists x \, P(x) \wedge \exists x \, Q(x)$$
$$\exists x \, P(x) \wedge \forall x \, Q(x) \Rightarrow \exists x \, (P(x) \wedge Q(x))$$

Teorema da Dedução:

1. Lp: Seja Γ um conjunto de fórmulas e A e B fórmulas. Então

$$\Gamma, A \models B \Leftrightarrow \Gamma \models A \to B$$

2. L
po: Considere que $\Gamma=\xi_1,...,\xi_n$ e φ são fórmulas. Então

$$\Gamma \vdash \varphi \Leftrightarrow \vdash \xi_1 \to (\xi_2 \to (\dots \to (\xi_n \to \varphi)\dots))$$

Teoremas da Incompletude de Gödel

Seja Q um conjunto finito de axiomas da aritimética minimal.

Primeiro Teorema de Incompletude de Gödel: "Não há nenhuma extensão axiomatizável, consistente e completa de Q." Boolos et al., Pag. 284.

Considere agora os axiomas da aritmética de Peano como um conjunto infinito de axiomas, através de passos indutivos, que estende \mathbf{Q} . Ainda, considere a teoria \mathbf{P} da aritmética de Peano como o conjunto de todas as sentenças da linguagem da aritmética que são demonstráveis a partir desses axiomas.

Segundo Teorema de Incompletude de Gödel: "Seja \mathbf{T} uma extensão consistente e axiomatizável de \mathbf{P} . Então a sentença de consistência para \mathbf{T} não é demonstrável em \mathbf{T} ." Boolos et al., Pag. 295.

Para verificar as consequências dos teoremas, precisamos também:

Teorema da indecidibilidade essencial: "Nenhuma extensão consistente de \mathbf{Q} é decidível (e, em particular, a própria \mathbf{Q} é indecidível)" Boolos et al., Pag. 284.

Traduzindo e demonstrando informalmente:

>> Ver Boolos et al. Cap. 17.

Próxima aula: Corrêa da Silva et al. Cap 7.

1. Verificação de Programas