## LÓGICA APLICADA À COMPUTAÇÃO

Mini Projeto AB2 - Elaboração de um tradutor de Linguagem Natural para Lógica Proposicional e verificador de validade de sentenças.

Link do projeto no GitHub: https://github.com/martanascimento1/logic-project

**Discentes:** Giovanna Alves Barbosa de Oliveira, Layza Nunes Faria Costa, Vitória Maria Santana Bigi, Marta Mirely Nascimento dos Santos

## Plataforma utilizada: Microsoft Copilot e GitHub Copilot

A princípio, foi pedido que a IA implementasse um código em python que verificasse a validade de premissas dadas em linguagem de lógica proposicional, com base nas regras de inferência dadas: Modus Ponens, Modus Tollens, Silogismo Hipotético, Silogismo Disjuntivo, e operações Jógicas (AND, OR, NOT)

```
project.py >  operacoes_logicas
   v def modus_ponens(premissas, conclusao):
 2 v if "P -> Q" in premissas and "P" in premissas:
             return True
         return False
 6 v def modus tollens(premissas, conclusao):
         if "P -> Q" in premissas and "/Q" in premissas:
             return True
        return False
11 ∨ def silogismo hipotetico(premissas, conclusao):
        if "A -> B" in premissas and "B -> C" in premissas:
             return True
         return False
     def silogismo_disjuntivo(premissas, conclusao):
         if "P * Q" in premissas and "/P" in premissas:
             return True
         return False
21 v def operacoes_logicas(premissas, conclusao):
         if "P & Q" in premissas and "P" in premissas:
             if "Q" in premissas:
                 return True
             return False
         elif "P & Q" in premissas and "Q" in premissas:
             if "P" in premissas:
                 return True
29
     """FORMATO DE ENTRADA EX.:
     PREMISSA = ["A -> B", "B -> C", "A"]
     CONCLUSÃO = "C" """
```

```
premissas_exemplo = input('Digite as premissas (usando letras de A a Z): ')
conclusao_exemplo = input("Digite a conclusão (usando letras de A a Z): ")

regras = [modus_ponens, modus_tollens, silogismo_hipotetico, silogismo_disjuntivo]

valido = False
for regra in regras:
    resultado = regra(premissas_exemplo, conclusao_exemplo)
    if resultado:
    valido = True
    break

if valido:
    print("O argumento é válido.")
else:
    print("O argumento é inválido.")
```

Como pode ser observado no print a seguir, o programa não estava retornando a saída correta para uma sentença dada relativamente simples:

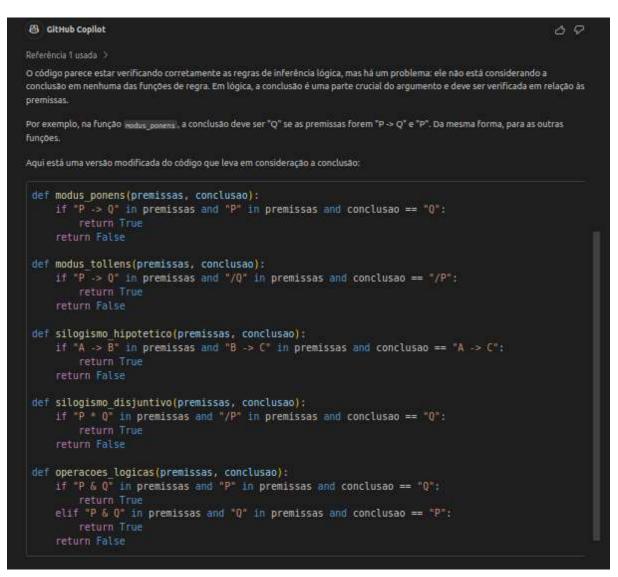
```
project.py >  operacoes_logicas
      PREMISSA = ["A -> B", "B -> C", "A"]
CONCLUSÃO = "C" """
      premissas_exemplo = input('Digite as premissas (usando letras de A a Z): ')
conclusao_exemplo = input("Digite a conclusão (usando letras de A a Z): ")
      regras = [modus_ponens, modus_tollens, silogismo_hipotetico, silogismo_disjuntivo]
      valido = False
      for regra in regras:
           resultado = regra(premissas exemplo, conclusao exemplo)
           if resultado:
                valido = True
                break
       if valido:
          print("O argumento é válido.")
       else:
           print("O argumento é inválido.")
                                     TERMINAL PORTS (2) GITLENS
giovanna@GIOVANNA-DESKTOP:~/ufal/logica/logic-project$ /bin/python3 /home/giovanna/ufal/logica/logic-project/project.py
Digite as premissas (usando letras de A a Z): ["P -> Q", "/P"]
Digite a conclusão (usando letras de A a Z): "Q
O argumento é válido.
giovanna@GIOVANNA-DESKTOP:~/ufal/logica/logic-project$
```

Dada a sentenças: "P -> Q" (se P, então Q), e "/P" (NOT P) e a conclusão "Q", a saída retornada deveria informar que o argumento é inválido, tendo em vista que a

conclusão correta seria "/Q" (NOT Q). No entanto, o programa retornou que o argumento era válido, fazendo assim com que fosse identificada a existência de um erro no mesmo.

Para corrigir o código, copiamos o mesmo e colamos no GitHub Copilot, informando que haviam erros para alguns casos de saída, e pedindo que este erro fosse corrigido:

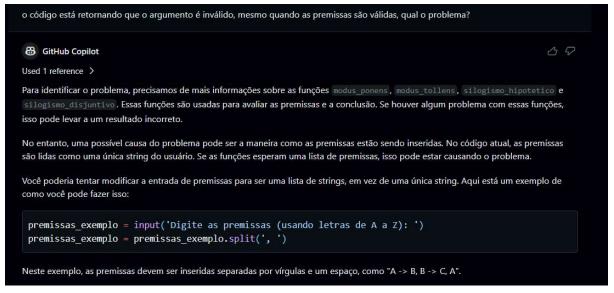
o código a seguir cumpre seu papel corretamento em boa parte dos testes, mas em alguns casos específicos ele da um resultado equivocado, tente achar onde está o erro que causa isso e me sugira uma modificação adequada: def modus\_ponens(premissas, conclusao): if "P -> Q" in premissas and "P" in premissas: return True return False



Após as correções apontadas pela IA, o código foi testado com as mesmas sentenças, e desta vez, a saída foi correta:

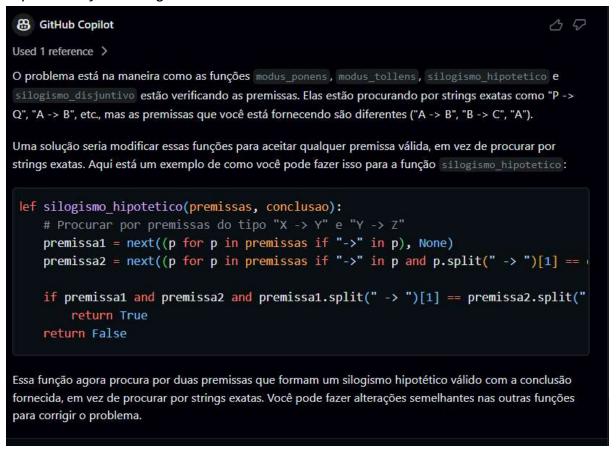
```
"""FORMATO DE ENTRADA EX.:
      PREMISSA = ["A -> B", "B -> C", "A"]
CONCLUSÃO = "C" """
      premissas_exemplo = input('Digite as premissas (usando letras de A a Z): ')
      conclusao_exemplo = input("Digite a conclusão (usando letras de A a Z): ")
      regras = [modus_ponens, modus_tollens, silogismo_hipotetico, silogismo_disjuntivo]
      valido = False
      for regra in regras:
          resultado = regra(premissas_exemplo, conclusao_exemplo)
          if resultado:
              valido = True
              break
     if valido:
         print("O argumento é válido.")
          print("O argumento é inválido.")
48
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS 5
giovanna@GIOVANNA-DESKTOP:~/ufal/logica/projetinho$ /bin/python3 /home/giovanna/ufal/logica/projetinho/interface.py
Digite as premissas (usando letras de A a Z): ["P -> Q", "/P"]
Digite a conclusão (usando letras de A a Z): "Q
O argumento é inválido.
giovanna@GIOVANNA-DESKTOP:~/ufal/logica/projetinho$
```

Mesmo após esta correção, identificamos que alguns casos de teste ainda estavam retornando a saída errada, e ao consultar a IA, fomos informados de que uma possível causa para os erros, seria o formato em que as premissas estavam sendo digitadas.



Com as correções sugeridas pelo programa, o modo em que as premissas deveriam ser digitadas, tornou-se mais fácil para o usuário. Além disso, a IA também nos informou que que o programa tinha sido construído para trabalhar com base em

strings exatas, ao invés de qualquer premissa válida, assim, sugerindo uma implementação mais geral.



Continuamos percebendo inconsistências durante as execuções do programa com diferentes quantidades de premissas e formatos de conclusão, e após análise, foi observado que para a execução correta do programa, as premissas e conclusão precisam estar digitadas corretamente do ponto de vista lógico. Assim, adicionamos um pequeno guia de entradas esperadas para que o verificador possa analisar corretamente:

```
DEBUG CONSOLE
                                     TERMINAL
giovanna@GIOVANNA-DESKTOP:~/ufal/logica/projetinho$ /bin/python3 /home/giovanna/ufal/lo
gica/projetinho/interface.py
FORMATOS DE ENTRADA E SAÍDA ESPERADOS:
MODUS PONENS:
premissas = P \rightarrow Q, P
conclusao = Q
MODUS TOLLENS:
premissas = P \rightarrow Q, /Q
conclusao = /P
SILOGISMO HIPOTÉTICO:
premissas = P \rightarrow Q, Q \rightarrow R
conclusao = P -> R
SILOGISMO DISJUNTIVO:
premissas = P * Q, /P
conclusão = Q
OPERAÇÕES LÓGICAS:
premissas = P & Q
conclusao = P
Digite as premissas (usando letras de A a Z):
```

Em relação a conversão das sentenças em linguagem natural para sentenças em linguagem lógica, nenhuma das IAs utilizadas conseguiu nos fornecer um algoritmo que fosse coerente na hora de cumprir essa tarefa. Tendo em vista que esse é um processo um pouco mais dificultoso, talvez elas ainda não possuam recursos adequados e suficientes.

A imagem abaixo representa um código fornecido pela IA generativa Github Copilot. Inicialmente a ideia era ter um algoritmo simples que converte uma sentença específica para linguagem lógica, e posteriormente seriam adicionadas funcionalidades que permitissem a entrada do próprio usuário com a linguagem natural, dentre outras, entretanto, nem com as sentenças estando já definidas no próprio código, não foi possível converter.

```
conversao.py - logic-project - Visual Studio Code
Arquivo Editar Seleção Ver Acessar Executar Terminal Ajuda
         conversao.py X
                 You, há 22 horas | 1 author (You)
                def traduzir_para_logica(sentenca):
          # Substituições para as palavras-chave
sentenca = sentenca.replace("E", "and")
sentenca = sentenca.replace("OU", "or")
sentenca = sentenca.replace("ENTÃO", "->")
sentenca = sentenca.replace("É falso que", "~")
                     return sentenca
11 # Exemplos de 300.
12 p = "Chove em Curitiba"
           q = "uso guarda-chuva"
 A
                 sentencal = f"{p} E {q}"
(1)
                sentenca2 = f"Tweety é um pássaro OU Tweety é um mamífero"
           17 sentenca3 = f"Se {p} ENTÃO {q}"
                sentenca4 = f"É falso que Tweety é um mamífero"
          print(traduzir_para_logica(sentenca1)) # Output: p and q
print(traduzir_para_logica(sentenca2)) # Output: r or s
print(traduzir_para_logica(sentenca3)) # Output: r -> u
           24 print(traduzir_para_logica(sentenca4)) # Output: ~u
(2)
    🎖 main 🔾 🐉 ⊗ 0 🛕 0 🕞 logic-project 🎇 Debug 🚳 🕨 🛱 🛍 👾 0 −INSERT−
                                                                                                                                            Ln 26, C
```

A saída esperada para as sentenças colocadas no código acima seriam:

P and Q

R or S

 $R \rightarrow U$ 

~U

No entanto, como a conversão não obteve sucesso, a saída foi a seguinte: