Emerson Costa, Rafael Carvalho, Vinicius Almeida

Documentação dos Códigos em Prolog

Brasil - São João del Rei/MG
2025

Sumário

1 Introdução

Este documento apresenta a documentação detalhada dos códigos desenvolvidos em Prolog para o trabalho da disciplina de Introdução à Lógica. O trabalho foi dividido em três níveis de complexidade, cada um abordando conceitos específicos da linguagem Prolog:

- **Nível 1**: Fundamentos de Prolog com fatos e regras básicas
- **Nível 2**: Inferência recursiva e manipulação de listas
- **Nível 3**: Desenvolvimento de um sistema especialista

Para cada nível, será apresentado o código desenvolvido, seguido de uma explicação detalhada da lógica implementada, exemplos de consultas e interpretação dos resultados.

2 Nível 1 - Fundamentos de Prolog

2.1 Visão Geral

O Nível 1 tem como objetivo apresentar os conceitos iniciais da linguagem Prolog, criando um pequeno conjunto de fatos e regras em um domínio simples. O tema escolhido foi "Consultor de Cardápios", que permite consultar itens de um cardápio organizados por tipo e categoria.

2.2 Código Fonte

```
% Fatos:
item(feijoada, prato_principal, brasileira).
item(pizza, prato_principal, italiana).
item(sushi, prato_principal, japonesa).
item(carbonara, prato_principal, italiana).
item(churrasco, prato_principal, brasileira).
item(pudim, sobremesa, doce).
item(tiramisu, sobremesa, doce).
item(brigadeiro, sobremesa, doce).
item(suco_laranja, bebida, nao_alcoolica).
item(agua_coco, bebida, nao_alcoolica).

% Regras:
prato_principal_da_culinaria(Prato, Culinaria) :-
    item(Prato, prato_principal, Culinaria).

sobremesa_disponivel(Sobremesa) :-
    item(Sobremesa, sobremesa, _).

bebida_nao_alcoolica(Bebida) :-
    item(Bebida, bebida, nao_alcoolica).
```

Listing 2.1 – Nível 1 - Consultor de Cardápios

2.3 Explicação da Lógica

2.3.1 Fatos

Os fatos definem os itens do cardápio e suas características. Cada fato `item/3` possui três argumentos:

- Nome do item (átomo)
- Tipo (prato_principal, sobremesa ou bebida)
- Categoria (nacionalidade para pratos principais, tipo para sobremesas e bebidas)

Foram definidos 10 fatos, atendendo ao requisito mínimo do trabalho.

2.3.2 Regras

Três regras foram implementadas para consultar os fatos de forma mais genérica:

1. `prato_principal_da_culinaria/2`: Encontra pratos principais de uma culinária específica.
    - Unifica `Prato` com itens onde o tipo é `prato_principal` e a categoria corresponde a `Culinaria`.
2. `sobremesa_disponivel/1`: Encontra todas as sobremesas disponíveis.
    - Usa uma variável anônima (_) para ignorar a categoria específica da sobremesa.
3. `bebida_nao_alcoolica/1`: Encontra bebidas não alcoólicas.
    - Filtra itens onde o tipo é `bebida` e a categoria é `nao_alcoolica`.

2.4 Exemplos de Consultas

1. `prato_principal_da_culinaria(X, brasileira).`
    - **Resultado:** X = feijoada; X = churrasco.
    - **Interpretação:** Retorna todos os pratos principais da culinária brasileira.
2. `sobremesa_disponivel(X).`
    - **Resultado:** X = pudim; X = tiramisu; X = brigadeiro.

- **Interpretação:** Retorna todas as sobremesas disponíveis, independente de sua categoria específica.

3. `bebida_nao_alcoolica(X).`

   - **Resultado:** X = suco_laranja; X = agua_coco.
   - **Interpretação:** Retorna todas as bebidas não alcoólicas do cardápio.

3 Nível 2 - Inferência Recursiva e Manipulação de Listas

3.1 Visão Geral

O Nível 2 aprofunda a construção de regras em Prolog utilizando recursão e manipulação de listas. O tema escolhido foi "Sistema de Rastreio de Rotas Aéreas", que permite encontrar rotas entre cidades, incluindo conexões, e manipular essas rotas como listas.

3.2 Código Fonte

```
% Fatos:
voo_direto(sao_paulo, rio_de_janeiro).
voo_direto(rio_de_janeiro, brasilia).
voo_direto(brasilia, salvador).
voo_direto(sao_paulo, belo_horizonte).
voo_direto(belo_horizonte, vitoria).
voo_direto(vitoria, salvador).
voo_direto(salvador, fortaleza).
voo_direto(rio_de_janeiro, curitiba).
voo_direto(curitiba, porto_alegre).
voo_direto(fortaleza, manaus).
voo_direto(manaus, belem).
voo_direto(porto_alegre, florianopolis).

% Regras:
rota(Origem, Destino) :-
    voo_direto(Origem, Destino).
rota(Origem, Destino) :-
    voo_direto(Origem, CidadeIntermediaria),
    rota(CidadeIntermediaria, Destino).

caminho(Origem, Destino, [Origem, Destino]) :-
    voo_direto(Origem, Destino).
caminho(Origem, Destino, [Origem | RestoCaminho]) :-
```

```
25      voo_direto(Origem, CidadeIntermediaria),
26      caminho(CidadeIntermediaria, Destino, RestoCaminho).
27
28  rota_evitando(Origem, Destino, CidadeEvitada, [Origem,
        Destino]) :-
29      voo_direto(Origem, Destino),
30      Origem \= CidadeEvitada,
31      Destino \= CidadeEvitada.
32  rota_evitando(Origem, Destino, CidadeEvitada, [Origem | Resto
        ]) :-
33      voo_direto(Origem, Intermediaria),
34      Intermediaria \= CidadeEvitada,
35      rota_evitando(Intermediaria, Destino, CidadeEvitada,
            Resto).
```

Listing 3.1 – Nível 2 - Sistema de Rastreio de Rotas Aéreas

3.3 Explicação da Lógica

3.3.1 Fatos

Os fatos `voo_direto/2` definem conexões aéreas diretas entre cidades. Cada fato representa uma rota direta de uma cidade de origem para uma cidade de destino.

3.3.2 Regras

Três regras principais foram implementadas:

1. `rota/2`: Determina se existe uma rota (direta ou com conexões) entre duas cidades.
    - **Caso base:** Existe um voo direto entre Origem e Destino.
    - **Passo recursivo:** Existe um voo direto para uma cidade intermediária e uma rota dessa cidade até o destino.
2. `caminho/3`: Encontra um caminho (lista de cidades) entre origem e destino.
    - **Caso base:** Constrói uma lista [Origem, Destino] se houver voo direto.
    - **Passo recursivo:** Constrói a lista adicionando a Origem à frente do subcaminho encontrado recursivamente.

3. `rota_evitando/4`: Encontra um caminho que evita uma cidade específica.
    - Verifica em cada passo se a cidade atual ou a próxima não é a cidade a ser evitada.
    - Usa o operador `\=` para garantir que as cidades não sejam iguais à cidade evitada.

3.4 Exemplos de Consultas

1. `rota(sao_paulo, salvador).`
    - **Resultado:** true.
    - **Interpretação:** Existe uma rota de São Paulo até Salvador (possivelmente com conexões).
2. `caminho(sao_paulo, salvador, Caminho).`
    - **Resultado:** Caminho = [sao_paulo, rio_de_janeiro, brasilia, salvador];
    - **Interpretação:** Retorna um possível caminho com conexões de São Paulo até Salvador.
3. `rota_evitando(sao_paulo, salvador, brasilia, Caminho).`
    - **Resultado:** Caminho = [sao_paulo, belo_horizonte, vitoria, salvador];
    - **Interpretação:** Retorna um caminho que evita passar por Brasília.

4 Nível 3 - Sistema Especialista

4.1 Visão Geral

O Nível 3 consiste no desenvolvimento de um sistema especialista simples, capaz de tomar decisões com base em regras lógicas e interações com o usuário. O tema escolhido foi "Diagnóstico Básico de Falhas em Redes Domésticas", que auxilia usuários a identificar e resolver problemas comuns em suas redes domésticas.

4.2 Código Fonte

```
1
2  % Inicio do diagnostico
3  % Da as boas-vindas, orienta o usuario e coleta o sintoma
       principal.
4  diagnosticar_rede :-
5      writeln('Ola! Vou te ajudar a diagnosticar problemas na
           sua rede domestica.'),
6      writeln('Diga qual o problema principal: sem_internet,
           wi_fi_lento ou nenhum_problema.'),
7      writeln('Digite o nome exatamente como esta, seguido de
           ponto final.'),
8      read(Sintoma),
9      iniciar_diagnostico(Sintoma).
10
11 % Decide o fluxo com base no sintoma informado
12 iniciar_diagnostico(sem_internet) :-
13     writeln('Vamos analisar a falta de conexao com a internet
           .'),
14     pergunta_luzes_modem.
15 iniciar_diagnostico(wi_fi_lento) :-
16     writeln('Vamos investigar a lentidao do Wi-Fi.'),
17     pergunta_interferencia_wi_fi.
18 iniciar_diagnostico(nenhum_problema) :-
19     writeln('Otimo! Sem problemas detectados.').
20 iniciar_diagnostico(_) :-
```

```
21      writeln('Sintoma nao reconhecido. Tente: sem_internet,
            wi_fi_lento ou nenhum_problema.').
22
23  % --- Fluxo para o problema "sem_internet" ---
24
25  % Verifica se o modem/roteador esta com luzes normais
26  pergunta_luzes_modem :-
27      writeln('As luzes do seu modem/roteador estao acesas e
            estaveis? (sim/nao)'),
28      read(Resposta),
29      (   Resposta == sim
30      ->  writeln('Tudo certo com o aparelho. Vamos ver se o
            problema e no provedor.'),
31          pergunta_conectar_outros_dispositivos
32      ;   Resposta == nao
33      ->  writeln('Pode ser problema fisico ou de energia.'),
34          recomendar_verificar_cabos_reiniciar
35      ;   writeln('Responda com "sim." ou "nao."'),
36          pergunta_luzes_modem
37      ).
38
39  % Verifica se outros dispositivos conseguem se conectar
40  pergunta_conectar_outros_dispositivos :-
41      writeln('Outros aparelhos conseguem se conectar a
            internet? (sim/nao)'),
42      read(Resposta),
43      (   Resposta == sim
44      ->  recomendar_problema_dispositivo_especifico
45      ;   Resposta == nao
46      ->  writeln('Parece um problema geral na rede.'),
47          pergunta_ping_provedor
48      ;   writeln('Responda com "sim." ou "nao."'),
49          pergunta_conectar_outros_dispositivos
50      ).
51
52  % Simula teste de conexao externa
53  pergunta_ping_provedor :-
54      writeln('Voce tentou reiniciar o modem/roteador e esperar
            5 minutos? (sim/nao)'),
55      read(RespostaReiniciar),
```

```
56      (   RespostaReiniciar == sim
57      ->  writeln('Se ja tentou isso e nao resolveu...'),
58          writeln('Pode ser falha do provedor.'),
59          recomendar_contatar_provedor
60      ;   RespostaReiniciar == nao
61      ->  writeln('Reinicie o modem e teste novamente.'),
62          recomendar_reiniciar_e_testar
63      ;   writeln('Responda com "sim." ou "nao."'),
64          pergunta_ping_provedor
65      ).
66
67  % --- Fluxo para o problema "wi_fi_lento" ---
68
69  % Verifica se ha obstaculos atrapalhando o sinal
70  pergunta_interferencia_wi_fi :-
71      writeln('Ha muitos obstaculos entre seu dispositivo e o
            roteador? (sim/nao)'),
72      read(Resposta),
73      (   Resposta == sim
74      ->  writeln('Isso pode afetar bastante o sinal.'),
75          recomendar_otimizar_posicao_roteador
76      ;   Resposta == nao
77      ->  writeln('Vamos checar outros fatores.'),
78          pergunta_muitos_dispositivos
79      ;   writeln('Responda com "sim." ou "nao."'),
80          pergunta_interferencia_wi_fi
81      ).
82
83  % Verifica se a rede esta sobrecarregada
84  pergunta_muitos_dispositivos :-
85      writeln('Ha muitos aparelhos usando a rede ao mesmo tempo
            ? (sim/nao)'),
86      read(Resposta),
87      (   Resposta == sim
88      ->  recomendar_limitar_uso_ou_upgrade
89      ;   Resposta == nao
90      ->  recomendar_verificar_atualizacoes_firmware
91      ;   writeln('Responda com "sim." ou "nao."'),
92          pergunta_muitos_dispositivos
93      ).
```

```
94
95  % --- Recomendacoes finais ---
96
97  recomendar_verificar_cabos_reiniciar :-
98      writeln('SUGESTAO: Verifique se os cabos estao bem
            conectados e reinicie o modem.').
99
100 recomendar_problema_dispositivo_especifico :-
101     writeln('SUGESTAO: O problema pode estar no seu aparelho.
            Tente reinicia-lo ou ajustar as configuracoes de rede
            .').
102
103 recomendar_contatar_provedor :-
104     writeln('SUGESTAO: Entre em contato com seu provedor de
            internet.').
105
106 recomendar_reiniciar_e_testar :-
107     writeln('SUGESTAO: Reinicie seu modem. Desligue por 1
            minuto e ligue novamente.').
108
109 recomendar_otimizar_posicao_roteador :-
110     writeln('SUGESTAO: Mude o roteador para um lugar mais
            aberto e central, ou aproxime o dispositivo.').
111
112 recomendar_limitar_uso_ou_upgrade :-
113     writeln('SUGESTAO: Reduza o numero de dispositivos
            conectados ou pense em melhorar seu plano/roteador.').
114
115 recomendar_verificar_atualizacoes_firmware :-
116     writeln('SUGESTAO: Verifique se ha atualizacoes de
            firmware no roteador ou, se necessario, faca um reset
            de fabrica.').
117
118 % Dica inicial para o usuario
119 ajuda :-
120     writeln('Digite: diagnosticar_rede. para iniciar.').
```

Listing 4.1 – Nível 3 - Diagnóstico de Falhas em Redes Domésticas

4.3 Explicação da Lógica

O sistema especialista implementado segue uma abordagem baseada em regras (if-then) para diagnosticar problemas em redes domésticas. A lógica principal está organizada em:

4.3.1 Fluxo Principal

1. `diagnosticar_rede/0`: Ponto de entrada do sistema, coleta o sintoma principal do usuário.

2. `iniciar_diagnostico/1`: Direciona para o fluxo de diagnóstico específico baseado no sintoma.

4.3.2 Regras de Decisão

O sistema possui dois fluxos principais de diagnóstico:

1. **Problema "sem internet"**:

   - Verifica o estado das luzes do modem
   - Testa a conexão em outros dispositivos
   - Recomenda ações específicas baseadas nas respostas

2. **Problema "wi-fi lento"**:

   - Verifica interferências físicas
   - Avalia o número de dispositivos conectados
   - Fornece recomendações para melhorar a velocidade

4.3.3 Mecanismo de Inferência

O sistema utiliza encadeamento lógico de condições através de:

- **Operador condicional (->) para implementar regras if-then:** O núcleo do sistema baseia-se em regras "se-então"(if-then) implementadas com o operador condicional (->) do Prolog. Este operador permite que o sistema avalie uma condição (a resposta do usuário a uma pergunta) e, se verdadeira, execute uma ação ou prossiga para uma nova meta. Por exemplo, se a resposta para o estado das luzes do modem é "sim", o sistema procede para verificar a conexão em outros dispositivos.

- **Recursão para repetir perguntas em caso de respostas inválidas:** Para garantir a validade das respostas do usuário, o sistema utiliza recursão. Se uma resposta inválida for fornecida (que não seja "sim"ou "não"), a pergunta correspondente é repetida até que uma entrada válida seja recebida. Isso assegura que o fluxo de inferência só avance com dados consistentes.
- **Padrões de encadeamento para guiar o usuário através do fluxo de diagnóstico:** O diagnóstico é guiado por padrões de encadeamento que direcionam o usuário passo a passo através do fluxo de resolução de problemas. A partir de um sintoma inicial (sem_internet ou wi_fi_lento), o sistema avança através de uma sequência lógica de perguntas e verificações. Cada resposta do usuário determina o próximo passo na cadeia de inferência, levando a um diagnóstico e recomendação específica.

4.4 Exemplos de Interação

1. **Cenário 1: Sem internet**
    - Usuário informa: `sem_internet.`
    - Sistema pergunta sobre luzes do modem
    - Usuário responde: `sim.`
    - Sistema pergunta se outros dispositivos estão concetados
    - Usuário responde: `nao.`
    - Sistema sugere que seja um problema geral na rede, e pergunta se usuário ja reiniciou o modem
    - Usuário responde *sim.*
    - Sistema responde *Pode ser falha no provedor. Entrem em contato*
2. **Cenário 2: Wi-Fi lento**
    - Usuário informa: `wi_fi_lento.`
    - Sistema pergunta sobre obstáculos
    - Usuário responde: `nao.`
    - Sistema pergunta se existem multiplos aparelhos conectados a rede
    - Usuário responde: `sim.`
    - Sistema sugere reduzir o numero de aparelhos concetados ou melhorar o plano de internet.

3. **Cenário 3: Sem problemas**

   - Usuário informa: `nenhum_problema.`
   - sistema confirma que não há problemas

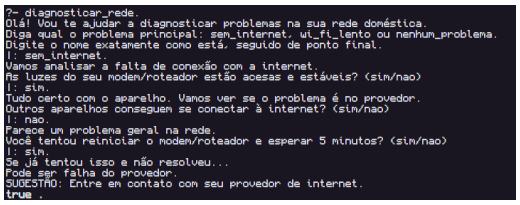

Figura 1 – Execução de cenário 1

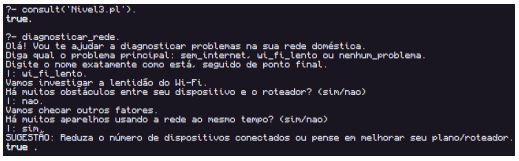

Figura 2 – Execução de cenário 2

5 Conclusão

Este trabalho demonstrou a aplicação prática da linguagem Prolog em três níveis de complexidade crescente. No Nível 1, foram explorados os conceitos básicos de fatos e regras. O Nível 2 introduziu recursão e manipulação de listas para resolver problemas mais complexos. Por fim, o Nível 3 mostrou como combinar essas técnicas para construir um sistema especialista capaz de interagir com usuários e tomar decisões baseadas em regras.

A implementação atendeu a todos os requisitos especificados, demonstrando o potencial da programação em lógica para aplicações de inteligência artificial e sistemas especialistas. Cada nível apresentou desafios específicos que foram superados através da aplicação adequada dos conceitos de Prolog.