Sistema especialista
Relatório

Antonio Rodrigues da Mata Neto

Adson Wesley Silva de Souza

Gabriel Mac'Hamilton Renaux Alves

Thomas Torreao de Brito Bastos

Modelagem em lógica de primeira ordem

1 – Todo paciente que tem fraqueza, tonturas, desmaios, taquicardia, palidez, palpitações ou icterícia e tem uma quantidade de hemoglobina ou de hematócritos baixa tem anemia.

$$\forall p\ (fraq(p) \vee tont(p) \vee desm(p) \vee taq(p) \vee pali(p) \vee palp(p) \vee ict(p) \wedge (hemob(p) \vee hemab(p)) \rightarrow anem(p))$$

2 – Todo paciente homem tem uma quantidade de hemoglobina baixa se os níveis de hemoglobina forem menores que 135 gramas por litro.

$$\forall p\ ((niv135(p) \wedge homem(p)) \rightarrow hemob(p))$$

3 – Toda paciente mulher tem uma quantidade de hemoglobina baixa se os níveis de hemoglobina forem menores que 120 gramas por litro.

$$\forall p\ ((niv120(p) \wedge mulher(p)) \rightarrow hemob(p))$$

4 – Todo paciente que tem anemia hemolítica congênita tem anemia, além de ter o nível de RBC do sangue baixo e tem um histórico congênito hemolítico e um determinante hemolítico congênito.

$$\forall p\ (anem(p) \wedge rbcb(p) \wedge (histcong(p) \wedge determcong(p)) \rightarrow anemhc(p))$$

5 – Todo paciente que tem um histórico congênito hemolítico tem icterícia, cálculos biliares, esfenomegalia, hepatomegalia, malformações ósseas ou retardo mental.

$$\forall p\ (ict(p) \vee calc(p) \vee esf(p) \vee hepato(p) \vee malf(p) \vee rm(p) \rightarrow histcong(p))$$

6 – Todo paciente que tem um determinante hemolítico congênito tem anemia relacionada à comida ou microcitose, eliptocitose, esferocitose ou anisopoikilocitose.

$$\forall p\ (anemcom(p) \vee (mic(p) \vee elip(p) \vee esfe(p) \vee aniso(p)) \rightarrow determcong(p))$$

7 – Todo paciente que tem anemia hemolítica adquirida tem anemia, um nível de Lactate dehydrogenase alto e não tem anemia hemolítica congênita.

$$\forall p\ (anem(p) \wedge ldha(p) \wedge \neg(anemhc(p)) \rightarrow anemha)$$

8 – Todo paciente que tem anemia por deficiência não tem anemia hemolítica congênita nem anemia hemolítica adquirida.

$$\forall p\ ((anem(p) \wedge \neg(anemha) \wedge \neg(anemhc)) \rightarrow anemdef(p))$$

Comentários sobre o código

```
1   :- use_module(library(pce)).
2   :- use_module(library(pce_style_item)).
3   :- dynamic sim/1,nao/1,homem/1,mulher/1,maior/1,menor/1.
4
5
6   consulta :-
7           new(D, dialog("Sistema especialista para deteccao de anemia e seus subtipos")),
8           send(D, size, size(640,480)),
9           new(T, text("O sistema ira tentar diagnosticar o tipo de anemia do paciente de acordo com
10          send(D, display, T, point(50,80)),
11          send(D, append(button(ok, message(D, destroy)))),
12          send(D, open),
13          sleep(4),
14          pergunta_digitar_n(Nome, "Informe o nome do paciente"),
15          hipotese(Resultado), nl,
16          new(K, dialog(Nome)),
17          new(Y, text("Segundo a analise, o paciente aparenta ", center)),
18          send(K, display, Y, point(50,80)),
19          new(Z, text(Resultado, center)),
20          send(K, display, Z, point(305,80)),
21          send(K, size, size(500,300)),
22          send(K, append(button(ok, message(K, destroy)))),
23          send(K, open),
24          registro(Resultado).
25
```

1 - O código começa carregando o módulo gráfico do Prolog, o XPCE, e seu módulo adicional. O XPCE será usado para a interface do sistema.

3 – Linha para declarar os predicados que podem ser mudados durante a execução do programa. A sua funcionalidade será mais bem explicada quando a função `assert/1` for usada.

6 – A linha de começo do código. O usuário começa o sistema digitando `consulta`. depois de carregar a base de dados no Prolog. Aqui são encontradas várias funções para construção da interface.

14 – A função é usada para abrir uma janela que recebe o input do usuário, nesse caso, o nome. Mais bem explicada posteriormente.

15 – A chamada para começar a análise das hipóteses, ao final da análise, `Resultado` guardará o valor que representa um dos três tipos de anemia (ou que diz que o paciente não possui anemia).

16 – Mais funções de interface. Essas mostram o resultado da analise.

24 – Função para registrar o resultado do paciente em um arquivo txt.

```
27    /* hipoteses para teste */
28
29    hipotese(anemia_hemolitica_congenita) :- anemia_hemolitica_congenita, !.
30    hipotese(anemia_hemolitica_adquirida) :- anemia_hemolitica_adquirida, !.
31    hipotese(anemia_por_deficiencia) :- anemia_por_deficiencia, !.
32    hipotese(nao_ser_anemico). /* paciente nao apresenta sinais de anemia */
33
34     /* regras de identificacao */
35
36    anemia_hemolitica_congenita :- anemia, rbc_baixo, historico_congenito, determinante_congenito.
37
38    anemia_hemolitica_adquirida :- anemia, ldh_alto.
39
40    anemia_por_deficiencia :- anemia.
```

29 – As hipóteses que serão testadas durante a execução. Se nenhuma das hipóteses for comprovada, o paciente não apresenta nenhum sinal de anemia. São usados cortes (!) nas hipóteses, pois, de acordo com as regras de identificação, o paciente não pode ter mais de um tipo de anemia, então quando o sistema chega a uma conclusão, ele não analisa outras hipóteses.

36 – Regras de identificação para os diferentes tipos de anemia. Essas regras são a tradução das regras já mencionadas nesse relatório. Então a regra:

4 – Todo paciente que tem anemia hemolítica congênita tem anemia, além de ter o nível de RBC do sangue baixo e tem um histórico congênito hemolítico e um determinante hemolítico congênito.

$\forall p\ (anem(p) \land rbcb(p) \land (histcong(p) \land determcong(p)) \rightarrow anemhc(p))$

É traduzida para:

```
anemia_hemolitica_congenita :- anemia, rbc_baixo, historico_congenito, determinante_congenito.
```

Sendo o `:-` um implica e as virgulas um `e`.

```
42    anemia :-
43        verificarsexo(Paciente),
44        (verificar(fraqueza);
45        verificar(tonturas);
46        verificar(desmaios);
47        verificar(taquicardia);
48        verificar(palidez);
49        verificar(palpitacoes);
50        verificar(ictericia)),
51        compararhemo(Paciente),
52        (compararhemo(Paciente);
53        compararmenor(36,hematocritos)).
54
```

42 – Regras para que o paciente seja identificado como anêmico. A função `verificarsexo/1` é usada aqui para facilitar a comparação do nível de hemoglobina, mas não influencia na detecção geral de anemia como regra, pois sempre retorna `true`.

Como nos tipos de anemia, as regras aqui também são equivalentes as anteriores, desta vez usando o `;` como `ou`.

A função verificar retorna `true` ou `fail` dependendo da resposta do paciente (`resposta = y ; resposta = yes` implica em `true`, qualquer input que não seja yes ou y retorna `fail`). As funções de comparação são similares, porém comparam o input do paciente a outros valores. Essas funções serão explicadas posteriormente.

```
55  rbc_baixo :-
56      compararmenor(4,rbc),
57      compararmenor(4,rbc).
58
59  historico_congenito :-
60      (verificar(ictericia);
61      verificar(calculos_biliares);
62      verificar(esfenomegalia);
63      verificar(hepatomegalia);
64      verificar(malformacoes_osseas);
65      verificar(retardo_mental)).
66
67
68  determinante_congenito :-
69      (verificar(microcitose);
70      verificar(eliptocitose);
71      verificar(esferocitose);
72      verificar(anisopoikilocitose));
73      verificar(anemia_relacionada_a_comida).
74
75  ldh_alto :-
76      compararmaior(333, lactate_dehydrogenase),
77      compararmaior(333, lactate_dehydrogenase).
```

55 – Mais regras de identificação.

Como visto anteriormente, o programa irá checar três hipóteses de anemia. A ordem usada nessas checagens é a seguinte:

```
35
36  anemia_hemolitica_congenita :- anemia, rbc_baixo, historico_congenito, determinante_congenito.
        1                          2       3           4                   5
```

Para 1 ser verdade, 2, 3, 4 e 5 também tem que ser. Então ele começa checando o 2 e fazendo as perguntas correspondentes, se 2 for verdade, ele checa o 3, se não, ele vai para a outra hipótese:

```
37
38  anemia_hemolitica_adquirida :- anemia, ldh_alto.
```

O problema aqui é que anemia está sendo checada novamente, isso quer dizer que todas as perguntas (verificar(fraqueza), verificar(tonturas), verificar(desmaios)...) serão feitas novamente ao paciente.

Para evitar essas repetições, é usada a função assert/1.

Avançando um pouco no código:

```
104  verificar(X) :-
105      (sim(X)
106      -> true ;
107          nao(X) -> fail ;
108              perguntar(X)).
109
```

A função `verificar/1` era usada para conferir se o paciente sofria de um sintoma `X`, Ex. `verificar(tontura)`

Essa função apresenta uma estrutura (nesse caso, nested) de if/else,

`( Condição ->  Then ; Else )`

`sim(tontura)` e `nao(tontura)` ainda não foram definidos, então a função `perguntar(tontura)` é chamada.

```
81  perguntar(Questao) :-
82          pergunta_digitar(Resposta, "(y/n)O paciente apresenta ", Questao),
83          ( (Resposta == yes ; Resposta == y)
84          -> assert(sim(Questao)) ;
85          assert(nao(Questao)), fail).
```

A função `perguntar_digitar/3` é usada apenas para mostrar a pergunta na interface e aceitar o input, que será igual à `Resposta`.

Aqui existe outra clausula condicional, se a resposta for sim (`y; yes`), a função `assert/1` irá mudar o predicado de `Questão` (`tontura` no caso) para `sim`, qualquer outra resposta implicará em uma mudança para `nao`.

Agora temos o predicado `sim(tontura)` ou `nao(tontura)`, dependendo do input do usuário. Da próxima vez que o programa quiser conferir `anemia`, ele vai passar por essa mesma verificação em `verificar(tontura)`.

```
104  verificar(X) :-
105      (sim(X)
106      -> true ;
107          nao(X) -> fail ;
108              perguntar(X)).
109
```

Porém, como `sim(X)` ou `nao(X)` já foram definidos, ele simplesmente retorna `true` ou `fail` dependendo do predicado, e a pergunta não é repetida para o usuário.

```
verificarsexo(Pessoa) :-
    (homem(Pessoa)
    ->  true ;
        mulher(Pessoa)
        -> true ;
            perguntagenero(Pessoa)).

compararmenor(S, X) :-
    (menor(X)
    -> true ;
        maior(X) -> fail ;
            perguntacomp(S, X)).

compararmaior(S, X) :-
    (maior(X)
    -> true ;
        menor(X) -> fail ;
            perguntacomp(S, X)).

compararhemo(Pessoa) :-
    ( (homem(Pessoa))
    -> compararmenor(135, hemoglobina);
        ( (mulher(Pessoa))
        -> compararmenor(120, hemoglobina);
            write("Invalido"))).
```

As outras funções se comportam de maneira similar, cada uma com sua respectiva função de pergunta para diferentes tipos de questionamento. Diferentes perguntas têm diferentes predicados (pois eles refletem as respostas). Funções de comparação utilizam os predicados `maior(X)` e `menor(X)` enquanto a função de verificação de sexo utiliza `homem(X)` e `mulher(X)`.

As funções restantes são usadas para as perguntas na interface. É criada uma janela dialog com uma barra para receber o input como texto e botões de cancelar e confirmar. Além disso, existe uma função extra que abre um arquivo de texto e escreve o resultado do exame nele.

```
/* desfazer os asserts */

undo :- retract(sim(_)),fail.
undo :- retract(nao(_)),fail.
undo :- retract(sexo(_)),fail.
undo :- retract(valor(_)),fail.
undo.
```

Finalmente, essas declarações desfazem os predicados construídos pela `assert /1` na sessão atual de prolog, para que não fiquem guardados dados de outro paciente em outra execução do programa.

```
pergunta_digitar_n(Resp, Pergunta) :-
        new(D, dialog("Sistema anemia")),
        new(G, text(Pergunta)),
        send(D, display, G, point(50,80)),
        send(D, size, size(500, 200)),
        send(D, append(new(ItemNome, text_item(r)))),
        send(D, append(button(ok, message(D, return, ItemNome?selection)))),
        send(D, append(button(cancel, message(D, return, @nil)))),
        send(D, default_button(ok)),
        get(D, confirm, Rval),
        free(D),
        Rval \== @nil,
        Resp = Rval.

pergunta_digitar(Resp, Pergunta, Termo) :-
        new(D, dialog("Sistema anemia")),
        new(G, text(Pergunta)),
        new(J, text(Termo)),
        new(U, text("?")),
        send(D, display, U, point(300,8)),
        send(D, display, J, point(170,8)),
        send(D, display, G, point(10,10)),
        send(D, size, size(500, 200)),
        send(D, append(new(ItemNome, text_item(r)))),
        send(D, append(button(ok, message(D, return, ItemNome?selection)))),
        send(D, append(button(cancel, message(D, return, @nil)))),
        send(D, default_button(ok)),
        get(D, confirm, Rval),
        free(D),
        Rval \== @nil,
        Resp = Rval.

informar_niveis(Resp, Pergunta, Termo) :-
        new(D, dialog("Sistema anemia")),
        new(G, text(Pergunta)),
        new(J, text(Termo)),
        send(D, display, J, point(170,8)),
        send(D, display, G, point(10,10)),
        send(D, size, size(500, 200)),
        send(D, append(new(ItemNome, text_item(r)))),
        send(D, append(button(ok, message(D, return, ItemNome?selection)))),
        send(D, append(button(cancel, message(D, return, @nil)))),
        send(D, default_button(ok)),
        get(D, confirm, Rval),
        free(D),
        Rval \== @nil,
        atom_number(Rval, X),
        Resp = X.

registro(Resultado) :-
        open(resultado, write, X),
        write(X,Resultado),
        write("registrado"),
        close(X).
```

Bibliografia

Wielemaker, Jan. Anjewierden, Anjo. *Programming in XPCE/Prolog*. Web.
http://www.swi-prolog.org/packages/xpce/UserGuide/index.html

Triska, Markus. *Expert Systems in Prolog*. Web.
https://www.metalevel.at/prolog/expertsystems

Monroy, Jr. Jesse. *Beginners Notes to Using Prolog*. Web.
http://www2.dcs.hull.ac.uk/NEAT/dnd/AI/beginprolog.html

"Learn Prolog Now!" *Learn Prolog Now!*. Web.
http://learnprolognow.org/

Amzi! Inc. Embeddable Extendable Prolog, Logic Server, Knowledge Engineering, Rule Engines, Artificial Intelligence. Web.
http://www.amzi.com/ExpertSystemsInProlog/xsipfrtop.htm

"Artificial Intelligence in Motion." *DiagnostiCar (1) - Introduction*. Web.
http://aimotion.blogspot.com.br/2011/06/diagnosticar-1-introduction.html

XPCE Examples. Web.
http://www.swi-prolog.org/packages/xpce/examples.html