Lógica para Programação

Relatório do Projecto

2010/2011

Base de Dados de Cinema

Grupo	**Nº: 66964**	Daniel Cardoso
40	**Nº: 66986**	Francisco Raposo

Breve descrição da implementação de cada funcionalidade

todosFilmes(Ano) - existindo a estrutura filme(F_id, Nome, Ano_estreia, Lugar_top_250) na base de dados foi só procurar todos filmes cujo Ano_estreia fosse o Ano pedido.

todosRealizadores – usando as estruturas pessoa(P_id, Nome, Ano_nascimento, Ano_morte), actividade(A_id, Nome) e participa(P_id, F_id, A_id), sabemos que a actividade realizador então foi só procurar todos os nomes de pessoas que participem em filmes como realizador.

maisQueNAnos(Num) – usando as estruturas já acima mencionadas, verificamos se a diferença entre os Anos de estreia entre dois filmes em que uma pessoa participa e maior ou igual a Num, todas as pessoas que verifiquem essa condição são colocadas numa lista. Se num for 1 ou menor, são impressos os nomes de todas as pessoas que participem em pelo menos um filme.

maisQueNOscares(Num, Actividade)- são verificados o numero de Óscares de cada pessoa em Actividade se forem mais ou os mesmos que Num o nome de essa pessoa e inserido na lista. O número de Óscares e calculado usando a estrutura nomeada(P_id, F_id, A_id, Ganhou?) em conjunto com as anteriores. Inserindo todos os Óscares ganhos numa lista e verificando o seu comprimento.

maisOscares(Actividade) – são inseridos os números de Óscares de cada pessoa numa lista, é verificado o maior valor dessa lista e é chamado o predicado anterior com esse valor, imprimindo assim apenas as pessoas com o num máximo de Óscares.

maisQueNFilmes(Num) - são verificados o numero de participações em filmes de cada pessoa, se forem mais ou os mesmos que Num o nome de essa pessoa e inserido na lista. O número de participações e calculado usando o predicado num_participacoes(P_id, Num) e as estruturas anteriores. Inserindo todos as participações numa lista e verificando o seu comprimento.

maisFilmes - são colocados numa lista o número de participações em filmes de todas as pessoas, e calculado o valor máximo dessa lista e depois e chamado o predicado anterior com esse valor, sendo assim devolvidos os nomes com o num máximo de participações.

redeSocial(Nome1, Nome2) - é percorrida a árvore de pessoas conhecidas do Nome1 em profundidade. Isto e, e encontrada uma pessoa(Nome3) que entre num filme com Nome1 se essa pessoa for Nome2 acaba a pesquisa, se não e verificada uma pessoa que entre num filme com Nome3 e Nome3 e inserido numa lista da ligação possível entre Nome1 e Nome2. Caso esse caminho de em falso essa Lista de ligação possível e descartada e é verificada outra pessoa que entre num filme com Nome1.

Pesquisa em língua Natural – foram separadas as palavras da frase usando o predicado mencionado no enunciado do projecto. Ficando assim com uma lista em que cada elemento e uma das palavras. Depois fizemos um predicado que tenta unificar os elementos da lista com as palavras das frases do enunciado. Caso unifiquem chama o predicado indicado.

Descrição dos principais problemas

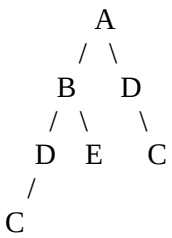
Inicialmente ao realizar os predicados médios, a nossa ideia era colocar numa lista todas as pessoas com Óscares na actividade pedida e posteriormente verificar se o numero de Óscares de cada pessoa, encontramos vários problemas ao criar novas listas com essas pessoas e em verificar se o numero de Óscares era maior que o pedido ou não, achamos também que esta não era a melhor solução, em relação a eficiência.

Acabamos por resolver este problema criando um predicado que calcula o numero de Óscares de uma determinada pessoa, e assim quando encontramos uma pessoa com o numero de Óscares maior ou igual ao pedido essa pessoa e logo inserida na lista.

Tivemos problemas semelhantes e soluções semelhantes nos predicados maisQueNFilmes e maisFilmes e as soluções encontradas foram igualmente semelhantes as anteriormente descritas.

Em relação a Rede Social nós começamos por tentar realizar uma pesquisa em largura, isto é procurar se existia uma ligação entre as duas pessoas usando o mínimo número de pessoas, ou seja o menor caminho entre essas pessoas. Se eles tivessem trabalhado no mesmo filme era fácil ou ate se eles estivessem ligados por apenas uma pessoa essa pessoa era fácil de encontrar também, mas em níveis mais baixos tivemos problemas em resolver a recursividade e obtivemos overflow da stack variadas vezes. Acabamos por desistir dessa ideia e procurar em profundidade, vendo um caminho possível ate encontrar uma pessoa que já tinha sido vista antes ou encontrar a pessoa desejada. Mesmo assim esta solução não e muito eficiente e caso as pessoas não se conheçam e tenham uma grande lista de amigos, facilmente o Prolog fica imenso tempo a procura só resposta ao predicado.

Um problema que mesmo assim encontramos nesta nossa solução foi, por exemplo:



Nesta caso queremos a redeSocial entre A e H. A é amigo de B e de D, ele começa por ir verificar B e D e C, o predicado falha, vai testar o caminho pelo E amigo de D falha também. Volta ao A e vai verificar o D, que já foi verificado uma vez, mas a nossa lista de pessoas visitadas só esta a guardar as pessoas ate obter esse

caminho ter sucesso ou falhar. Caso falhe essa lista e descartada e vai testar outro caminho. Nesta caso não e muito grave mas imaginando que D tinha uma lista maior de amigos, o guardar a lista de pessoas visitadas sempre, pouparia bastante tempo de execução.

Exemplos de utilização que demonstrem o correcto funcionamento dos predicados:

```
1 ?- todosFilmes(2001).
The Lord of the Rings: The Fellowship of the Ring
+
true.

2 ?- todosRealizadores.
martin scorsese
bryan singer
steven spielberg
quentin tarantino
katia lund
francis ford coppola
michael curtiz
frank darabont
milos forman
alfred hitchcock
peter jackson
irvin kershner
akira kurosawa
sergio leone
george lucas
sidney lumet
fernando meirelles
+
true.

3 ?- maisQueNAnos(20).
john rhys-davies
brad dourif
ron gilbert
+
true.

4 ?- maisQueNOscares(1,'realizador').
steven spielberg
francis ford coppola
michael curtiz
milos forman
peter jackson
+
```

true.

5 ?- maisOscares('realizador').
steven spielberg
francis ford coppola
michael curtiz
milos forman
peter jackson
+
true .

6 ?- maisQueNFilmes(7).
peter jackson
george lucas
+
true.

7 ?- maisFilmes.
peter jackson
+
true.

8 ?- redeSocial('george lucas','jono manks').
alfred molina
dennis muren
glenn randall jr.
john rees
john rhys-davies
+
true.

9 ?- questao('mostre todos os filmes de 1994').
Pulp Fiction
The Shawshank Redemption
+
true.

10 ?- questao('quem são os realizadores').
martin scorsese
bryan singer
steven spielberg
quentin tarantino
katia lund
francis ford coppola
michael curtiz
frank darabont
milos forman
alfred hitchcock

peter jackson
irvin kershner
akira kurosawa
sergio leone
george lucas
sidney lumet
fernando meirelles
+
true.

11 ?- questao('quem e o melhor actor').
frase desconhecida
true.

```
%%%%%%%%%%%%%%%%%%%%%%%%%
%Grupo Numero 40        %
%Daniel Cardoso 66964   %
%Francisco Raposo 66986 %
%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%       PARTE 1 - Pesquisa Simples           %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Funcoes Auxiliares

imprime(Lista):- imprime_lista(Lista),writeln('+').

imprime_lista([]).
imprime_lista([Elem]) :- imprime_elemento(Elem),!.
imprime_lista([Elem|Resto]) :- imprime_elemento(Elem),
imprime_lista(Resto),!.

imprime_elemento((X,_,_)) :- writeln(X),!.
imprime_elemento((X,_)) :- writeln(X),!.
imprime_elemento(X) :- writeln(X),!.

membro(X,[H|_]) :- X==H,!.
membro(X,[_|T]) :- membro(X,T).

removeDuplicados([],[]).
removeDuplicados([H|T],C) :- membro(H,T),!, removeDuplicados(T,C).
removeDuplicados([H|T],[H|C]) :- removeDuplicados(T,C).

maxList([A],A).
maxList([A|List],Max):-
     maxList(List,Max1),(A>=Max1, Max=A; A<Max1, Max=Max1),!.

%Predicados Basicos
% 1. todosFilmes(Ano): devolve uma lista sem elementos repetidos contendo
todos os
% nomes dos filmes do ano Ano;

todosFilmes(Ano) :- findall(Nome,filme(_,Nome,Ano,_),ListaNomesFilmes_Rep)
               ,removeDuplicados(ListaNomesFilmes_Rep,ListaNomesFilmes)
               ,imprime(ListaNomesFilmes).

% 2. todosRealizadores: devolve uma lista sem elementos repetidos contendo
todos os
% nomes dos realizadores da base de dados

todosRealizadores :-
     findall(Nome,(pessoa(P_id,Nome,_,_),actividade(A_id,'realizador'),
               participa(P_id,_,A_id)),ListaTodosRealizadores_Rep),

     removeDuplicados(ListaTodosRealizadores_Rep,ListaTodosRealizadores),
               imprime(ListaTodosRealizadores).

% 3. maisQueNAnos(Num): devolve uma lista sem elementos repetidos com todos
os nomes
```

```
% das pessoas que entraram em filmes durante Num ou mais anos
(independentemente da actividade).

maisQueNAnos(Num):-  Num>=2,!,
               findall(Nome,(pessoa(P_id,Nome,_,_),filme(F_id1,_,Ano1,_),

     participa(P_id,F_id1,_),filme(F_id2,_,Ano2,_),participa(P_id,F_id2,_),
               diferenca(Num,Ano1,Ano2)),ListaPessoasMaisQUeNAnos_Rep),

     removeDuplicados(ListaPessoasMaisQUeNAnos_Rep,ListaPessoasMaisQUeNAnos
),
               imprime(ListaPessoasMaisQUeNAnos).

maisQueNAnos(Num):-  Num<2,!,
               findall(Nome,(pessoa(P_id,Nome,_,_),participa(P_id,_,_)),
               ListaPessoasMaisQueNAnos_Rep),

     removeDuplicados(ListaPessoasMaisQueNAnos_Rep,ListaPessoasMaisQueNAnos
),
               imprime(ListaPessoasMaisQueNAnos).

%funcao auxiliar a alinea 3 dos Predicados Basicos

     % diferenca(Num,Ano1,Ano2): predicado que indica se o valor absoluto
     % da diferenca entre Ano1 e Ano2 e maior ou igual a Num

     diferenca(Num,Ano1,Ano2):- abs(Ano1 - Ano2) >= Num.

%Predicados Medios

% 1. maisQueNOscares(Num, Actividade): devolve uma lista sem elementos
repetidos
% com o(s) nome(s) da(s) pessoa(s) que ganhou(aram) Num ou mais óscares,
tendo como
% actividade Actividade;

maisQueNOscares(Num, Actividade) :-
oscaresNumActividade(Actividade,Num,ListaPessoasMaisQueNOsc_Rep),
removeDuplicados(ListaPessoasMaisQueNOsc_Rep,ListaPessoasMaisQueNOsc),
imprime(ListaPessoasMaisQueNOsc).

% Funcoes Auxiliares a alinea 1 dos predicados medios.

     % oscaresNumActividade(Actividade,Num,Lista): e um predicado que
indica que os Nomes
     % em Lista(possivelmente repetidos) tem mais que Num oscares a
realizar Actividade

     oscaresNumActividade(Actividade,Num,ListaPessoasMaisQueNOsc) :-
findall(Nome,

     (pessoa(P_id,Nome,_,_),actividade(A_id,Actividade),num_oscares(P_id,A_
id, Num_Oscares),
               Num_Oscares>=Num)
                 ,ListaPessoasMaisQueNOsc).
```

```
     % num_oscares(P_id, A_id, Num) :- e um predicado que indica que a
pessoa com id P_id
     % ganhou Num oscares a realizar a actividade com id A_id.

     num_oscares(P_id, A_id, Num_Oscares) :-

     findall((P_id),(filme(F_id,_,_,_),participa(P_id,F_id,A_id),nomeada(P_
id,F_id,A_id,1)), Lista_Oscares),
          length(Lista_Oscares, Num_Oscares).

% 2. maisOscares(Actividade): devolve uma lista sem elementos repetidos com
o(s)
% nome(s) da(s) pessoa(s) que ganhou(aram) mais óscares, tendo como
actividade Actividade

maisOscares(Actividade) :-
oscaresActividade(Actividade,ListaNumOscaresEmActividadePorPessoa),maisOsca
res(Actividade,ListaNumOscaresEmActividadePorPessoa),!.

% Funcoes Auxiliares a alinea 2 dos predicados medios.

     %oscaresActividade(Actividade,Lista): e um predicado que indica que os
numeros em Lista
     % sao sao os numeros de oscares de todas as pessoas a realizar
Actividade, sendo que
     % tem de ter mais que 0 oscares.

     oscaresActividade(Actividade,Lista) :-
findall(Num_Oscares,(pessoa(P_id,_,_,_)

     ,actividade(A_id,Actividade),num_oscares(P_id,A_id,Num_Oscares),Num_Os
cares>0),Lista).

     % maisOscares(Actividade,ListaNumOscaresEmActividade) : devolde uma
lista sem elementos
     % repetidos com os nomes das pessoas com o maior numero de oscares em
Lista a realizar
     % Actividade. Caso Lista seja vazia devolve a lista vazia.

     maisOscares(Actividade,ListaNumOscaresEmActividade):-
maxList(ListaNumOscaresEmActividade,NumMaximo),maisQueNOscares(NumMaximo,
Actividade).

     maisOscares(_,[]):- imprime([]).

% 3. maisQueNFilmes(Num): devolve uma lista sem elementos repetidos
contendo o(s)
% nome(s) das pessoas que entraram em Num ou mais filmes, independentemente
do cargo
% (isto é, se uma pessoa entrou num filme como actor e noutro como
realizador, deve contar
% como dois filmes;

maisQueNFilmes(Num) :-
participaNumFilmes(Num,ListaPessoasComMaisQueNFilmes_Rep),
```

```
     removeDuplicados(ListaPessoasComMaisQueNFilmes_Rep,ListaPessoasComMais
QueNFilmes),
               imprime(ListaPessoasComMaisQueNFilmes).

% Funcoes Auxiliares a alinea 3 dos predicados medios.

     % participaNumFilmes(Num,Lista): e um predicado que indica que os
Nomes
     % em Lista(possivelmente repetidos) participam em mais que Num filmes

     participaNumFilmes(Num,Lista):-findall(Nome,
               (pessoa(P_id,Nome,_,_),filme(F_id,_,_,_)
               ,participa(P_id,F_id,_),num_participacoes(P_id,Num_Filmes),
               Num_Filmes>=Num),Lista).


     % num_participacoes(P_id, Num) :- e um predicado que indica que a
pessoa com id P_id
     % participou em Num filmes.

     num_participacoes(P_id, Num_Participacoes) :-
          findall(F_id, participa(P_id, F_id,_), Lista_filmes),
          length(Lista_filmes, Num_Participacoes).

% 4. maisFilmes: devolve uma lista sem elementos repetidos com o(s) nome(s)
da(s)
% pessoa(s) que entrou(aram) em mais filmes, independentemente da
actividade. Ou seja, se 4
% pessoas entraram em 6 filmes e esse e o maximo valor obtido, devem ser
devolvidos os
% nomes dessas 4 pessoas;

maisFilmes :-
participaFilmes(ListaPessoasParticipamFilmes),maisFilmes(ListaPessoasPartic
ipamFilmes).

% Funcoes Auxiliares a alinea 4 dos predicados medios.

     % participaFilmes(Lista): e um predicado que indica que os numeros em
Lista
     % sao sao os numeros de participacoes em filmes de todas as pessoas
sendo que
     % tem de participar em mais que 0 filmes.

     participaFilmes(Lista):-findall(Num_Filmes,
               (pessoa(P_id,_,_,_),filme(F_id,_,_,_),participa(P_id,F_id,_)
               ,num_participacoes(P_id,Num_Filmes),Num_Filmes>0),Lista).


     % maisFilmes(ListaPessoasParticipamFilmes) : devolde uma lista sem
elementos
     % repetidos com os nomes das pessoas com o maior numero de
participacoes em
     % filmes em Lista. Caso Lista seja vazia devolve a lista vazia.

     maisFilmes(ListaPessoasParticipamFilmes):-
```

```
maxList(ListaPessoasParticipamFilmes,NumMaximoParticipacoes),maisQueNFilmes
(NumMaximoParticipacoes).

     maisFilmes([]):- imprime([]).

%Predicados Avancados
% 1. redeSocial(Nome1, Nome2): devolve uma lista com o nome de pessoas que
% representam uma ligação possível entre as pessoas com nome Nome1 e Nome2.
Por exemplo,
% se o Nome1 contracenou com a actriz Xpto1 que, por sua vez entra num
filme realizado
% por Xpto2 que dirige outro filme em que entra Nome2, então a lista
[Xpto1, Xpto2] deve
% ser devolvida.

redeSocial(Nome1,Nome2) :- pessoa(_,Nome2,_,_),
     redeSocial(Nome1,Nome2,[_|LigacaoPossivel],[Nome1])
     ,!,imprime(LigacaoPossivel).

redeSocial(Nome1,Nome2,LigacaoEntre1e2,PessoasVistasAteAgora):-
     pessoa(P_id1,Nome1,_,_),
     pessoa(P_id2,Nome2,_,_),
     participa(P_id1,F_id,_),
     participa(P_id2,F_id,_),
        LigacaoEntre1e2=PessoasVistasAteAgora.

redeSocial(Nome1,Nome2,LigacaoEntre1e2,PessoasVistasAteAgora):-
     pessoa(P_id1,Nome1,_,_),
     participa(P_id1,F_id,_),
     participa(P_id3,F_id,_),
     pessoa(P_id3,Nome3,_,_),
     not(membro(Nome3,PessoasVistasAteAgora)),
     append(PessoasVistasAteAgora,[Nome3],PessoasVistasAteAgoraMais3),
     redeSocial(Nome3,Nome2,LigacaoEntre3e2,PessoasVistasAteAgoraMais3),

     LigacaoEntre1e2=LigacaoEntre3e2.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%    PARTE 2 - Pesquisa em Lingua Natural    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% O objectivo desta segunda etapa do projecto e permitir um conjunto de
% pesquisas em portugues (ex:mostre todos os filmes de 1967), relativas
% a parte da funcionalidade implementada anteriormente. As frases
% utilizadas nesta nova pesquisa deverao ser confirmadas como fazendo
% parte do limitado conjunto de frases compreendidas pelo sistema e
% posteriormente associadas a um dos predicados implementados
% anteriormente que sera de seguida invocado, permitindo a pesquisa em
% causa.
%
% Esta parte e realizada atraves do predicado questao(Frase) que
% identifica a Frase e faz a pesquisa desejada ou se nao conhecer a
% frase devolve 'frase desconhecida'.


questao(Frase) :- concat_atom(Lista,' ',Frase),hipotese(Lista).
```

```
hipotese(['filmes','de',Ano]):-
     term_to_atom(Num,Ano),
     number(Num),
     todosFilmes(Num),!.

hipotese(['mostre','todos','os','filmes','de',Ano]) :-
     term_to_atom(Num,Ano),
     number(Num),
     todosFilmes(Num),!.

hipotese(['liste','todos','os','filmes','de',Ano]) :-
     term_to_atom(Num,Ano),
     number(Num),
     todosFilmes(Num),!.

hipotese(['quais','os','filmes','de',Ano]):-
     term_to_atom(Num,Ano),
     number(Num),
     todosFilmes(Num),!.


hipotese(['mostre','os','realizadores']) :-
     todosRealizadores,!.

hipotese(['liste','os','realizadores']) :-
     todosRealizadores,!.

hipotese(['quem','sao','os','realizadores']) :-
     todosRealizadores,!.

hipotese(['quem','ganhou','mais','oscares','como',Actividade]) :-
     actividade(_,Actividade), maisOscares(Actividade),!.

hipotese(_):- writeln('frase desconhecida').
```