Trabalho Final Prolog

Victor Borges Zuccolotto - **11811BCC034**
Vinnicius Pereira da Silva - **11821BCC046**

Lyfoes é um jogo no qual existem copos com cores diferentes e o objetivo é deixar os copos com a mesma cor, sendo que só se pode passar uma cor a um copo que tenha como cor no topo a mesma cor e não esteja cheio ou um copo vazio.

Codigo:

```
cheioVazio([]) :- !.
cheioVazio([X,X,X,X]).


fim([]) :- !.
fim([H|T]) :- cheioVazio(H), fim(T).


de_para([],[H,H,H,H],_,_):- !,fail.%se uma lista vazia e uma cheia nao necessita iteracao
de_para([],[H,H,H],_,_):- !,fail.
de_para([],[H,H],_,_):- !,fail.
de_para([],[_],_,_):- !,fail.

de_para([H,H,H,H],[],_,_):- !,fail.
de_para([H,H,H],[],_,_):- !,fail.
de_para([H,H],[],_,_):- !,fail.
de_para([_],[],_,_):- !,fail.

de_para([H,X,Y],[H],[X,Y],[H,H]):- X\=H,Y\=H.
de_para([H],[H,X,Y],[H,H],[X,Y]):- X\=H,Y\=H.

de_para([H,X,Y],[H,H],[X,Y],[H,H,H]):- X\=H,Y\=H.
de_para([H,H],[H,X,Y],[H,H,H],[X,Y]):- X\=H,Y\=H.

de_para([H,X,Y],[H,H,H],[X,Y],[H,H,H,H]):- X\=H,Y\=H.
de_para([H,H,H],[H,X,Y],[H,H,H,H],[X,Y]):- X\=H,Y\=H.

de_para([],[H2|T2],[H2],T2).
de_para([H|T1],[],T1,[H]).
de_para([H|T1],[H|T2],T1,[H,H|T2]) :- length([H | T2], X), X < 4.


pega_par([H|T],H,E) :- pega_um(T,E).
pega_par([_|T],P1,P2) :- pega_par(T,P1,P2).
pega_par([H|T],E,H) :- pega_um(T,E).
```

```
pega_um([H|_],H).
pega_um([_|T],X) :- pega_um(T,X).


troca([A|Js],A,NA,[NA|Js]) :- !.
troca([J|Js],A,NA,[J|N]) :- troca(Js,A,NA,N).

jogo(J, _) :- fim(J),  writeln(J),!.
%executa o jogo com as devidas regras.
jogo(J, N) :- N>0,
        pega_par(J,P1,P2),
        de_para(P1,P2,NP1,NP2),
        troca(J,P1,NP1,NJ1),
        troca(NJ1,P2,NP2,NJ2),
                N2 is N-1,
                jogo(NJ2, N2).
```

REGRAS

cheioVazio([]) :- !.
cheioVazio([X,X,X,X]).

Esta regra verifica se a lista passada é vazia ou possui 4 elementos iguais(X,X,X,X)

```
fim([]) :- !.
fim([H|T]) :- cheioVazio(H), fim(T).
```

Esta regra recebe uma lista de listas, que no caso cada copo seria uma dessas sublistas, e verifica cada uma dessas sublistas com a regra cheioVazio.

```
pega_par([H|T],H,E) :- pega_um(T,E).
pega_par([_|T],P1,P2) :- pega_par(T,P1,P2).
pega_par([H|T],E,H) :- pega_um(T,E).
```

Esta regra pega um par de copos da lista, pegando o primeiro copo da cabeça e fazendo comparações recursivamente até o fim da lista, percorrendo duas ordens, para frente, e para trás.

```
pega_um([H|_],H).
pega_um([_|T],X) :- pega_um(T,X).
```

Esta regra pega outro copo para futuras iterações.

```
troca([A|Js],A,NA,[NA|Js]) :- !.
troca([J|Js],A,NA,[J|N]) :- troca(Js,A,NA,N).
```

Esta regra efetua a troca na lista pelos novos copos já alterados com a nova configuração de bolinhas em uma nova lista.

de_para([],[H,H,H,H],_,_):- !,fail.
Se é comparado um copo vazio com um copo completo falha pois não necessita iteração.

de_para([],[H,H,H],_,_):- !,fail.
Se é comparado um copo vazio com um copo de 3 elementos iguais falha pois não necessita iteração.

de_para([],[H,H],_,_):- !,fail.
Se é comparado um copo vazio com um copo de 2 elementos iguais falha pois não necessita iteração.

de_para([],[_],_,_):- !,fail.
Se é comparado um copo vazio com um copo de 1 elemento falha pois não necessita iteração.

de_para([H,H,H,H],[],_,_):- !,fail.
Se é comparado um copo completo com um copo vazio falha pois não necessita iteração.

de_para([H,H,H],[],_,_):- !,fail.
Se é comparado um copo de 3 elementos iguais com um copo vazio falha pois não necessita iteração.

de_para([H,H],[],_,_):- !,fail.
Se é comparado um copo de 2 elementos iguais com um copo vazio falha pois não necessita iteração.

de_para([_],[],_,_):- !,fail.
Se é comparado um copo de 1 elemento com um copo vazio falha pois não necessita iteração.

de_para([H,X,Y],[H],[X,Y],[H,H]):- X\=H,Y\=H.
de_para([H],[H,X,Y],[H,H],[X,Y]):- X\=H,Y\=H.

Se tem uma lista de elementos, e uma lista com um único mesmo elemento do topo da outra lista retorna uma lista com esse mesmo elemento e outra com o resto, evitando trocas desnecessárias.

de_para([H,X,Y],[H,H],[X,Y],[H,H,H]):- X\=H,Y\=H.
de_para([H,H],[H,X,Y],[H,H,H],[X,Y]):- X\=H,Y\=H.

Se tem uma lista de elementos, e uma lista com um 2 mesmos elementos do topo da outra lista retorna uma lista com esses 2 mesmos elementos e outra com o resto, evitando trocas desnecessárias.

de_para([H,X,Y],[H,H,H],[X,Y],[H,H,H,H]):- X\=H,Y\=H.
de_para([H,H,H],[H,X,Y],[H,H,H,H],[X,Y]):- X\=H,Y\=H.

Se tem uma lista de elementos, e uma lista com um único mesmo elemento do topo da outra lista retorna uma lista com esses 2 mesmos elementos e outra com o resto, evitando trocas desnecessárias.

de_para([],[H2|T2],[H2],T2).
Se é comparado uma lista vazia com outra com elementos, retorna uma lista com a cabeça dessa lista de elementos e outra com a cauda dessa mesma lista.

de_para([H|T1],[],T1,[H]).
Se é comparado uma lista de elementos com uma vazia, retorna uma lista com a cauda dessa lista de elementos e outra com a cabeça dessa mesma lista.

de_para([H|T1],[H|T2],T1,[H,H|T2]) :- length([H | T2], X), X < 4.
Se o copo comparado tiver a mesma cabeça e o tamanho menor que 4, retorna a cauda da lista e ou com as duas cabeças.

```
jogo(J, _) :- fim(J),  writeln(J),!.
jogo(J, N) :- N>0,
        pega_par(J,P1,P2),
        de_para(P1,P2,NP1,NP2),
        troca(J,P1,NP1,NJ1),
        troca(NJ1,P2,NP2,NJ2),
                N2 is N-1,
                jogo(NJ2, N2).
```

Esta regra executa o jogo, seguindo a ordem de regras, primeiro pega o par que deve ser trocado, depois ele checa se pode ser trocado, caso possa, ele efetua a troca dos dois tudo na lista, e chama o jogo outra vez com a nova lista. O n é o limitador de comparações para não entra em loop em alguma comparação, e não prosseguir com o código.

Execução do código: no SWI coloque jogo(aListadeCopos,NumeroDeJogadas).

Testes:

Amarelo 0 Verde 3 azul claro 6

azul 1 Rosa 4

vermelho 2 Branco 5

Dificuldade 1:

Fase 1: [[0,1,0,2],[1,1,2,2],[0,0,2,1],[],[]] instantâneo

Fase 2: [[0,1,0,1],[1,2,2,0],[2,0,2,1],[],[]] instantâneo

Fase 3: [[2,0,2,0],[1,1,0,2],[0,1,2,1],[],[]] instantâneo

Dificuldade 2

Fase 1: [[1,2,2,3],[4,4,0,2],[0,1,3,1],[3,0,2,0],[1,3,4,4],[],[]] 3 seg

Fase 2: **[[1,0,2,2],[0,2,4,1],[3,4,0,2],[1,3,4,1],[4,3,3,0],[],[]]** instantâneo

Fase 3: **[[4,4,2,0],[3,2,1,4],[3,1,2,3],[0,1,3,4],[0,1,2,0],[],[]]** instantâneo

Dificuldade 3

Fase 1: **[[2,4,3,3],[5,3,6,6],[5,0,0,1],[3,6,2,4],[5,2,5,6],[0,1,4,1],[2,4,1,0],[],[]]** +10 minutos

Fase 2: **[[4,6,6,2],[5,2,0,2],[1,2,6,1],[1,0,0,3],[5,1,4,5],[4,6,3,5],[3,0,4,3],[],[]]** +10 minutos

Fase 3: **[[6,1,2,1],[5,3,5,2],[3,4,2,3],[4,0,0,3],[2,6,6,1],[5,4,5,4],[0,6,1,0],[],[]]** +10 minutos