Relatório do Trabalho Prático
Programação III

Trabalho realizado por:
Diogo Carreiro nº48729
Rodrigo Alves nº48681

Docente:
Professor Salvador Abreu

Introdução

Neste trabalho foi-nos proposto a realização de um programa que permite aceitar jogadas de um jogo em xadrez, numa das notações seguintes: Algébrica, Descritiva ou Postal. A notação utilizada neste programa foi a Algébrica. Basicamente o programa deve ler o input com as jogadas efetuadas durante um jogo, que está dentro de um ficheiro de texto, e mostrar o estado do tabuleiro ao fim de cada jogada. Para realizar este trabalho podíamos usar ambas as linguagens aprendidas na cadeira, Prolog e Ocaml. No nosso caso optámos por escolher a realização em Prolog, pois achámos que seria melhor a sua utilização, derivado do nosso maior conhecimento em relação a Ocaml.

Organização do Código

Para iniciarmos um jogo de xadrez, em primeiro lugar devemos definir as posições de cada peça no tabuleiro. Para isso criámos o predicado **init_b/0**, que insere as posições das peças inicialmente no tabuleiro. Assim utilizámos o predicado **assertz/1**, este predicado serve para atualizar as posições no fim de cada jogada sendo que o **retract/1** serve para eliminar a posição que foi alterada.

```
% Define initial positions of pieces
:- dynamic(piece/3).

init_b :-
    retractall(piece(_,_,_)),
    assertz(piece(wr,a,1)),
    assertz(piece(wr,h,1)),
    assertz(piece(wn,b,1)),
    assertz(piece(wn,g,1)),
    assertz(piece(wb,c,1)),
    assertz(piece(wb,f,1)),
    assertz(piece(wq,e,1)),
    assertz(piece(wk,d,1)),


    assertz(piece(wp,a,2)),
    assertz(piece(wp,b,2)),
    assertz(piece(wp,c,2)),
    assertz(piece(wp,d,2)),
    assertz(piece(wp,e,2)),
    assertz(piece(wp,f,2)),
    assertz(piece(wp,g,2)),
    assertz(piece(wp,h,2)),

    assertz(piece(bp,a,7)),
    assertz(piece(bp,b,7)),
    assertz(piece(bp,c,7)),
    assertz(piece(bp,d,7)),
    assertz(piece(bp,e,7)),
    assertz(piece(bp,f,7)),
    assertz(piece(bp,g,7)),
    assertz(piece(bp,h,7)),

    assertz(piece(br,a,8)),
    assertz(piece(br,h,8)),
    assertz(piece(bn,b,8)),
    assertz(piece(bn,g,8)),
    assertz(piece(bb,c,8)),
    assertz(piece(bb,f,8)),
    assertz(piece(bq,e,8)),
    assertz(piece(bk,d,8)).
```

O predicado **piece/3** serve para definir uma dada peça que está localizada numa determinada posição do tabuleiro.

Depois de definidas as posições das peças no tabuleiro, este é apresentado no terminal. Este processo é feito utilizando o predicado **print_board/0.**

```
% Print Board
print_board :-
    findall(piece(Piece, Col, Row),
    piece(Piece, Col, Row), Board),
    print_board(Board).

print_board(Board) :-
    write('  a  b  c  d  e  f   g  h'), nl,
    print_board(Board,8).

print_board(_,0) :- !.

print_board(Board,Row) :-
    Row1 is Row - 1,
    write(Row), write(' '),
    char_code('a',X),
    print_row(Board,Row,X),
    nl,
    print_board(Board,Row1).


print_row(_,_,H) :- char_code('i',H),!.

print_row(Board,Row,Col) :-
    from_char_code(Col,C),
    (   member(piece(Piece,C,Row),Board)
    ->  write(Piece), write(' '),
        char_code(C,Col),
        Col1 is Col + 1,
        print_row(Board,Row,Col1)
    ;   write('.  '),
        Col1 is Col + 1,
        print_row(Board,Row,Col1)
    ).
```

Neste predicado faz se uma procura de todas as peças que estão introduzidas no tabuleiro, utilizando o predicado **findall/1.** De seguida são escritas no terminal as letras de “a” até “h” que representam as colunas do tabuleiro. Posteriormente irá ser feita uma averiguação do tabuleiro linha a linha de modo a verificar qual é a peça numa dada posição da linha que está a ser averiguada. Se não existir nenhuma peça o predicado irá mostrar um “.”, caso contrário mostra a representação da peça correspondente.

Após o predicado anteriormente explicado ser terminado com sucesso o programa irá fazer a leitura do ficheiro de texto, que inclui todas as jogadas de um determinado jogo.

O predicado responsável por esta ação é o **read_file_and_splitline/1**. Basicamente este predicado abre o ficheiro para leitura, o qual vai ser lido linha a linha, sendo que essa linha vai ser traduzida para uma gramática criada pelo grupo, pois nesta etapa houve dificuldade na obtenção dos dados provenientes do ficheiro, sendo por isso útil a criação da gramática ***type***.

É a partir desta gramática que obtemos as jogadas, tanto das peças brancas como das pretas, bem como o resultado do jogo quando acaba.

```
% (Gramatic) Types of Strings
type(movimentos(X,Y)) --> jogada_branca(X), " " , jogada_preta(Y).
type(movimentos(X,R)) --> jogada_branca(X), " ", resultado(R).
type(movimentos(X,Y,R)) --> jogada_branca(X), " " , jogada_preta(Y), " " , resultado(R).
```

Aqui está o código relativo à nossa gramática **type/1.** Nesta imagem é possível observar como é feita a divisão da string extraída do ficheiro de texto a partir do predicado **read_file_and_splitline/1**.

Cada jogada pode ser definida de diferentes maneiras, pois há diferentes formas de representar um movimento, ou operação de uma peça. As diferentes jogadas estão especificadas a partir da gramática **jogada_branca** e **jogada_preta**, sendo que a diferença está na ordem em que a jogada está na linha (as jogadas das peças brancas são sempre representadas em primeiro lugar na linha do ficheiro). Neste trabalho é feita a leitura de todas as jogadas, mesmo daquelas que possuem erros na sua forma, ou seja, não estão de acordo com a notação algébrica.

```
% (Gramatic) jogada_branca
jogada_branca(jb(P,L1,S1,C,L2,S2)) --> f(P),l(L1),signal(S1),c(C),l(L2),signal(S2). %R2xb8+
jogada_branca(jb(P,C1,S1,C2,L,S2)) --> f(P),l(C1),signal(S1),c(C2),l(L),signal(S2). %Rbxb8+
jogada_branca(jb(P,L1,C2,L2,S)) --> f(P),l(L1),c(C2),l(L2),signal(S).              % N3d2+ / N3d2#       error
jogada_branca(jb(P,C1,C2,L,S)) --> f(P),c(C1),c(C2),l(L),signal(S).                % Nbd2+ / Nbd2#       error
jogada_branca(jb(P,S1,C,L,S2)) --> f(P),signal(S1),c(C),l(L),signal(S2).           % Nxf6+ / Nxf6#
jogada_branca(jb(P,C,L,S)) --> f(P),c(C),l(L),signal(S).                           % Nf6+ / Nf6#
jogada_branca(jb(C1,S1,C2,L,S2)) --> c(C1),signal(S1),c(C2),l(L),signal(S2).       % dxf6+ / dxf6#
jogada_branca(jb(C,L,S)) --> c(C),l(L),signal(S).                                  % e6+ / e6#
jogada_branca(jb(P,L1,S,C2,L2)) --> f(P),l(L1),signal(S),c(C2),l(L2).              % N1xf6               error
jogada_branca(jb(P,C1,S,C2,L)) --> f(P),c(C1),signal(S),c(C2),l(L).                % Ndxf6               error
jogada_branca(jb(P,S,C,L)) --> f(P),signal(S),c(C),l(L).                           % Nxf6
jogada_branca(jb(C1,S,C2,L)) --> c(C1),signal(S),c(C2),l(L).                       % dxf6
jogada_branca(jb(P,L1,C2,L2)) --> f(P),l(L1),c(C2),l(L2).                          % N3d2                error
jogada_branca(jb(P,C1,C2,L)) --> f(P),c(C1),c(C2),l(L).                            % Nbd2                error
jogada_branca(jb(P,C,L)) --> f(P),c(C),l(L).                                       % Nf6
jogada_branca(jb(C,L)) --> c(C),l(L).                                              % e6
jogada_branca(jb(Cast)) --> cast(Cast).                                            % 0-0 / 0-0-0
```

```
% (Gramatic) jogada_preta
jogada_preta(jp(P,L1,S1,C,L2,S2)) --> f(P),l(L1),signal(S1),c(C),l(L2),signal(S2). %R2xb8+
jogada_preta(jp(P,C1,S1,C2,L,S2)) --> f(P),l(C1),signal(S1),c(C2),l(L),signal(S2). %Rbxb8+
jogada_preta(jp(P,L1,C2,L2,S)) --> f(P),l(L1),c(C2),l(L2),signal(S).               % N3d2+ / N3d2#       error
jogada_preta(jp(P,C1,C2,L,S)) --> f(P),c(C1),c(C2),l(L),signal(S).                 % Nbd2+ / Nbd2#       error
jogada_preta(jp(P,S1,C,L,S2)) --> f(P),signal(S1),c(C),l(L),signal(S2).            % Nxf6+ / Nxf6#
jogada_preta(jp(P,C,L,S)) --> f(P),c(C),l(L),signal(S).                            % Nf6+ / Nf6#
jogada_preta(jp(C1,S1,C2,L,S2)) --> c(C1),signal(S1),c(C2),l(L),signal(S2).        % dxf6+ / dxf6#
jogada_preta(jp(C,L,S)) --> c(C),l(L),signal(S).                                   % e6+ / e6#
jogada_preta(jp(P,L1,S,C2,L2)) --> f(P),l(L1),signal(S),c(C2),l(L2).               % N1xf6               error
jogada_preta(jp(P,C1,S,C2,L)) --> f(P),c(C1),signal(S),c(C2),l(L).                 % Ndxf6               error
jogada_preta(jp(P,S,C,L)) --> f(P),signal(S),c(C),l(L).                            % Nxf6
jogada_preta(jp(C1,S,C2,L)) --> c(C1),signal(S),c(C2),l(L).                        % dxf6
jogada_preta(jp(P,L1,C2,L2)) --> f(P),l(L1),c(C2),l(L2).                           % N3d2                error
jogada_preta(jp(P,C1,C2,L)) --> f(P),c(C1),c(C2),l(L).                             % Nbd2                error
jogada_preta(jp(P,C,L)) --> f(P),c(C),l(L).                                        % Nf6
jogada_preta(jp(C,L)) --> c(C),l(L).                                               % e6
jogada_preta(jp(Cast)) --> cast(Cast).                                             % 0-0 / 0-0-0
```

```
% (Gramatic) resultado
resultado(branca_vence) --> "1-0".
resultado(preta_vence) --> "0-1".
resultado(empate) --> "1/2-1/2".
```

Ao utilizar a gramática anteriormente referida é possível ler o conteúdo de uma certa jogada. No nosso trabalho o que sai em X e em Y no predicado **movimentos (X, Y)** é um predicado do tipo **jb/1** ou **jp/1.** Estes predicados vão servir de auxílio para o predicado **extract_caracters_b/1**, e **extract_caracters_p/1.** Nestes predicados anteriormente referidos é possível saber qual a jogada efetuada, e saber em certos casos sobre que peça é feita.

```
% Read file
read_file_and_splitline(Filename) :-
    open(Filename, read, Stream),
    set_input(Stream),
    read_lines(movimentos(X,Y)),
    extract_characters_b(X),
    write(' '),
    extract_characters_p(Y),
    nl,
    read_file_and_splitline.


read_file_and_splitline :-
    read_lines(movimentos(X,Y)),
    extract_characters_b(X),
    write(' '),
    extract_characters_p(Y),
    nl,
    read_file_and_splitline.

read_lines(X) :-
    gets(L),
    phrase(type(X), L, []),!.
```

Uma vez que não foi possível realizar as validações das jogadas lidas devido aos problemas encontrados até esta etapa, o que seria implementado de seguida seria um predicado que dada uma certa jogada recebida iria verificar se estava de acordo com alguma jogada da nossa gramática. Caso não estivesse não iria ser contabilizada e iria ser criado um contador que, para cada cor contasse o número de erros. Ao fim de 3 erros efetuados era automaticamente atribuída a vitória ao outro jogador, sendo que o jogo terminava de imediato. Se a jogada fosse validada, ou seja, de acordo com as regras da notação, esta seria executada sobre uma peça que se encontra no tabuleiro. Após a atualização do estado do tabuleiro iria ser novamente chamado o predicado print_board para mostrar o tabuleiro com as alterações.

Conclusão

Concluindo, infelizmente não conseguimos alcançar o objetivo final deste trabalho, porém neste trabalho foram atingidos alguns requisitos pedidos. Contudo a realização do mesmo permitiu-nos aumentar não só o conhecimento da linguagem Prolog, bem como o seu comportamento, uma vez que esta linguagem foi aplicada num contexto diferente do habitual. Para além disso deu nos uma noção da complexidade de um jogo de Xadrez, uma vez que este apresenta uma extensa variedade de possibilidades de jogada.

Ao longo da realização do trabalho tivemos algumas dificuldades, o que não nos permitiu completar o código da forma que idealizávamos. Alguns desses problemas são: a leitura de um ficheiro e a utilização desse conteúdo para exibir a jogada correspondente, a conversão do código ASCII para caracter e vice-versa, encontrar todas as representações introduzidas nos ficheiros de texto e a validação de uma jogada.