Marvin Oliver Schneider

Turno da manhã

Categoria: 1

Manual do Sistema: Monarch

Inteligência Artificial no Exemplo de um Programa de Xadrez

Centro Universitário Positivo
Curitiba
21/10/2000

PROFESSORES ORIENTADORES

Foram professores orientadores deste projeto:

Prof. Fábio Vinícius Binder

Prof. Carlos Marcelo Pedroso

Prof. Eduardo Hruschka

Prof. Estevam Hrschuka

Prof. Francisco Kantek

Prof. Luiz Roberto Baracho Rocha

Prof. José Augusto Niviadonsky

Prof. Maurício Schafranski

Prof. Parahuari Branco

Prof. Pedro Kantek

DEDICATÓRIA

Eu dedico este projeto a minha esposa Alessandra que me incentivou muito a fazer o curso de informática e sempre me apoiou.

AGRADECIMENTOS

Gostaria de agradecer meu pai Prof. Dr. Dr. Hans Joachim Schneider, professor de direito penal e criminologia, que sempre será um ídolo para mim como cientista e pesquisador e minha mãe Hildegard Schneider pela paciência e boa vontade.

Com relação ao projeto agradeço especialmente para a ótima orientação do Prof. M.Sc. Fábio Vinícius Binder e aos Srs. Élcio Holzmann e Marcelo Razzolini, analistas de sistema da gedas do Brasil, que fizeram os *beta-testes* do sistema e ajudaram a corrigir erros ortográficos nos manuais.

SUMÁRIO

1. INTRODUÇÃO

Xadrez é um dos jogos mais antigos do mundo. Existem várias lendas e opiniões sobre o surgimento do jogo. Na metade do sétimo século o jogo é mencionado pela primeira vez em um documento escrito. Escavações em Afrisab na Índia dão margem a teorias que o jogo foi inventado no sexto século na Índia com o nome "Tschaturanga".

Da Índia o jogo chegou na China e na Pérsia, onde ele foi modificado um pouco e de lá comerciantes trouxeram ele para o oeste. Somente no século onze ele alcançou a Europa onde as regras foram novamente modificadas com a finalidade de deixar ele mais popular.

Os primeiros livros sobre o jogo são do século quinze.[1]

2. DESCRIÇÃO DO AMBIENTE

O Xadrez jogado por computadores tem uma história muito mais longa do que a história dos computadores modernos. A primeira tentativa de se criar uma máquina de Xadrez foi feita em 1770 pelo Barão von Kempelen no palácio do rei em Viena. Ele elaborou um aparelho chamado "o turco" completamente mecânico.

A primeira máquina que realmente jogava foi feita pelo eletromecânico Leonardo Torres y Quevedo em 1890 na Espanha.

Enquanto pessoas muito conhecidas da "computação" da época fizeram suas contribuições para o desenvolvimento de algoritmos e máquinas[2] os primeiros conceitos fundamentais dos programas atuais foram elaborados nos anos 50 do século passado.

[1] Veja Gerhard Hund - „Die Ursprünge des Schachspiels" e Gerhard Hund - „Geschichte des Schachspiels"
[2] envolvidos: Alan Turing, Konrad Zuse, Tihamer Memes

O primeiro campeonato entre computadores foi feito no meio dos anos 60 nos Estados Unidos.

Em 1968, Levy aposta que nenhum computador vai conseguir ganhar dele nos dez anos subseqüentes e após várias prorrogações ele perde somente em 1988 contra Deep Thought – o computador de Xadrez *top de linha* da época.

O computador atualmente mais potente para jogar Xadrez se chama de "Deep Blue". Ele até conseguiu ganhar de Kasparov.[3]

O programa "Monarch" se insere na linha dos programas comuns para jogar Xadrez em computadores pessoais. Por não contar com o mesmo hardware dos programas mais poderosos deste momento nem com o mesmo tempo disponível de desenvolvimento, ele tem limitações. Porém o programa é feito com um algoritmo especial e, em termos, novo.

3. DESCRIÇÃO DO PROJETO

3.1. DESCRIÇÃO GERAL DO PROJETO

O projeto "Monarch" é um programa de Xadrez simples e amigável com a possibilidade de escolher vários jogadores virtuais para treinar ou o modo padrão para jogadores mais avançados e campeonatos.

Objetivo geral do produto "Monarch" é tentar reencontrar o prazer em jogar Xadrez contra um programa, deixar o programa pedagógico de tal maneira que mesmo um iniciante pode aprender algo para jogos posteriores contra adversários humanos. O programa é criado de tal forma a atender as necessidades da maioria dos jogadores, os jogadores comuns, que já se

cansaram de sempre perder ou suportar a monotonia de ver sempre as mesmas jogadas dos programas existentes. Com isto o objetivo é também alcançar jogadores novos que talvez não tenham a possibilidade de treinar por falta de companheiros com o mesmo nível.

Considerando que existem vários programas no *Public Domain*[4] que jogam Xadrez razoavelmente bem, sugere-se oferecer o produto ao usuário grátis num site de vários outros programas de Xadrez. Não existe restrição de usuário e além de ser um projeto com a finalidade de desenvolver um algoritmo completo de Xadrez, ele também deve ser utilizado para popularizar o jogo – assim melhor oferecido sem cobrar o custo de desenvolvimento.

O programa "Monarch" tem as seguintes características:

3.1.1. Adversários Simulados

Pode se escolher entre 8 adversários simulados pelo computador. Cada um deles tem um caráter, ou seja, uma maneira muito especial de jogar. De propósito são implantadas possibilidades a erros e limitações para cada um deles. Isto se faz necessário porque os computadores tem geralmente uma maneira muito mecânica de jogar que se diferencia da maneira de humanos jogarem em vários aspectos. Esta característica de programas comuns diminui consideravelmente o prazer do jogo para muitos jogadores.

O programa – dependendo das habilidades do usuário – neste modo nem sempre ganhará. Cada um dos jogadores simulados tem um nível diferente e o usuário do programa pode identificar assim o progresso dele. Finalmente os pontos fracos dos jogadores virtuais são diferentes, treinando aspectos diferentes do jogo.

[3] veja Myrko Thum

3.1.2. Modo padrão

Pode-se selecionar operar o programa também no modo dos outros programas de Xadrez existentes no mercado. Neste modo ele evita erros na jogada da melhor maneira possível e tenta ganhar o jogo de qualquer maneira.

O algoritmo foi mantido o mais enxuto possível, somente utilizando recursividade quando for absolutamente necessário.

3.1.3. Tutorial

Foi implantado um tutorial para explicar os princípios básicos do Xadrez. Entre as regras também são abordadas as táticas básicas para o iniciante poder começar seu jogo.

3.1.4. Três línguas

Com a finalidade de alcançar mais usuários ainda e fazer testes do código com o maior número de pessoas possível, o projeto é implementado a disponibilizar três línguas de operação: português, alemão e inglês.

3.2. MACRO-FUNÇÕES

[4] Public Domain = Domínio Publico, isto é, os programas podem ser copiados e distribuídos livremente

3.2.1. Iniciar o jogo conforme modo escolhido

Esta função inicializa o jogo, o tabuleiro e - informando a configuração atual - dará partida no processo de jogar.

3.2.2. Ler/Salvar jogo

Pode-se salvar uma posição para depois continuar a jogar.

3.2.3. Tutorial

O tutorial explica as regras e táticas básicas. Ele é criado para jogadores iniciantes que não sabem ainda como jogar e muito menos ganhar.

3.2.4. Configuração

O adversário pode ser escolhido e também a língua de operação.

3.2.5. Fazer a melhor jogada possível

O algoritmo central do jogo divide – dependendo do número de peças no tabuleiro e a posição das peças – o jogo em fases. Isso é importante para determinar a tática e estratégia utilizada. São fases do jogo: *Começo, Batalha Central, Batalha Estratégica, Finalização*. Na fase do começo é utilizada uma rotina de desenvolvimento. Nas outras fases são consideradas – ou não dependendo da situação – influência no centro, trocas vantajosas, liberdade de

movimento, garfos, bloqueios e outras construções táticas básicas. Sendo algo que gaste muito tempo de máquina, o mate é buscado mais no fim do jogo do que no começo.

3.3. LIMITES, RESTRIÇÕES E ABRANGÊNCIA

O desenvolvimento do programa foi dividido em 4 fases. Somente a primeira fase foi definida como obrigatória para o projeto final:

3.3.1. Implementação do algoritmo básico

Esta fase - conforme a proposta - foi definida como obrigatória para o projeto final. Nesta fase foi implantado o algoritmo básico dando possibilidade ao programa de ganhar contra um iniciante de Xadrez, ou seja, alguém que conheça as regras e algumas técnicas muito básicas de se jogar.

3.3.2. Implementação dos jogadores e melhorias no algoritmo

Esta fase é um opcional conforme a proposta. Foram implementados os 8 jogadores e o algoritmo foi melhorado de tal maneira a ganhar de jogadores mais fortes e de outros programas.

3.3.3. Implementação do tutorial

Esta fase é considerada opcional para a entrega do projeto final. Nela o tutorial com a finalidade de ensinar Xadrez foi escrito.

3.3.4. Implementação das línguas

Na última fase opcional foram implementadas as várias línguas, isto é, a possibilidade de escolher entre português, alemão ou inglês.

3.3.4. Plataforma de Hardware e Software

O programa foi preparado para funcionar na família de computadores pessoais utilizando processadores Intel ou compatíveis nas configurações 486 ou superior e tendo Windows 95/98/NT instalado. Recomenda-se, porém, um Pentium com pelo menos 233 MHz para facilitar a operação do programa.

4. ANÁLISE

4.1. DESCRIÇÃO DO ALGORITMO UTILIZADO

Cada programa de jogo de Xadrez tem a obrigação - por incapacidade de recursos de hardware e de linguagens de programação – de achar um meio-termo entre um processamento detalhado e profundo e uma velocidade agradável de se utilizar o jogo. É necessário achar algoritmos inteligentes e otimizados para conseguir o máximo em qualidade de jogo.

No projeto Monarch foi criado um algoritmo de análise de tabuleiro em qual são utilizados vários sub-analisadores conforme os seguintes passos:

4.1.2. Primeiro passo: Analisador de Movimentos

Primeiramente o programa precisa ficar ciente das possibilidades de movimentos que existem numa certa posição. Isto consegue-se através de um analisador de movimentos e um *array*[5]. O analisador de movimentos varre o tabuleiro de Xadrez atual e analisa para cada figura do jogador que está na vez, quais movimentos esta figura pode fazer. Nisto está sendo levado em consideração que as diversas figuras executam movimentos diferentes e que certos movimentos dependem de momentos no jogo (*roque*[6] ou *en passent*[7]). Os resultados são gravados no *array* que foi escolhido por causa da segurança da memória e do fácil acesso. O *array* tem um comprimento de 140 posições, o que nunca será preenchido por ser o máximo

[5] campo com vários elementos que podem ser acessados diretamente
[6] troca de rei e torre
[7] forma de tomar um peão com um outro peão na hora dele passar dois campos para a frente

teórico de movimentos possíveis com todas as figuras de uma cor no tabuleiro. No *array* são salvos os movimentos e os pontos para cada um deles. O resultado é lido deste *array* no final da análise executando o movimento com a maior pontuação.

Para poder julgar se um movimento feito por um jogador humano é válido ou não, o analisador de movimentos do projeto Monarch acompanha a situação do tabuleiro a cada momento.

4.1.2. Segundo passo: Analisador Central

O analisador central analisa todos os movimentos a partir de uma posição dando pontos positivos e negativos, deletando movimentos que não devem ser executados e gerenciando as rotinas existentes conforme a fase do jogo.

4.1.2.1. Rotinas de Mate

Como primeiro passo o analisador central tenta achar um mate em um lance, o que – se for o caso – é executado sem passar pelo resto da análise, porque um "*mate em um*" significa vitória e é o objetivo do jogo.

Caso o mate não tenha sido encontrado, logo em seguida serão eliminados todos os movimentos que dão origem a mate em um lance do adversário, assim achando indiretamente movimentos possíveis de defesa. Se esta análise deixar um *array* de movimentos vazio, isto é, todos os movimentos dão a possibilidade do adversário ganhar, prossegue-se normalmente com os movimentos para as demais análises. Isto ocorre porque primeiramente não se

conhece o nível do jogador e pode ser que ele não ache o mate. Segundo, precisa-se destravar a posição pois com um *array* de movimentos vazio o computador não fará nenhuma jogada.

Para todas as rotinas que eliminam movimentos existe um algoritmo anti-trava para evitar que o computador em situações ruins não prossiga.

As rotinas que tratam as situações de mate em níveis mais avançados atribuem somente uma pontuação (altamente decrescente com o número de jogadas necessárias) aos movimentos que as possibilitam ou – caso o computador precise se defender de um mate eminente – eliminam as jogadas do array de movimentos.

4.1.2.2. Rotinas de Vantagem Material

Vantagem material é certamente um dos maiores objetivos durante um jogo de Xadrez, isto é, eliminar peças do inimigo protegendo as próprias peças e assim ganhando vantagem geral.

No projeto Monarch toda a análise estratégica é feita a partir da análise de vantagem material, ou seja, toda vantagem tem que se expressar em números materiais para valer a pena.

Jogadas que obviamente dão somente origem a desvantagem material são eliminadas no começo da análise. Movimentos vantajosos recebem uma pontuação conforme a peça conquistada. Para olhar também o outro lado da medalha calcula-se quais possibilidades de resposta o adversário tem, e eventuais problemas são tratados.

Finalmente, leva-se em conta as peças atacadas tentando tirá-las do ataque se não houver proteção suficiente.

4.1.2.3. Rotina de Desenvolvimento

Em vez de uma biblioteca de abertura, uma rotina de desenvolvimento trata com a abertura da posição. Os movimentos mais comuns recebem um pequeno adicional que na fase inicial pode fazer muita diferença.

4.1.2.4. Rotina Anti-Empate

O programa Monarch busca não empatar com o adversário. Os respectivos movimentos são eliminados do array de pontos. Possíveis empates originados pelo adversário porém não são eliminados pois assume-se que um empate neste caso é sinal que Monarch está perdendo, deixando-os vantajosos.

4.1.2.5. Rotinas tratando Ataque Duplo

Um ataque duplo é a situação de pelo menos dois ataques ao mesmo tempo, assim – na maioria dos casos – somente dando a possibilidade de salvar uma das duas peças. É então uma fonte de ganho material contra jogadores mais avançados que não estão "caindo nas armadilhas normais“.

O programa Monarch se defende contra este tipo de ataque com uma pontuação baixa, mas não eliminando os movimentos, pois nem todos os jogadores conhecem estas possibilidades.

Monarch também busca fazer ataques duplos, porém ele atribui uma pontuação menos positiva a estes movimentos do que aos ganhos "limpos“, ou seja, retirando uma figura do adversário diretamente.

4.1.2.6. Rotinas Estratégicas para as diferentes Fases do Jogo

Monarch usa rotinas estratégicas específicas para cada fase do jogo distinguindo entre quatro fases:

Abertura – nesta fase a posição inicial de batalha será criada dando a base aos ataques que devem trazer vantagem material ao jogador.

Batalha Central – as forças concentradas no centro do tabuleiro em certo momento explodirão. É o momento da batalha central que irá colocar um jogador em vantagem óbvia ou – se as forças forem equilibradas – montar um verdadeiro quebra-cabeça no tabuleiro.

Batalha Estratégica – a fase em qual cada um tenta invadir a área do outro, inicializar as possibilidades de mate e alcançar a vitória.

Finalização – com poucas peças no tabuleiro o único objetivo agora é buscar o mate e fechar o jogo.

Durante a *Abertura* Monarch busca principalmente o desenvolvimento das peças.

Na *Batalha Central* e *Batalha Estratégica*, Monarch tenta invadir a área do outro jogador abrindo espaço para movimentos, atacando as figuras de maior valor, criando linhas de defesa e assim esperando para conseguir alcançar uma situação que permita ganhar vantagem material.

Na finalização Monarch se aproxima cada vez mais do rei do adversário ganhando tempo dando Xeque e tentando conseguir controle de linhas para prevenir ataques.

4.1.2.7. Randomizador

Se Monarch não achar uma única melhor jogada ele escolhe randomicamente entre as melhores para variar o jogo.

4.1.2.8. Comportamento Especial em Situações de Troca

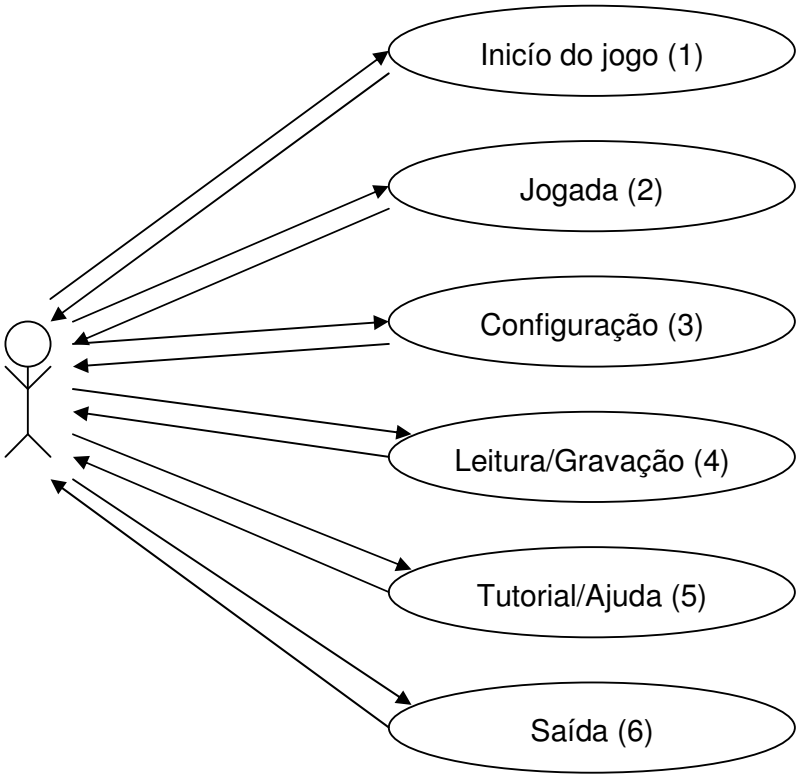
Monarch é programado para aceitar trocas com a finalidade de simplificar a posição no tabuleiro, assim ganhando vantagem. Se o adversário iniciar uma troca, Monarch é fixado nesta troca para não se interessar em outra coisa e assim produzir um problema estratégico.

4.2. IDENTIFICAÇÃO DA TÉCNICA DE ANÁLISE UTILIZADA

Sendo que o sistema não interage com ninguém fora o jogador humano e o sistema também não apresenta nenhuma interface muito sofisticada, escolheu-se a análise orientada a objetos assim dando ênfase à parte importante do programa: **o algoritmo**.

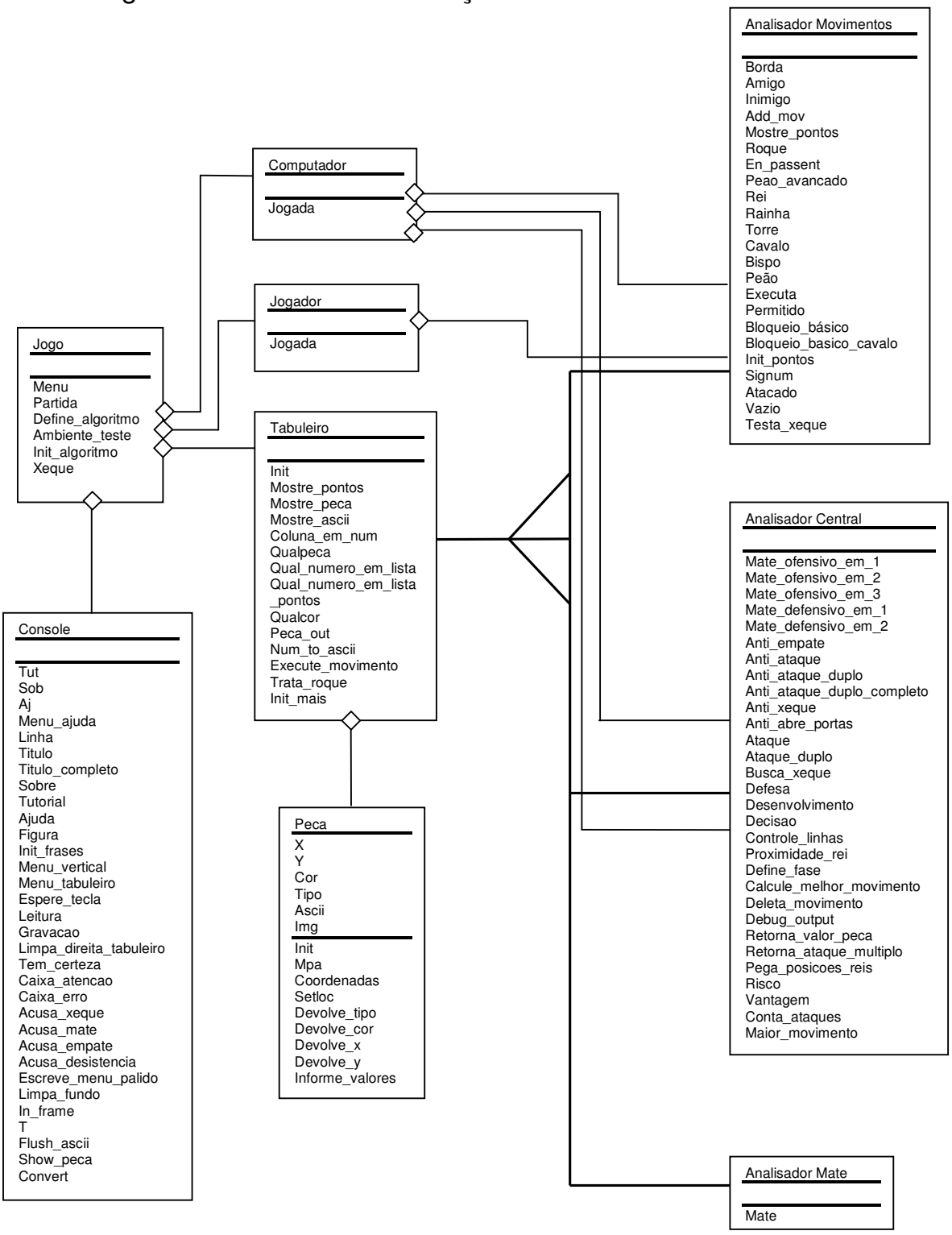
4.3. DIAGRAMAS ESPECÍFICOS

4.3.1. Diagrama de Casos de Uso e Descrição



Número do Caso	Ator	Nome do Caso	Descrição
1	Jogador	Início do jogo	Ocorre quando o jogador quiser iniciar um jogo com as configurações definidas anteriormente
2	Jogador	Jogada	Ocorre quando o usuário faz uma jogada contra o computador ou um outro jogador
3	Jogador	Configuração	O usuário pode configurar quem joga contra quem e a língua de operação
4	Jogador	Leitura/ Gravação	A posição do jogo atual pode ser gravada em disco pelo jogador e restaurada a qualquer momento do jogo
5	Jogador	Tutorial/Ajuda	O jogador pode ler o tutorial embutido no programa e obter informações da tela de ajuda escolhendo a opção no menu principal
6	Jogador	Saída	Ocorre quando o jogador der o comando para sair do programa

4.3.2. Diagrama de Classes e Descrição



Nome do elemento	Tipo	Descrição
Console	Classe	Classe para rotinas relacionadas à console
Tut	Método	Imprime “Tutorial” no cabeçalho
Sob	Método	Imprime “Sobre” no cabeçalho
Menu_ajuda	Método	Facilita o acesso aos menus mostrados na tela de ajuda
Aj	Método	Imprime “Ajuda” no cabeçalho
Linha	Método	Traça uma linha na tela
Titulo	Método	Mostra o logotipo do jogo
Titulo_completo	Método	Mostra o título completo do jogo na tela
Sobre	Método	Mostra informações gerais sobre o sistema no console
Tutorial	Método	Mostra o tutorial no console
Ajuda	Método	Mostra informações de ajuda no console
Figura	Método	Mostra um diagrama de Xadrez no tutorial
Init_frases	Método	Inicializa as frases operacionais nas três línguas
Menu_vertical	Método	Mostra na tela um tipo de menu verticalmente
Espere_tecla	Método	Interrompe o fluxo do programa para esperar o usuário apertar uma tecla qualquer

Leitura	Método	Lê uma posição de disco
Gravacao	Método	Escreve uma posição no disco
Limpa_direita_tabuleiro	Método	Deleta as informações no lado direito do tabuleiro
Tem_certeza	Método	Pergunta por segurança ao usuário se ele tem certeza
Caixa_atencao	Método	Mostra uma janela de mensagem para despertar a atenção do usuário
Caixa_erro	Método	Mostra uma mensagem de erro em uma janela
Acusa_xeque	Método	Acusa na tela que houve um Xeque
Acusa_mate	Método	Acusa na tela que houve um Mate
Acusa_empate	Método	Acusa na tela que houve um Empate
Acusa_desistencia	Método	Acusa na tela que um jogador desistiu
Escreve_menu_palido	Método	Escreve um menu na forma desativada
Limpa_fundo	Método	Limpa o fundo da tela
In_frame	Método	Coloca informações numa janela
T	Método	Tabulador (algumas posições para a direita)
Flush_ascii	Método	Mostra o tabuleiro em *ascii* após a leitura de disco
Show_peca	Método	Mostra uma peça no tabuleiro *ascii*
Convert	Método	Converte uma posição na string a ser lida do arquivo em um número
Jogo	Classe	Superclasse que constrói o ambiente

		total do programa
Menu	Classe	Gerencia o diálogo principal do sistema
Partida	Método	Faz o gerenciamento de uma partida
Define_algoritmo	Método	Função de teste para definir manualmente quais rotinas serão ligadas e desligadas
Ambiente_teste	Método	Chama o ambiente de teste de desenvolvimento
Init_algoritmo	Método	Inicializa o algoritmo do programa
Xeque	Método	Informa se houve um Xeque
Computador	Classe	Rotinas relacionadas ao comportamento do computador numa partida
Jogada	Método	Execução de uma jogada em modo padrão
Jogador	Classe	Classe do jogador humano
Jogada	Método	Execução de uma jogada do jogador humano
Tabuleiro	Classe	Classe relacionada ao tabuleiro virtual estabelecido
Init	Método	Inicializa o tabuleiro
Mostre_pontos	Método	Rotina para teste que mostra os pontos do array de pontos no console
Mostre_peca	Método	Mostra uma peça na tela
Mostre_ascii	Método	Mostra o tabuleiro inteiro na forma

		ascii
Coluna_em_num	Método	Converte a coluna (letra) em um número
Qualpeca	Método	Devolve a peça relacionada a duas coordenadas
Qual_numero_em_lista	Método	Mostra qual número uma peça tem na lista de movimentos
Qualcor	Método	Devolve a cor de determinada peça
Peça_out	Método	Mostra informações para uma peça na tela
Num_to_ascii	Método	Converte um número para o código *ascii*
Execute_movimento	Método	Executa um movimento em cima do tabuleiro virtual
Trata_roque	Método	Trata o *roque* na execução de um movimento
Init_mais	Método	Inicializa a matriz de direções
Peça	Classe	Classe da peça no tabuleiro
X	Atributo	Coordenada x da peça
Y	Atributo	Coordenada y da peça
Cor	Atributo	Cor da peça
Tipo	Atributo	Tipo da peça
Ascii	Atributo	Caracter *ascii* relacionada à peça
Img	Atributo	Imagem relacionada à peça (para versões futuras do programa)
Init	Método	Inicializa a peça
Mpa	Método	Mostra uma peça utilizando o

		código *ascii*
Coordenadas	Método	Checa se uma peça está em certas coordenadas
Setloc	Método	Define o lugar de uma peça
Devolve_tipo	Método	Devolve o tipo da peça
Devolve_cor	Método	Devolve a cor da peça
Devolve_x	Método	Devolve a coordenada x da peça
Devolve_y	Método	Devolve a coordenada y da peça
Informe_valores	Método	Mostra os valores da peça atual
Biblioteca	Classe	Classe relacionada à biblioteca de aberturas
Executa	Método	Executa a biblioteca de aberturas
Analisador_Movimento	Classe	O analisador de movimentos do sistema
Borda	Método	Retorna se chegou à borda do tabuleiro
Amigo	Método	Devolve se uma peça é amigo (da mesma cor)
Inimigo	Método	Devolve se uma peça é inimigo (de cor diferente)
Add_mov	Método	Adiciona um movimento na lista de movimentos
Mostre_pontos	Método	Mostra os pontos na lista de movimentos
Roque	Método	Analisa se o *roque* se aplica
En_passent	Método	Analisa se um *en passent* se aplica
Peao_avancado	Método	Analisa se há um peão avançado

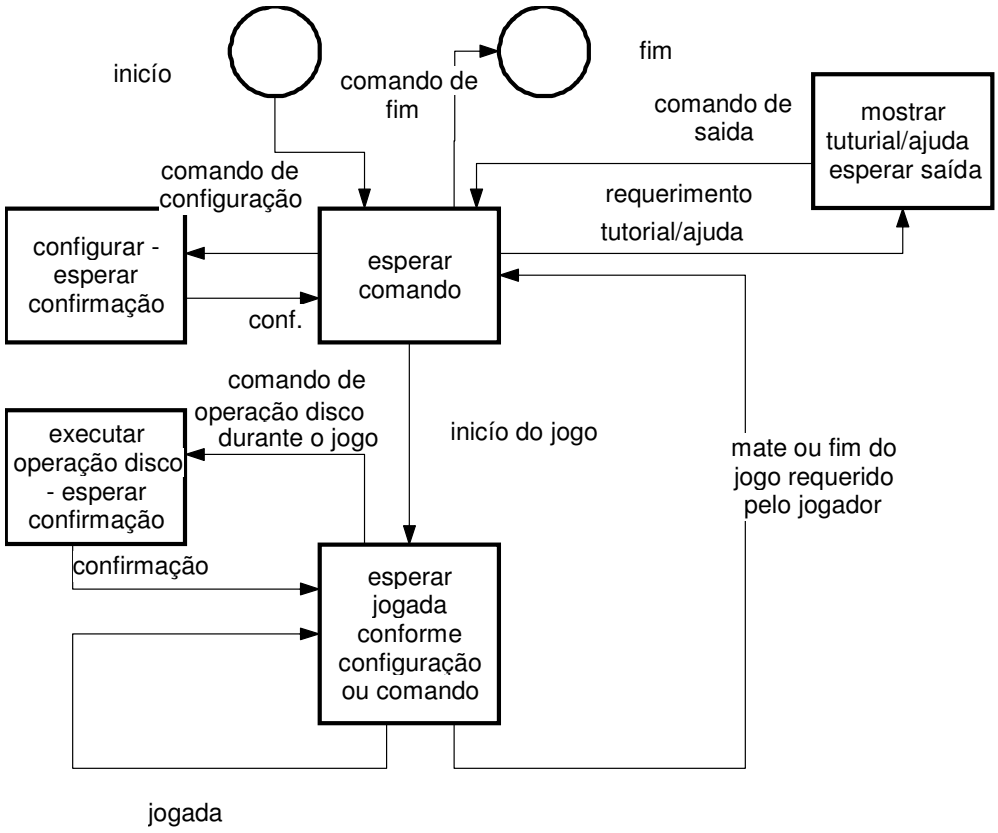
Rei	Método	Movimentos do rei
Rainha	Método	Movimentos da rainha
Torre	Método	Movimentos da torre
Cavalo	Método	Movimentos do cavalo
Bispo	Método	Movimentos do bispo
Peão	Método	Movimentos do peão
Executa	Método	Gerenciador do analisador de movimentos
Permitido	Método	Devolve se determinado movimento é permitido
Bloqueio_básico	Método	Devolve se determinada posição está bloqueada para movimentos
Bloqueio_basico_cavalo	Método	Devolve se determinada posição está bloqueada para movimentos do tipo "cavalo", isto é, pulando por cima de outras figuras
Init_pontos	Método	Inicializa a lista de pontos
Signum	Método	Devolve o valor da função *signo* (para alguns cálculos)
Atacado	Método	Informa se determinada posição está atacada
Vazio	Método	Informa se determinada posição está vazia
Testa_xeque	Método	Informa se o rei está em Xeque
Analisador_Mate	Classe	O analisador de mate direto do sistema
Mate	Método	Informa se um mate ocorreu

Analisador_Central	Classe	Gerencia o comportamento do computador em uma partida de Xadrez
Mate_ofensivo_em_1	Método	Busca o mate em 1 a partir da posição atual
Mate_ofensivo_em_2	Método	Busca o mate em 2
Mate_ofensivo_em_3	Método	Busca o mate em 3
Mate_defensivo_em_1	Método	Tenta impedir que o adversário dê mate em 1
Mate_defensivo_em_2	Método	Tenta impedir que o adversário dê mate em 2
Anti_empate	Método	Evita que o computador produza um empate
Anti_ataque	Método	Evita ataques e pontua execuções de jogadas vantajosas
Anti_ataque_duplo	Método	Tenta impedir que o adversário estabeleça um ataque duplo
Anti_ataque_duplo_completo	Método	Mesmo que anterior, mas serão checadas todas as possibilidades, não somente cavalo e peão
Anti_xeque	Método	Evita Xeques
Anti_abre_portas	Método	Tenta não possibilitar boas jogadas do adversário
Ataque	Método	Ataca o adversário
Ataque_duplo	Método	Tenta fazer ataques duplos
Busca_xeque	Método	Busca dar xeque
Defesa	Método	Defende as próprias peças de ataque

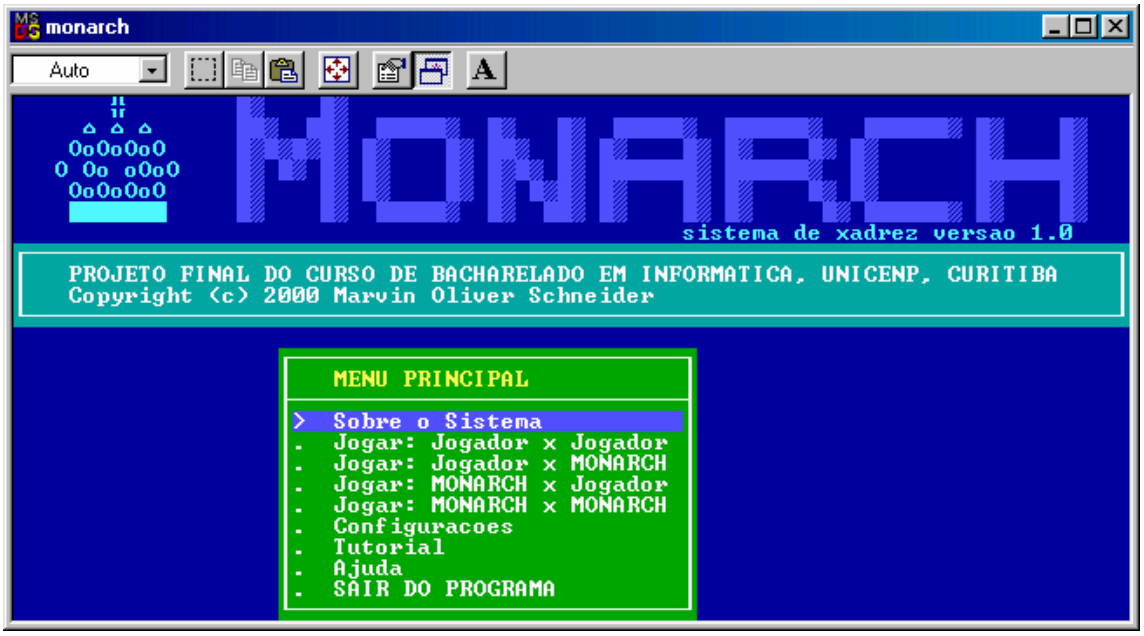
Desenvolvimento	Método	Desenvolve as peças principalmente na abertura
Decisao	Método	Decide a melhor jogada a base dos dados fornecidos
Controle_linhas	Método	Tenta estabelecer controles de linhas
Proximidade_rei	Método	Acha movimentos mais próximos ao rei do inimigo
Define_fase	Método	Define a fase do jogo
Calcule_melhor_movimento	Método	Calcula o melhor movimento (rotina mestre)
Deleta_movimento	Método	Deleta um movimento do array de movimentos
Debug_output	Método	Mostra o nome do analisador atual na tela para debug
Retorna_valor_peca	Método	Retorna o valor estratégico de uma peça
Retorna_ataque_multiplo	Método	Retorna se a partir de uma situação existem vários ataques múltiplos vantajosos
Pega_posicoes_reis	Método	Grava as posições dos reis em variáveis separadas
Risco	Método	Calcula o risco que um jogador corre deixando uma peça em determinado lugar
Vantagem	Método	Calcula a vantagem de um movimento

Conta_ataques	Método	Conta os ataques que existem em uma posição
Maior_movimento	Método	Devolve a pontuação do melhor movimento

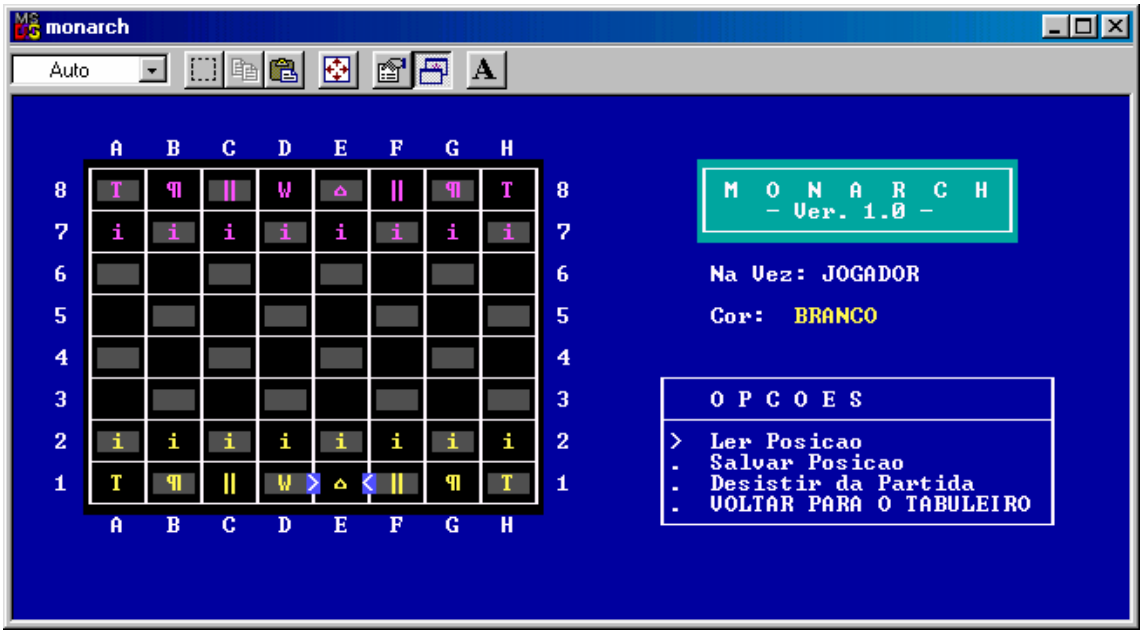
4.3.3. Diagrama de Transição de Estados



4.3.4. Modelo de Telas



A tela principal do jogo que utiliza – como todas as telas em quais o usuário pode interagir com o programa – uma estrutura de menu *ascii*.



A tela de jogadas – a única que utiliza uma estrutura bem diferente da tela anterior.

4.3.5. Estrutura de Banco de Dados

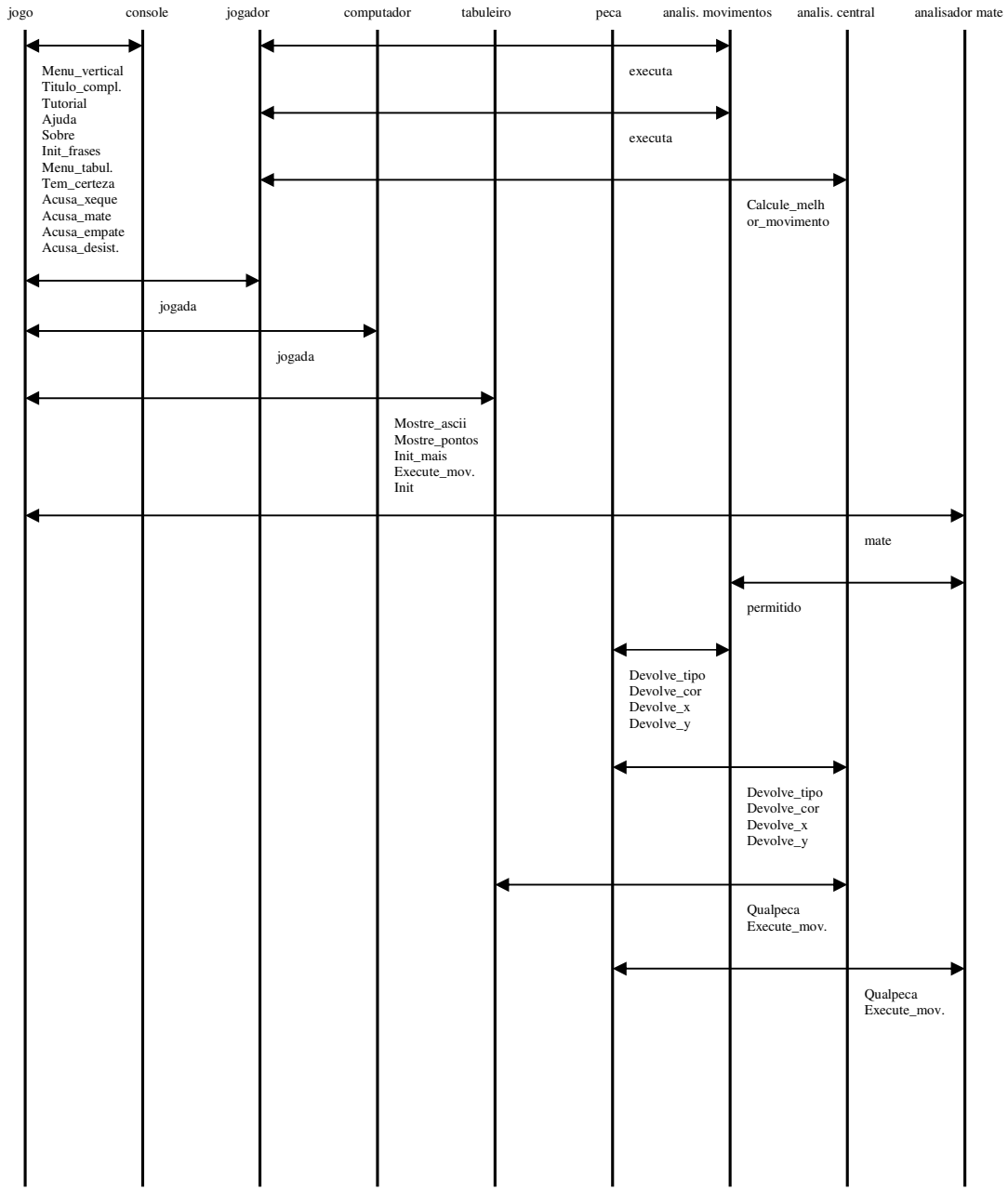
O sistema Monarch **não faz uso de um banco de dados**. As informações necessárias se encontram embutidas no código fonte.

Monarch é capaz de gravar a posição atual do tabuleiro em disco e restaurá-la do disco. O arquivo criado é um arquivo texto que grava 32 conjuntos de dados (um para cada peça) com a seguinte formatação:

Variável	Tipo	Uso
X	Texto (‘0’ a ‘9’)	Coordenada X da peça no tabuleiro interno
Y	Texto (‘0’ a ‘9’)	Coordenada Y da peça no tabuleiro interno
Cor	Texto (‘0’ a ‘9’)	Cor da peça (PRETO ou BRANCO)
Tipo	Texto (‘0’ a ‘9’)	Rei, Rainha, Bispo….

O código ‘9’ significa ‘INVÁLIDO’ e determina que a peça nesta posição do array de peças foi deletada.

4.3.6. Diagrama de Rastreamento de Eventos



5. REFERÊNCIAS BIBLIOGRÁFICAS

- HUND, Gerhard. *Geschichte de Schachspiels*. http://www.teleschach.de/forum/shistory
- HUND, Gerhard. *Die Ursprünge des Schachspiels*. http://buene.muenster.de/mauritz/projekte_in/Sport_98/Schach/GESCHICHTE.html
- LÓPEZ-ORTIZ, Alejandro. *Computer Chess – Past to Present*. http://www.cs.unb.ca/~alopez-o/divulge/chimp.html
- PFLEGER, Helmut et alii. *Zug um Zug – Schach für Jedermann 3 – Offizielles Lehrbuch des Deutschen Schachbundes zur Erringung des Königsdiploms*. Niedernhausen/Alemanha: Falken Verlag, 1989, 2a. Ed.
- THUM, Myrko. *Computerschach*. http://www.informatik.hu-berlin.de/~thum/chess/compschach/compschach.html