UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS GRADUAÇÃO EM INFORMÁTICA

FERRAMENTAS E MÉTODOS PARA APOIAR O ENSINO DE XADREZ NA FRONTEIRA ENTRE OS FUNDAMENTOS E A PERÍCIA

CURITIBA
2007

FRANCISCO MEIRA AGUIAR

FERRAMENTAS E MÉTODOS PARA APOIAR O ENSINO DE XADREZ NA FRONTEIRA ENTRE OS FUNDAMENTOS E A PERÍCIA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.

Orientador: Prof. Dr. Alexandre Ibrahim Direne

CURITIBA

2007

Dedico este trabalho a minha querida esposa
Dalva cujo amor, compreensão e apoio nunca
me faltaram.

AGRADECIMENTOS

Ao Prof. Dr. Alexandre Ibrahim Direne pela confiança depositada em nosso trabalho e também pelo entusiasmo, envolvimento e dedicação durante suas aulas e orientações, que estão sempre a servir de modelo para os seus aprendizes na arte do bem ensinar. Apresentou-nos com brilhantismo os caminhos já desbravados e os que estão para serem desbravados na empolgante área do conhecimento humano que são os STI, motivou-nos e orientou-nos para que superássemos os inúmeros obstáculos encontrados durante a realização deste trabalho. Nosso muito obrigado.

Ao corpo docente da UFPR que transmitiram com maestria seus ensinamentos e suas orientações que nos faltavam para o desenvolvimento desta dissertação. Aos colegas da turma de 2005 cuja ajuda e incentivo recebi e que quero agradecer, pena não caber o nome de todos eles neste texto.

Agradeço também aos meus pais – in memoriam -, meus irmãos, cunhados e sobrinhos que sempre me ensinaram muitas coisas. Em especial, agradeço a minha irmã Regina, a primeira mestra da família, que muito nos incentivou nesse trabalho e sempre teve uma palavra de carinho e incentivo para a educação e para os educadores.

Agradeço a minha esposa Dalva e as minhas filhas Mariana e Débora com a consciência que dei muito trabalho a vocês nesses anos todos, principalmente durante o desenvolvimento deste trabalho.

E por fim, mas não por último, agradeço a Deus esta Saúde, Paz e Felicidade que sempre me acompanharam e que, seguramente, foram de inestimável valor na realização desse empreendimento.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACT	-	Adaptive Control of Thought
ADL	-	Advanced Distributed Learning Initiative
API	-	Application Program Interface
ASCII	-	American Standard Code for Information Interchange
CACAREJE	-	Colaboração Alternada com Competição na Aprendizagem Referenciada por Jogos Educativos
CAI	-	Computer-Aided Instruction System
CAM	-	Content Aggregation Model
CEX	-	Centro de Excelência em Xadrez
CSS	-	Cascading Style Sheet
DoD	-	Department of Defense
FAE	-	Função de avaliação estática
FEN	-	Forsyth-Edwards Notation
HTML	-	HyperText Markup Language
IA	-	Inteligência artificial
ITS		Inteligent Tutoring System
LISP	-	List Processing
LMS	-	Learning Management System
LTM	-	Long-Term Memory
MELON	-	Meta-Language for Defining Operation Commands
PGN	-	Portable Game Notation
PHP	-	PHP: Hypertext Preprocessor
PIF	-	Package Interchange File
PROTEX	-	Projeto de Tipificação do Ensino de Xadrez
RTE	-	Run-Time Environment
SCO	-	Shareable Content Object
SCORM	-	Shareable Content Object Reference Model
SGA	-	Sistemas de Gerenciamento de Aprendizagem
SN	-	Sequencing and Navigation
STI	-	Sistemas Tutores Inteligentes
STM	-	Short-Term Memory
STR	-	Regra das sete marcas
UFPR	-	Universidade Federal do Paraná
UI	-	Unidade instrucional

RESUMO

Esta dissertação apresenta um conjunto de ferramentas e métodos que tem por objetivo ajudar mestres enxadristas no ensino de suas habilidades a leigos e iniciantes. A necessidade de tais artefatos partiu da constatação das vantagens do jogo de Xadrez no desenvolvimento do raciocínio lógico e em atividades lúdicas e, também, das dificuldades no seu ensino e na sua aprendizagem. Estas ferramentas e métodos foram então projetadas e prototipadas em computador para permitir a inclusão de unidades instrucionais organizadas em cursos, módulos e lições para o ensino de conhecimentos declarativos e, também, de exercícios que criem a chance de fixar o conhecimento do aprendiz por meio de reações altamente especializadas de um tutor automático que se originam de um módulo de diagnóstico. Uma pesquisa na literatura existente foi realizada em busca de subsídios para o projeto e verificou-se que só recentemente alguma pesquisa tem sido conduzida em STI - Sistemas Tutores Inteligentes que ensinem a jogar Xadrez. Todavia, essas pesquisas ainda carecem de testes práticos e de modelos do estudante e pedagógico que diagnostiquem as ações do aprendiz ao longo de toda a partida e promovam uma retroalimentação imediata às ações do aluno. O protótipo apresentado nesta dissertação incorpora conceitos da psicologia cognitiva, principalmente o ACT* desenvolvido por Anderson e seus colaboradores (1983,1984 e 1988), além dos conceitos referentes às características diferenciadoras entre peritos e novatos desenvolvidas por Lesgold (1989) e Direne (2001). Do trabalho de Anderson destacaram-se as regras de produção, o conhecimento declarativo e sua compilação que se materializaram, neste protótipo, na forma de currículo, comandos, heurística e uma retroalimentação imediata. Das contribuições de Legold e Direne destacou-se o conceito de que a perícia é formada por exposições a "casos exemplares" os quais se materializaram, neste protótipo, em funções de autoria de material de curso fornecido por um instrutor humano e a interpretação inteligente do material por meio do diagnóstico e da tutoria automática diante de estímulos fornecidos pelo aprendiz.

ABSTRACT

This dissertation describes one set of tools and methods which aims to help chess masters and grand masters to teach her/his skill to layman and learner. The necessity of these artefacts emerges of the finding out of the advantages of the chess game in the grow up of the logic reasoning and in ludics activity, and also on teaching and learning difficulties. So, this tools and methods are both designed and prototyped in a computer system to lend inclusion of instructionals units arranged in course, modules and lessons, which aims to teaching of declaratives knowledges and, also, at exercises to produce a chance of "to fix" the learner knowledge through reactions highly specialized of the automatic tutor initiated by one diagnostic module. A searching on the literature existent was realized and it was ascertained that only lately some searching was made about Inteligent Tutoring System (ITS) what teaching Chess game. However, these works yet need of pratic tests and of one student and pedagogic model which diagnostic the learner actions during the whole match and lend a imediate feed-back to learner actions. The prototype expose in this dissertation incorporate concepts of the cognitive psychology, mainly the ACT* developed by Anderson and his partners (1983,1984 e 1988), besides concepts relative the characteristcs that distinguish expert and novice developed by Lesgold (1989) and Direne (2001). From the firsts, it's emphasized the production rules, the declarative knowledge and its compilation which was materialized, in this prototype, in the shape of curriculum, commands, heuristic and imediate feed-back. From the seconds, it's emphatized the concept that expertise is formed by exposure to "exemplary cases" which was materialized, in this prototype, in functions of author courseware of materials made for humans tutor and a intelligent interpretation of this material through of one automatic diagnostic and tutoring from stimulus furnished to learner.

1. INTRODUÇÃO

1.1 O PROBLEMA CENTRAL

Estudos demonstram que jogos baseados em regras são importantes na educação pois desencadeiam reflexões nos aprendizes e proporcionam construções significativas do ponto de vista cognitivo (Piantavini, 1999). Além disso, jogos baseados em regras de fundo heurístico são mais eficazes que os jogos de regras baseados em ação (Bogatschov, 2001). O jogo de Xadrez, um jogo de regras de fundo heurístico é reconhecidamente de grande importância para o desenvolvimento do raciocínio lógico e como atividade lúdica. Esse fato tem gerado várias iniciativas no sentido de trazer o jogo de Xadrez para dentro da escola. Por exemplo, o estado do Paraná conta com o CEX – Centro de Excelência em Xadrez, que entre outras finalidades tem a de difundir o Xadrez nas escolas. Iniciativas mais avançadas como a de universalizar o ensino de Xadrez nas escolas esbarram em dificuldades como, por exemplo, a escassez de professores de Xadrez. O aumento da relação aluno/professor é uma forma de superar os problemas decorrentes dessa escassez. Esse aumento poderia ser obtido se a área de Informática, mais especificamente sua subárea Sistemas Tutores Inteligentes, disponibilizasse artefatos de software para apoio de mestres enxadristas na criação de material de ensino eletrônico focado nos fundamentos e na perícia inicial desse jogo. Um artefato desse tipo deve priorizar a aprendizagem de princípios ou fundamentos do jogo. Um desses princípios é, por exemplo, o ensino do movimento das peças. Na Fig. 1, apresenta-se uma interação possível de um sistema hipotético que ensina um importante princípio do jogo de Xadrez, o movimento do cavalo.

CAVALO

Os cavalos possuem um movimento particular bastante diferente das demais peças. Para simplificar, digamos que o cavalo anda uma casa como torre e uma casa como bispo. O cavalo é a única peça que salta sobre as outras.

Um cavalo na casa e5 pode ir para 8 casas diferentes (c4, c6, d3, d7, f3, f7, g4 e g6).

Quando o cavalo sair de uma casa preta irá parar em uma casa branca e vice-versa.

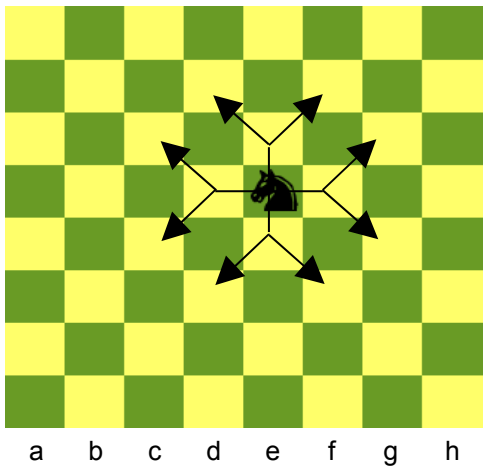

Fig. 1 – Exemplo de ensino de princípios em Xadrez – O Cavalo

Ainda no plano de fundamentos do jogo tais como regras básicas de movimento ou captura de peça, o Xadrez possui movimentos especiais como o roque, o "en passant" e a regra do 50º movimento que representam dificuldades adicionais para o iniciante. O roque é um movimento, tipicamente defensivo, que envolve duas peças ao invés de uma. Ele é executado com a movimentação do rei duas casas em direção à torre, e a torre, pulando o rei, ocupa a casa ao lado dele. Esse movimento só pode ser executado se: a) O rei e a torre envolvida nunca foram movimentados, b) Não há peças entre o rei e a torre, c) O rei não estiver em xeque, d) As casas por onde o rei passar não estiverem atacadas, e e) O rei ao rocar não terminar em xeque. O *en passant* (de passagem) é um movimento atípico que possibilita a um peão capturar outro peão quando este, de sua casa inicial, anda duas casas à frente e fica ao lado do peão adversário. O peão adversário pode então capturá-lo como se aquele tivesse andado somente uma casa a partir da sua casa inicial. A captura *en passant* deve ser executada imediatamente após o avanço do peão, pois se o jogador optar por outro movimento ele perde o direito à captura. A regra do 50º movimento permite que qualquer jogador reivindique o empate do jogo se nos últimos 50 lances não houver: a) A captura de alguma peça, ou b) O avanço de um peão.

Além da aprendizagem de conhecimentos de princípios, esse artefato de software deve oferecer ferramentas que apoiem o desenvolvimento de alguma perícia inicial. Pesquisadores das habilidades diferenciadoras entre peritos e novatos, como Lesgold (1989) e Direne (2001), identificaram que, resumidamente, o desenvolvimento de perícia se dá pela exposição a casos exemplares e não exemplares, e que a aprendizagem se dá por indução a partir desses casos. No jogo de Xadrez, um caso exemplar, por exemplo, é a "harmonia de peões", estudada por Hartmann (2005), que afirma que uma harmônica disposição dos peões é considerada vantajosa por peritos enxadristas. Outro exemplo de caso exemplar é o "domínio do centro" que considera o domínio das casas e4, e5, d4 e d5 como vantajosa pois as peças que se situam nessa região têm seu raio de ação maximizado (Tirado, A. e Silva W. 2005).

Na literatura especializada no ensino de Xadrez, a fronteira entre princípios e perícia tem sido coberta por procedimentos relativamente simples, os quais são organizados como breves desafios ao aprendiz para que ele atinja um determinado resultado. Normalmente, são apresentados tabuleiros, como o da Fig. 2, e desafia-se o aprendiz a determinar a seqüência de lances que levam uma dada cor de peça à vitória (xeque-mate). Todavia, esses procedimentos podem se tornar bastante complexos quando da sua representação em um Sistema Tutor Inteligente, principalmente quando sabe-se que para uma ação pedagógica efetiva esses exercícios precisam ser acompanhados de explicações ao aluno que sirvam como

retroalimentação imediata às suas ações. Pelos motivos expostos acima, a disponibilização de linguagens e ferramentas que permitam a um tutor organizar o seu material de curso de forma que possa ensinar princípios do jogo de Xadrez e apresentar exercícios na fronteira com a perícia é de muita importância. A apresentação de um protótipo de tal artefato é o ponto central desta dissertação.

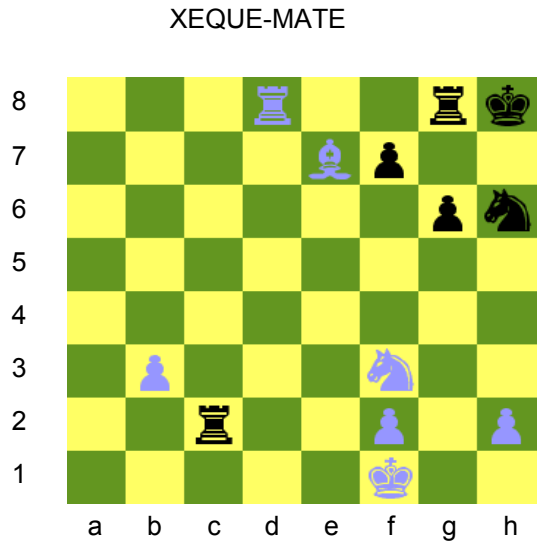

Fig. 2 – Exemplo de procedimento para ensinar perícia em Xadrez – As brancas dão Xeque mate em 2 lances

Esforços para o desenvolvimento de tutores inteligentes se desenrolam desde os anos setenta em vários domínios, tais como: aritmética, radiologia médica, álgebra, geometria e outros. Em pesquisas clássicas que visam dar uma ampla e profunda visão da área como os de Wenger (1987) e Murray (1999) os quais apresentam grande número de sistemas tutores inteligentes, em nenhum deles é abordado um STI - Sistema Tutor Inteligente para a área de domínio do jogo de Xadrez. Entretanto, pesquisas mais recentes apresentam sistemas tutores na área de domínio do Xadrez os quais, todavia, ou são focados na competição de heurística (Martineschen, 2006 e Feitosa, 2006), ou na identificação de harmonia das peças (Hartmann, 2005), ou em finais envolvendo Bispo e Peões (Gadwal, 2000).

Desses fatos conclui-se que o protótipo objeto deste trabalho possui aspectos originais tanto no seu objetivo mais geral – um STI para a aprendizagem de princípios do jogo de Xadrez – como no seu objetivo mais específico – diagnóstico automático do desempenho do aluno ao longo de toda a partida na fronteira com a perícia.

1.2 OBJETIVOS GERAIS

Este estudo apresenta o protótipo XadrEx – Expertise em Xadrez, um Sistema Tutor Inteligente na área de domínio do jogo de Xadrez, que possibilita a aprendizagem de habilidades desse jogo na fronteira entre os fundamentos e a perícia. Os objetivos gerais que foram alcançados são os seguintes:

a) Mestres enxadristas podem, através de uma linguagem intuitiva e de fácil aprendizado, descrever conhecimentos sobre princípios do jogo (na terminologia de Lesgold) para serem assimilados pelo aprendiz.

b) Mestres enxadristas podem também descrever conhecimentos já pertencentes ao estágio pericial (na terminologia de Lesgold) para serem assimilados pelo aprendiz de maneira prática.

c) Aprendizes podem "aprender fazendo" as habilidades ensinadas pelos mestres.

1.3 CONTEXTO DO PROJETO

Este projeto se insere no contexto do projeto PROTEX – Projeto de Tipificação do Ensino de Xadrez cujos objetivos são apresentados em Direne (2004). O projeto PROTEX, além de dar todo o suporte de hardware e software ao CEX – Centro de Excelência em Xadrez (www.cex.org.br) também compreende o apoio computacional destinado ao ensino de Xadrez nas escolas públicas brasileiras (promovido e financiado pelos Ministérios da Educação e do Esporte do Governo Federal). Aplica-se o conceito de Tecnologia Educacional no contexto de ferramentas baseadas em Software Livre, capazes de permitir tanto a autoria do material eletrônico de um curso quanto o aprendizado por experimentação e competição. A necessidade destas ferramentas surge da constatação de que a grande maioria dos sistemas para o ensino de Xadrez assume que o aprendiz humano é um especialista e não um iniciante. Este projeto já desenvolveu estimulantes pesquisas no seu âmbito de atuação. Hartmann (2005) desenvolveu um STI de nome brKChess que utiliza como motor inteligente o algoritmo da Inteligência Artificial subsimbólica denominado rede de Kohonen para identificar a harmonia de peças em um tabuleiro. Martineschen (2005) desenvolveu um STI de nome CACAREJE -Colaboração Alternada com Competição na Aprendizagem Referenciada por Jogos Educativos que permite implementar a alternância entre atividades competitivas e colaborativas no ensino de Xadrez através de registro e comparação dos resultados obtidas pela heurística de cada participante. Feitosa (2005) desenvolveu o front-end

para o sistema CACAREJE, uma linguagem de especificação de heurística chamada DHJOG a qual permite a formalização completa do conhecimento heurístico dos participantes.

2. TRABALHOS CORRELATOS

Neste capítulo apresenta-se uma resenha de trabalhos desenvolvidos na área de Sistemas Tutores Inteligentes e que exerceram forte influência nesta dissertação e fundamentaram, em grande parte, a pesquisa realizada. Inicialmente, conceitua-se os sistemas desenvolvidos no paradigma dos STI - Sistemas Tutores Inteligentes contrapondo-os aos seus antecessores CAI – Computer-Aided Instruction System. Depois, analisa-se uma teoria do currículo[1] e apresenta-se o modelo psico-cognitivo – ACT* - Adaptive Control of Thought devido a Anderson e seus colaboradores o qual suportou várias abordagens em STI. A seguir, aponta-se características diferenciadoras entre peritos e iniciantes tanto na Radiologia Médica quanto no Xadrez além de diferentes formas de diálogo tutorial. E, finalmente, apresenta-se trabalhos desenvolvidos em STI na área de jogos e também uma classificação para os Sistemas Tutores Inteligentes.

2.1 SISTEMA TUTOR INTELIGENTE

Um Sistema Tutor Inteligente se destaca pela sua capacidade de comunicar o conhecimento (Wenger, 1987). Para atingir esse objetivo esses sistemas são construídos através da modelagem e articulação de quatro elementos básicos:

1) Modelo do domínio: - Representa a fonte do conhecimento a ser comunicado. Esse conhecimento pode ser registrado na forma de um currículo ou incorporado no modelo. Em um STI típico a apresentação desse conhecimento deve sofrer variação conforme o comportamento do estudante.

2) Modelo do estudante: - Registra aspectos que tem repercussão para o desempenho e aprendizado do aprendiz. O sistema obtém essas informações a partir do diagnóstico do aluno.

3) Modelo pedagógico: - Representa a maneira como o conhecimento será comunicado. Num nível global essas decisões afetam a seqüência dos assuntos a serem apresentados. Num nível local determina intervenções do sistema tais como: a) Guiar o aluno na atividade, b) Explanar o assunto relacionado à atividade, ou c) Remediar uma ação errônea do aluno.

[1] É comum encontrarmos também na forma latina: Curriculum. Usamos a forma adotada no dicionário Houaiss: Currículo.

4) Interface: - Representa a forma como a comunicação se processa. Aspectos como : Qual a linguagem do estudante na interação com o sistema?, Qual a linguagem do tutor usada para registrar o conhecimento?, Qual a linguagem que o sistema deve utilizar para comunicar suas intervenções?, Quão flexível é a interação com o sistema?; são todos tratados neste módulo.

Os Sistemas CAI - Computer-Aided Instruction, antecessores[2] tecnológicos dos STI, apresentam, normalmente, um conjunto de quadros ou frames dispostos em uma ordem pré-programada de aprendizado. Essa ordem não muda conforme o desempenho do aluno que navega pelos assuntos através da escolha em menus ou através de comandos de avançar ou retroceder. As ações dos alunos não são avaliadas de forma que se reflita em estimulantes *feed-back* para o aprendiz. Portanto, quando comparados aos sistemas CAI, os mais marcantes elementos constituintes de um STI são os módulos do estudante e pedagógico, principalmente naquelas funções que permitem o diagnóstico das ações do estudante e das respectivas ações tutoriais.

Para realizar seu objetivo de comunicar o conhecimento os STI tornam-se um empreendimento multidisciplinar (Wenger, 1987) envolvendo várias áreas do conhecimento humano tais como: Epistemologia, Psicologia, Ciência da cognição humana, Inteligência artificial, Educação, Lingüística, Antropologia e Interação homem-máquina. Ainda assim, esses artefatos são difíceis de construir e requerem anos de trabalho, entre outros motivos, porque ensinar é difícil.

2.2 MODELO DO DOMÍNIO

O conhecimento que deve ser comunicado ao aprendiz é tratado quando do projeto da aplicação no Modelo do domínio. Tal conhecimento pode ser incorporado ao modelo quando então ele é programado junto com o sistema ou ser independente do modelo e, neste caso, registrado posteriormente por um tutor humano. No caso de ser registrado pelo tutor humano, é necessário que o projeto da aplicação possua interfaces adequadas para o registro desse conhecimento.

O conhecimento do domínio deve ser pedagogicamente ordenado antes de ser apresentado ao aluno. Esta particular ordem de apresentação dos assuntos é chamada de Currículo. Nesta seção apresentamos uma teoria do currículo proposta por Lesgold e

[2] Antecessor no sentido tecnológico e não temporal visto que hoje a maior parte dos sistemas instrucionais ainda são construídos no paradigma CAI.

resumimos SCORM – um padrão da indústria para o desenvovimento de Sistemas de Gerenciamento de Aprendizagem (SGA).

2.2.1 Uma Teoria do Currículo

Lesgold (1988) apresenta uma proposta de arquitetura para a organização do currículo em três níveis, a saber:

a) Nível do conhecimento (*Knowledge Layer*):- Nesse nível, representa-se o conhecimento através de objetos lição[3]. Esses objetos contém tanto o conhecimento declarativo como o conhecimento procedural (ver seção 2.3). Esses objetos são então organizados através de pré-requisitos. Quanto ao tamanho desses objetos, Gagné (1971 apud Lesgold, 1988) estabeleceu que as leis psicológicas do aprendizado devem determinar o nível desse detalhamento. Van Lehn (1983 apud Lesgold, 1988) sugere que uma simples lição não deve exigir que o estudante aprenda uma regra que possua uma condição disjuntiva. Assim, a definição do tamanho dos objetos lição deve considerar tanto os aspectos psico-pedagógicos do aprendizado quanto evitar durante sua construção encadeamentos disjuntivos (OR/OU).

b) Nível da rede de metas (*Goal Lattice*):-Nesse nível utiliza-se uma árvore que representa a progressiva decomposição do assunto em metas de aprendizado. Em cada nó dessa árvore há um título que identifica um assunto a ser ensinado e cada um nó filho é também identificado por um título e representa uma submeta (subparte) do nó pai. Um nó qualquer dessa árvore ou aponta para um objeto lição (nó folha) ou é uma meta que pode ser decomposta em submetas.

c) Nível dos Meta-assuntos (*Metaissues*):- Os nós raízes da árvore construída na rede de metas podem ser agrupados, conforme a classe de aprendizes que se quer ensinar, em assuntos (ou pontos de vista). Na Fig. 3, apresenta-se um exemplo, extraído de Lesgold (1988), onde um meta-assunto, representando um eventual Curso de Eletricidade Básica, agrupa vários nós raízes do nível da rede de metas.

[3]No original Lesson Object. Atualmente a comunidade tem preferido o termo Objetos de aprendizagem (Learning objects).

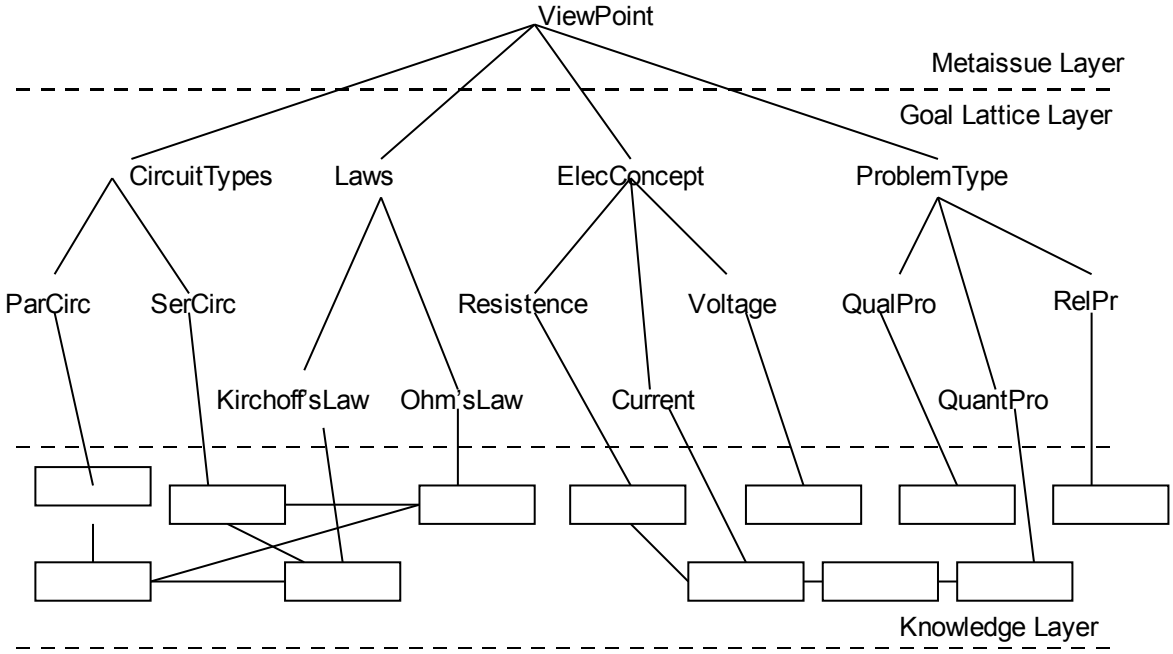

Fig. 3 – Exemplo da arquitetura em 3 níveis para o currículo - Lesgold

2.2.2 Modelo de Referência para Compartilhamento de Objetos de Contéudo

A partir de meados dos anos 90, com o crescimento vertiginioso do uso da Internet, vários sistemas de treinamento e educação foram construídos ou transferidos para essa mídia. Esse treinamento, realizado de forma *on-line*, é também conhecido como *e-learning*, e os sistemas que os suportam podem ser vistos como a articulação de dois artefatos distintos. O primeiro é o Sistema de Gerenciamento de Aprendizagem, ou *Learning Management System* (*LMS*) em inglês, o qual é responsável por direcionar o aprendiz para treinamentos relevantes, acompanhar seu progresso como por exemplo sua avaliação, registrar a localização do aluno no currículo e manter vários outros tipos de dados de acompanhamento do aluno. O outro artefato é o material instrucional ou o conteúdo a ser ministrado ao aluno. Essa dicotomia possibilita que tutores criem objetos de aprendizagem para incluí-los em um LMS sem precisar conhecer detalhes da implementação do LMS. Entretanto, sem a definição de padrões adequados que permitam a comunicação entre o LMS e os objetos de aprendizagem eles seriam construídos "soldados" em um só artefato. Para possibilitar a independência entre esses dois elementos, várias iniciativas, tais como AICC, IMS, ARIADNE e LTSC do IEEE[4], foram desenvolvidas para padronizar essa comunicação.

[4]Ver significado das siglas no parágrafo seguinte.

Em meados de 1999 o Departamento de Defesa (DoD) americano criou a ADL - Advanced Distributed Learning Initiative cujo objetivo é prover acesso rápido e de qualidade ao aprendizado, feito sob medida para necessidades individuais, entregues por preço justo a qualquer tempo e em qualquer lugar (ADL, 2006). Para tanto, essa entidade quis acelerar o desenvolvimento em larga escala de softwares e sistemas de aprendizagem e estimular o mercado para esses produtos. Para alcançar estas metas, ADL aglutinou aquelas iniciativas retrocitadas no padrão SCORM - Sharable Content Object Reference Model que objetiva encorajar a criação de conteúdos de aprendizagem reusáveis como "objetos instrucionais" dentro de uma arquitetura técnica comum para aprendizagem baseadas na computação e na Web. SCORM é baseado no trabalho de várias especificações de e-learning distintas, e em padrões de várias organizações, as quais continuam a trabalhar com a ADL desenvolvendo e refinando suas próprias especificações e padrões, ajudando a construir e desenvolver SCORM. Entre as várias organizações que contribuem, há quatro que são chaves para SCORM.

• ARIADNE - Alliance of Remote Instructional Authoring & Distribution Networks for Europe (http://www.ariadne-eu.org/) .

• AICC - Aviation Industry CBT Committee (http://www.aicc.org/).

• IEEE – LTSC - Institute of Electrical and Electronics Engineers. Learning Technology Standards Committee (http://ieeeltsc.org/).

• IMS - Global Learning Consortium, Inc. (http://www.imsglobal.org/).

Para ajudar a estimular os acordos industriais e realizar tal modelo, SCORM define requisitos funcionais de alto nível para todos os ambientes de *e-learning* os quais combinados com a assunção baseada em Web permite que SCORM ofereça as seguintes habilidades:

• A habilidade de um LMS baseado em Web disponibilizar conteúdos que são autorados usando ferramentas de diferentes fornecedores e trocar dados com aquele conteúdo.

• A habilidade de produtos LMS baseado em Web de diferentes fornecedores disponibilizar o mesmo conteúdo e trocar dados com esse conteúdo durante a execução.

SCORM é apresentado em uma coleção de "livros técnicos". Quase todas as especificações e manuais foram tomados de outras organizações. Esses livros técnicos e suas principais características são resumidas a seguir.

a) Overview:- Conceitua e faz breves introduções para os elementos do modelo.

b) CAM-Content Aggregation Model:- Define como reunir, identificar e acondicionar os conteúdos de aprendizagem. Detalha vários conceitos como Objetos de conteúdo

compartilhável (SCO), Agregação de conteúdo, Pacote, Arquivo de intercâmbio de pacotes (PIF), Metadado, Manifesto, Informação de sequenciamento, Informação de navegação.

c) RTE-Run-Time Environment:- Analisa a API – *Aplication Program Interface* que define como os LMS podem gerenciar o ambiente em tempo de execução, incluindo disparo de programas, comunicação do conteúdo para o LMS, acompanhamento, transferência de dados e manuseio de erros.

d) SN-Sequencing and Navigation:- Trata do sequenciamento e navegação através dos conteúdos. Envolve árvore de atividades, atividade de aprendizado, informação de sequenciamento, informação de navegação e modelo de dados de navegação.

SCORM especifica um modo de construir LMS e conteúdos de treinamento de forma que eles funcionem com outros produtos também construídos baseados em SCORM. Um conteúdo atômico é chamado de SCO- Sharable Content Object e define a menor parte do conteúdo que é reusável e independente, de forma que um LMS trata-o de forma separada em uma tabela de conteúdos, são acompanhados separadamente de outros SCO's, possui seu próprio bookmark, pontuações e status de completação. Conforme a abrangência desejada pelo tutor SCO's podem ser chamados de curso, módulo, capítulo, página e etc.

As diferentes versões de SCORM definem, basicamente, duas coisas: empacotar o conteúdo e trocar dados em tempo de execução. Empacotar conteúdos determina como uma peça de conteúdo seria fisicamente entregue. O coração do empacotamento é um documento entitulado “imsmanifest”. Esse arquivo contém todo o tipo de informação requerida pelo LMS para importar e disponibilizar conteúdos sem intervenção humana. Esse arquivo de manifesto contém um código XML que descreve a estrutura de um curso tanto da perspectiva do aprendiz quanto da perspectiva do arquivo físico. Questões como: Quais documentos devem ser disponibilizados? e Qual o nome desse conteúdo? são respondidas por esse documento. A comunicação em tempo de execução, ou troca de dados, especifica como o conteúdo “conversa” com o LMS enquanto ele é apresentado ao aluno. Há dois componentes principais para essa comunicação. Primeiro, o conteúdo tem que “achar” o LMS e uma vez que o encontre, ele pode então conversar através de uma série de chamadas “get” e “set” e outros vocábulos da linguagem. Por exemplo, o SCO pode “solicitar o nome do aprendiz” ao LMS e o SCO pode “informar ao LMS que a nota do aprendiz foi de 95% nessa teste”. Baseados nesse vocabulário vários e ricos diálogos interativos podem ser realizados entre o SCO e o LMS.

Concluindo, SCORM define uma linguagem que possibilita a comunicação entre SCO's e LMS's que aderirem ao padrão, mas não é seu objetivo e, portanto, não tece nenhum

comentário sobre os detalhes do SCO como seu projeto, métodos pedagógicos que utiliza, sua aparência visual, ou qualquer outra coisa que afete a experiência educacional.

2.3 DIAGNÓSTICO AUTOMÁTICO DAS AÇÕES DOS APRENDIZES

Na modelagem de um STI, apresentado na seção 2.1, destacou-se a necessidade de diagnóstico das ações dos estudantes, a qual pode ser justificada através de vários estudos clássicos entre os quais destacamos o ACT* de Anderson at al.. ACT* - Adaptive Control of Thought é um modelo cognitivo clássico, desenvolvido por Anderson e seus colaboradores, que aborda como os aprendizes formam seus modelos cognitivos e como esses modelos podem ser utilizados para o desenvolvimento de um STI (Anderson, 1983, 1984 Apud Wenger, 1987 e Anderson, 1988). De uma forma bem resumida, esse modelo afirma que:

a) As funções cognitivas podem ser representadas por regras de produção;

b) O conhecimento pode ser do tipo declarativo ou procedural.

c) Inicialmente o conhecimento é adquirido declarativamente através de instrução, comunicação ou observação. Porém, esse conhecimento não afeta diretamente o comportamento e, portanto, precisa ser convertido e reorganizado em conhecimento procedural, que deve ser feito através da experiência para então refletir-se em comportamento. Essa compilação do conhecimento, adquirido via experiência, pode se dar de duas formas:

c.1) Procedimentalização: - Uma parcela geral de um conhecimento é convertido em uma produção específica para se aplicar em uma classe especial de casos.

c.2) Composição de regras: - Umas poucas regras usadas em seqüência para atingir uma meta são agrupadas em uma só regra que combina seus efeitos.

d) Existem dois tipos de memória:

d.1) Uma memória de longo prazo (Long-Term Memory = LTM) que, para efeitos práticos, pode ser considerada de tamanho infinito e onde são armazenadas as regras de produção após o processo de compilação.

d.2) Uma memória de curto prazo (Short-Term Memory = STM) de tamanho limitado que é utilizada como memória de trabalho e onde é armazenado o conhecimento declarativo anterior ao processo de compilação.

Depreende-se desse modelo que durante a prática do aluno, o tutor deve acompanhá-lo e intervir imediatamente caso ele cometa um erro, ou demonstre estar “bloqueado”. Uma intervenção tardia ou não existente pode levar o estudante, ou a não compilar regra nenhuma, ou a compilar o conhecimento de regras erradas, e portanto, não aprender ou aprender errado.

Para a implementação desse modelo num caso da vida real, Anderson desenvolveu o Lisp Tutor no qual utilizou um conjunto de produções organizadas em árvores E/OU - ver Fig. 4 - que representam a forma correta como um dado exercício – comandos de um programa na linguagem Lisp - deve ser resolvido pelo aprendiz. Associada a esta árvore existe um catálogo de erros – *bug catalog* – que registra erros conhecidos. A cada passo do exercício o sistema cria, a partir do *bug catalog* e das produções definidas para aquele passo, um conjunto de regras de produção conhecido como conjunto de conflitos – *conflict set* – o qual sempre contém uma ou mais regras de produção corretas e, geralmente, uma ou mais regras incorretas. O sistema compara a entrada do aprendiz com esse conjunto e se a entrada coincide com uma das produções corretas, nenhuma ação tutorial é tomada e o controle volta para a interface de forma a obter nova entrada do aprendiz. Porém, caso a entrada do estudante não coincida, o sistema intervêm com a ação tutorial prevista para aquele erro (Anderson, 1988). Caso o erro do aprendiz não esteja previsto, o sistema informa que não entendeu aquela entrada e após um certo número de tentativas o tutor informa a resposta correta (Wenger, 1987).

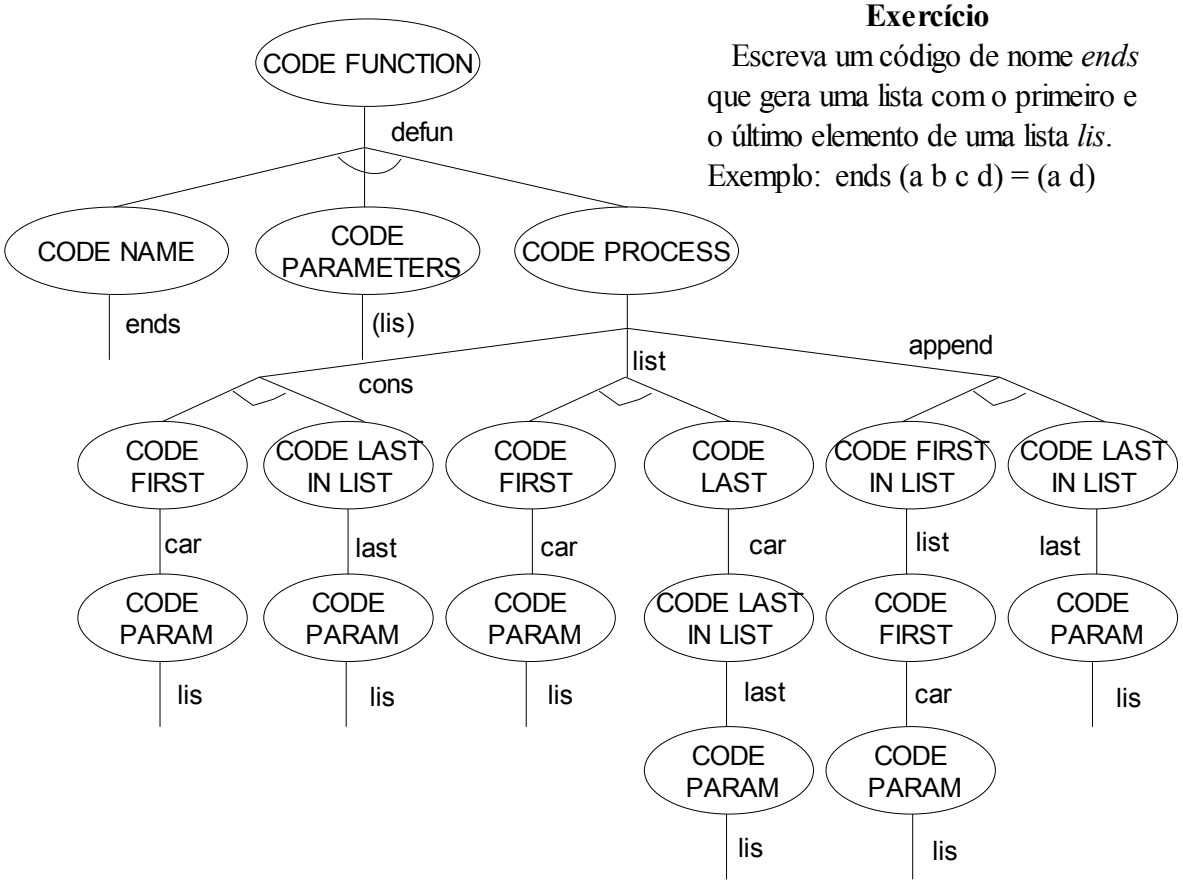

Fig. 4 - Árvore E/OU aplicada em um exercício típico do Lisp Tutor

No exemplo apresentado na Fig. 4, o estudante deve iniciar a resolução do exercício digitando *defun*. Se ele digitar algo diferente disso, a intervenção prevista é imediatamente acionada. Se, por outro lado, ele digitar corretamente, então deve, na seqüência, digitar *ends* e depois *(lis)* e depois o código do processo. Para a digitação do código do processo ele pode, ou iniciar digitando *cons,* ou digitando *list,* ou ainda *append* pois quaisquer dessas palavras fazem parte de uma das três soluções possíveis. Dessa forma, o estudante é rígidamente acompanhado pelo sistema e diagnóstico às suas ações é imediatamente acionado.

Esse tipo de diagnóstico, onde o estudante é dirigido tão de perto pelo modelo foi chamado de *model-tracing* por Anderson. Entretanto, a abordagem *model-tracing* apresenta alguns problemas na sua implementação.

O primeiro deles diz respeito à própria construção do catálogo de erros, pois a variedade de erros do aprendiz é muito grande, tornando impossível a cobertura de todos os erros possíveis. Enquanto que as possibilidades de acerto, no domínio do ensino de programação, são grandes porém limitadas, as possibilidades de erro são ilimitadas. Daí, um algoritmo para tratar todas as possibilidades de erro é, em geral, intratável (Self, 1988). Em um dos seus artigos, Anderson (1988 pg. 97) afirma com respeito ao LISP Tutor : "O modelo do estudante, ... correntemente, consiste de, aproximadamente, 1200 regras para gerar o código correto e o incorreto, representando cerca de 75% do código envolvido em todo o tutor". Qual seja, os tutores construídos no paradigma *model-tracing* demandam uma quantidade enorme de código para definir as ações tutoriais no módulo do estudante. Esse fato levou outros pesquisadores a desenvolverem mecanismos mais genéricos.

Blessing com o seu Demonstr8 (lê-se "demonstreite") procura automatizar a criação da interface, a introdução da lição, as regras de produção e o catálogo de erros para o domínio da aritmética evitando a necessidade de conhecimentos de programação de computadores para tutorar o domínio ensinado, no caso aritmética (Blessing, 1997). Direne, no seu SATELIT-1 um programa para o ensino de programação de centrais telefônicas, permite que o tutor insira objetos que enriquecem a linguagem tutorial utilizando a meta linguagem MELON (MEta-Language for defining OperatioN commands) (Direne, 1997). Santos G. sugere a criação de uma linguagem de autoria de padrões, que permita ao tutor estabelecer os critérios para o sistema verificar a correção e emitir o *feed-back,* no domínio do ensino de linguagens de programação (Santos G., 2003).

Outro problema do paradigma *model-tracing* é saber qual é exatamente a resposta correta, que permita a criação de um catálogo de erros para gerar o *feed-back* ao estudante. Em alguns domínios, como por exemplo na maior parte do jogo de xadrez (início e meio do

jogo), sabe-se que há uma melhor jogada, mas não como definí-la com precisão e, por conseguinte, a criação de catálogos de erros precisos é impossível[5]. Finalmente, em domínios onde não é possível identificar uma única resposta correta, devido às inúmeras características do objeto a ser ensinado, como acontece, por exemplo, no ensino de conceitos, o paradigma *model-tracing* torna-se impraticável.

Conforme exposto na seção 2.1., o *feed-back* ao estudante pode ser feito: a) guiando-o na atividade, b) explanando o assunto relacionado à atividade ou c) remediando a ação errônea do aluno. Ao privilegiar a ação de remediar, o paradigma *model-tracing* restringe sua aplicação à modelagem de procedimentos em um mundo fechado e formal como o ensino de subtração ou programação. Em função das limitações apontadas acima Self (1988) sugere que o modelo do estudante dê menos ênfase ao paradigma *model-tracing* e desempenhe o papel de "*provocar os estudantes a considerar e questionar as justificativas e implicações de suas crenças*".

2.4 PERÍCIA

O assunto da perícia, mais especificamente a diferenciação entre as habilidades do perito e do novato em uma dada especialidade, intriga os estudiosos de STI e da Psicologia cognitiva. Na área da Radiologia médica, trabalhos como os de Lesgold (1989) e de Direne (1997, 1997a, 2001) mostram que:

a) O perito possui uma enorme quantidade de modelos - características típicas de imagens associadas a doenças - em sua LTM, os quais foram construídos através do exame de 10.000 a 200.000 exemplares - chapas radiológicas – em, no mínimo, 10 anos de experiência.

b) Esse conhecimento tende a ser adquirido indutivamente. Desta forma, novos modelos são construídos na LTM do perito, a partir dos casos observados na experiência diária e armazenadas na STM e confrontados com os casos de sucesso anteriores armazenados na LTM.

c) Quando o perito examina uma nova chapa, as características que ele identifica são comparadas com aquelas armazenadas na LTM e, ao identificar características comuns, o perito estabelece o diagnóstico, isto é a doença mais provável.

d) O conhecimento obtido indutivamente pode apresentar erros devido a sobre-generalização (Direne, 1997a). Esse fato reforça a tese, abordada na seção 2.3, de que o

[5] Não é o caso para o final de jogo pois a árvore de jogo se torna menos ramificada. Ver GADWAL (2000) e o sistema UMRAO descrito em 2.6.3.

aprendiz deve ser acompanhado e um mecanismo de *feed-back* que confronte suas crenças deve ser constantemente acionado.

Estes trabalhos sugerem a existência de dois tipos de conhecimento. O conhecimento de princípios e o conhecimento experiencial[6] . O primeiro se refere ao conhecimento formal obtido durante a formação do especialista, por exemplo, na área médica, os conhecimentos de anatomia, teoria da doença, radiologia entre outros. O segundo envolve a integração desse conhecimento com a experiência para a obtenção de diagnósticos corretos em curto espaço de tempo. Enquanto que o conhecimento de princípios é semelhante tanto no perito quanto no aprendiz, o conhecimento experiencial apresenta várias e fortes diferenças. Lesgold (1989) observa seis dessas características diferenciadoras, número este que, posteriormente, foi ampliado para quinze por Direne (2001).

A maneira como o perito da radiologia médica raciocina e usa suas memórias parece não ser específica para esse profissional e pode, ressalvada as peculiaridades de cada domínio, ser estendida para o enxadrista. Pesquisas específicas para o Xadrez (Hyötyniemi 1999, Campitelli 2005 e Burns 2004) permitem inferir que o raciocínio de um grande mestre[7] é análogo ao do perito radiologista. Ou seja, o raciocínio de um grande mestre não é muito mais profundo do que o do novato, sendo que a diferença básica entre eles reside na grande quantidade de "casos exemplares" que o grande mestre obteve ao longo de vários anos de prática, que ele utiliza ao realizar um lance.

A habilidade do jogador de xadrez depende de dois mecanismos (Burns, 2004). O primeiro, chamado de mecanismo rápido, acontece quando o jogador busca em sua memória de longo prazo, imagens exemplares para o lance em questão e, o segundo, chamado de mecanismo lento, corresponde a uma busca no espaço de possíveis movimentos disponíveis para ele, e a respectiva melhor resposta do adversário. As várias pesquisas científicas realizadas para entender esses mecanismos verificaram que o jogador habilidoso se destaca pela utilização mais eficiente do mecanismo rápido. De Groot (1978) expôs jogadores de vários níveis a um tabuleiro com uma particular disposição de peças, por 2-15 segundos, para em seguida, retirar o tabuleiro do alcance de suas vistas, e solicitar que eles reconstituíssem o tabuleiro[8]. Concluiu-se, que a reconstrução do tabuleiro foi uma função da habilidade do jogador, com os grande mestres obtendo quase 100% de tabuleiros corretos. Chase e Simon (1973b, 1973a) desenvolveram testes semelhantes, porém com a diferença que a disposição

[6] que deriva da experiência (Dic. Houaiss)
[7] nome que se dá ao perito no jargão do Xadrez.
[8] O termo "um tabuleiro" será, na maioria das vezes, empregado no sentido de "um tabuleiro com uma particular disposição de peças".

das peças no tabuleiros foi gerada aleatoriamente e, portanto, poder-se-ia encontrar disposições de peças improváveis ou até impossíveis de acontecer em um jogo real. Após os testes, concluiu-se que os resultados dos grandes mestres foram tão pobres quanto os obtidos pelos novatos. Mais tarde Gobet e Simon (1996a) refizeram o experimento e concluiram que, embora os resultados dos grandes mestres tenham piorado bastante, eles ainda eram melhores do que os dos novatos. A partir desses estudos Gobet e Simon (1996b, 2000) apresentaram a teoria dos padrões[9] – uma extensão da teoria dos *chunks* (Chase e Simon 1973b, 1973a) para explicar como os seres humanos utilizam suas memórias. Resumidamente, a teoria dos padrões afirma que:

- Durante a sua carreira os jogadores de xadrez aprendem *chunks* -pedaço ou bloco grande - de configurações típicas que são estocadas em suas LTM.
- Com a prática e o estudo alguns desses *chunks* de 3 ou 4 peças são transformados em padrões de 10 a 12 peças.
- Essas configurações produzem o núcleo do padrão que pode ser completado com informações adicionais.
- Essa estrutura é automaticamente ativada quando o jogador percebe uma posição "familiar" no tabuleiro.
- Quanto mais experiência o jogador tem, mais padrões são estocados em sua LTM e sua performance se torna melhor.

A teoria dos padrões explica porque na reconstrução de tabuleiros possíveis de aparecer em partidas reais, cujas estruturas provavelmente já se encontram em suas LTM, o desempenho do perito é muito superior ao do novato. Por outro lado, também explica porque na reconstrução de tabuleiros com disposição de peças aleatórias, cujas estruturas, provavelmente não estarão em sua memória de longo prazo, o desempenho do jogador experiente é muito prejudicado.

Dessas pesquisas, conclui-se que, seja na aprendizagem da radiologia médica, seja na aprendizagem do jogo de xadrez, a exposição a casos exemplares e não exemplares, através da prática com *feed-back,* é de importância capital para o desenvolvimento da perícia.

2.5 DIÁLOGOS TUTORIAIS

As intervenções tutoriais são muito importante no processo de aprendizado, e a qualidade da linguagem utilizada nessas intervenções agrega um valor cognitivo às lições

[9]No original, em inglês, utiliza-se o termo template.

apresentadas. O estudo dessas linguagens – Taxonômico, Sintático e Semântico - leva em conta contribuições da área da Lingüística e da Interação Humano Computador que foge ao escopo deste trabalho e deve ser objeto de pesquisas futuras. Entretanto, faz-se aqui uma classificação, seguramente incompleta, dos diálogos tutoriais utilizados pelos autores pesquisados.

2.5.1 Diálogo *Model-Tracing*

Neste trabalho, o termo *model-tracing* designará o diálogo – na verdade monólogo – dos sistemas construídos sob esse paradigma. Por trabalharem com catálogos de erros, esses sistemas, normalmente, apresentam intervenções padronizadas, pré-gravadas, às vezes com preenchimento de espaços (*slots*) e que, geralmente, não permitem diálogo com o estudante.

Na Fig. 5, apresenta-se um exemplo, com 3 telas, da intervenção pedagógica do Lisp Tutor de Anderson (1988 pg. 85).

Define a function called ends that takes one argument, which must be a list, and returns a new list containing the first and last items in the argument. For exemple, (ends (a b c d)) = (a d)	Define a function called ends that takes one argument, which must be a list, and returns a new list containing the first and last items in the argument. For exemple, (ends (a b c d)) = (a d)	You will need to call the function CAR, but not yet. You need to construct a list containing the first item in the argument and the last item in the argument, so you need to call a list combining function here.
CODE for ends	CODE for ends	CODE for ends
(defun [a]	(defun <NAME> <PARAMETERS> <PROCESS>) [b]	(defun ends (lis) (car) [c]

Fig. 5 – Exemplo de diálogo no Lisp Tutor.

A resposta que se espera do aluno é *(defun ends (lis)*
(cons (car lis) (last list)))

Na tela [a], o sistema solicita ao aluno que construa uma função que retorne o valor inicial e final de uma lista. O aluno começa, acertadamente, digitando: *(defun*. O sistema verifica o acerto e, na tela [b], se antecipa ao aluno, e solicita os parâmetros: nome da função (<NAME>), parâmetros da função (<PARAMETERS>) e o que a função faz (<PROCESS>). O aluno deve preencher os campos exatamente como o sistema espera que ele faça. Na tela

[c], o aluno digita corretamente o nome da função e os parâmetros da função, porém comete um erro ao digitar o processo, digitando *car*, ao invés de *cons* como previsto. O sistema, automaticamente, dá saída a uma mensagem pré-gravada, que aponta o erro e mostra o que deve ser feito. Não há como o aluno retrucar caso ele tenha alguma dúvida.

2.5.2 Diálogos Computacionais e Lingüística

As dificuldades para aprimorar o diálogo com o computador são de natureza sintática e semântica originando um fértil campo de pesquisas na área de lingüística. Com a finalidade de estudar como o tempo, o modo, aspectos e verbos modais são utilizados na língua inglesa, Isard (1974) utilizou uma variante do jogo da velha para criar um estimulante diálogo entre o computador e o oponente. O programa é capaz de jogar, responder questões acerca do andamento do jogo, inclusive discutir situações hipotéticas no passado e no futuro, e responder questões acerca de eventos possíveis e atuais. Nessa variante, o tabuleiro do jogo é representado pelo "quadrado mágico"[10] ao invés de células vazias e cabe ao sistema substituir o valor das células por um xis (X) quando ele joga, ou um zero (0) quando o oponente joga. Considerando U como o usuário e P como o programa, um diálogo típico seria:

U : Display the board P : 2 7 6 9 5 1 4 3 8 U : What have you taken? P : Nothing U : Move P : 0 7 6 9 5 1 4 3 8 U : I take five. U : You move. P : 0 7 6 9 X 1 4 3 0	U : Had you taken eight when I took five? P : No. U : If I had taken six when I took five, would you have taken what you did? P : No. U : What would you have taken? P : 4. U : Have you taken two? P : Yes. U : Could you have taken two when I took five? P : No. U : Did you take seven? P : No. U : What did you take? P : Eight	U : If I had taken four when I took five, what would you have done? P : Take six. U : Could you have won? P : Yes. U : Will you win? P : Dunno. U : Move. P : I can'not move now. U : I take eight. P : I took eight. U : Had you taken eight when I took 2? P : You haven't taken 2. U : Had you taken two? P : Under what circumstances? U : When you took eight. P : Yes. U : Can I win? P : No.

Fig. 6 – Diálogo computacional e lingüística – Isard 1974

[10]Uma matriz 3x3 onde a soma de qualquer linha, coluna ou diagonal resulta em 15.

Para construir tais diálogos, o programa traduz o texto digitado para funções desenvolvidas em POP-2. Por exemplo, o texto "Would you have taken seven" é traduzido para SUBJUNCT(%WILL(%HAVE(%TAKE(%7%)(%YOU%)%)%)%)[11],onde SUBJUNCT, WILL, HAVE são funções que recebem o resultado de outra função como argumento. Assim, cada frase pode assumir esse aspecto geral, formado pelo modo do verbo (SUBJUNCT), pelo tempo (WILL), pelo aspecto (HAVE) e pela ação (TAKE). Quando, devido às características da frase, uma função não se aplica, utiliza-se funções de enchimento com o objetivo de manter o padrão. Assim, por exemplo, na frase no tempo presente "*I take seven*" substitui-se o modo e o aspecto pelas funções de enchimento NOMODAL, NOASPECT e então teremos algo assim: PRES(%NOMODAL(%NOASPECT(%TAKEN(%7%)(%I%)%)%)%).

A tradução é realizada sobre orações ao invés de frases inteiras. Assim, em frases compostas por mais de uma oração, utiliza-se uma árvore sintática, que decompõe a frase, recursivamente, em suas frases constituintes até se obter uma oração constituída de sujeito + verbo + complemento. Neste nó folha, a tradução é então realizada, as funções obtidas são processadas, e seu resultado dá entrada nas funções, que ficaram pendentes nos níveis superiores da árvore. Na Fig. 7, vê-se um exemplo de decomposição da frase composta: *I took what you would have taken.*

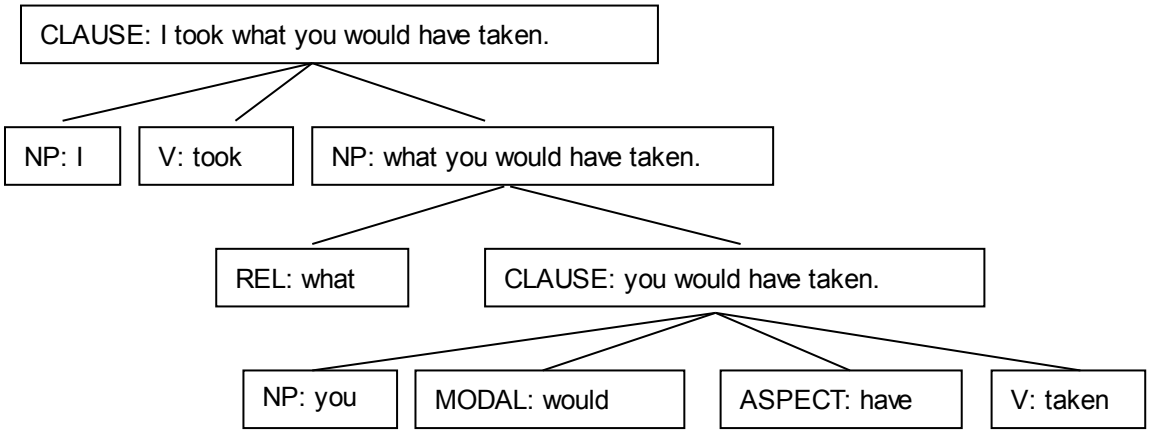

Fig. 7 – Árvore sintática para a frase: *I took what you would have taken.*

Em cada nó dessa árvore – os retângulos da Fig. 7 - um tradutor é invocado. No nó raiz temos um tradutor de frases (CLAUSE) que faz uma análise da frase e chama um tradutor de frases nominais[12] (NP), depois um tradutor de verbos (V) e novamente NP. No segundo processamento de NP, o sistema identifica uma frase composta através do elemento de ligação

[11] Os assustadores sinais de % é uma idiossincrasia da linguagem POP-2 que exige que os parâmetros de uma função estejam cercados por esse sinal

[12] *Nounphrase* no original em inglês.

what e chama o tradutor de elementos de ligação REL, e, novamente, o tradutor de frases CLAUSE. E assim por diante, até ser possível o tratamento de uma oração constituida de um sujeito tratado por NP, um modo tratado por MODAL, um aspecto tratado por ASPECT e um verbo tratado pela função V.

Dessa forma, cada palavra encontrada na frase é associada a uma função. Inicialmente, a função completa é preenchida com funções de enchimento, de forma a manter um padrão, que será modificado à medida que a tradução avança. Quando as palavras são identificadas através de uma análise da frase (*parsing*), sua função associada substitui as funções de enchimento. Os verbos são associados com a função de enchimento "V" e, caso a palavra "*take*", por exemplo, seja encontrada durante o *parsing* o V é substituído pela função TAKE, ilustrado nos exemplos acima. A substituição nem sempre é tão trivial. Por exemplo, verbos intransitivos, como "*win*" em *Will you win?,* além de provocar a substituição do V pela função WIN, também substitui o complemento previsto no padrão pela função de enchimento INTRANS. Existem duas funções que indicam tanto tempo como modo. A primeira é PRES que indica o tempo presente e o modo não subjuntivo. A segunda é REMOTE que tanto pode se comportar como o tempo passado como em "*I took seven*", como um modo subjuntivo, por exemplo, em "*If I took seven, what would you do?*". REMOTE funciona como tempo passado por padrão, mas passa a se comportar como subjuntivo, ao identificar o termo "*if*" junto com os verbos modais "*would*", "*could*" e "*might*". Através desses e de outros mecanismos, o programa consegue emular com perfeição um diálogo com o oponente.

2.5.3 Diálogos Tipo Socrático

O sistema RUI – Representations for Understanding Images (Direne, 1997a) é representativo de uma classe de sistemas[13] que apresenta um diálogo do tipo Socrático com o aluno. O sistema é apresentado em três níveis. O nível de Produção, o nível Instrucional e o nível Conceitual.

No nível de produção, o tutor insere imagens e informações relacionadas que posteriormente, no nível Instrucional, são apresentadas ao aprendiz. No nível Conceitual, são registradas classes de anormalidade – uma visão orientada a objeto que encapsula tanto conhecimentos do domínio (atributos da classe) quanto operações de ensino (operações da classe). Esses atributos e operações podem ser informados pelo tutor através de uma linguagem específica do sistema. No nível Instrucional, são apresentados conhecimentos de

[13] Veja também Radiology Tutor (Sharples, 1991) e Guidon (Clancey, 1987)

princípios – na terminologia de Lesgold – através da apresentação do conhecimento inserido pelo tutor no nível de Produção. O conhecimento experiencial – também na terminologia de Lesgold – é obtido através de um diálogo tipo Socrático, que estende o conhecimento experiencial do aprendiz e utiliza as informações armazenadas no nível conceitual.

A Fig. 8 apresenta um exemplo extraído de Direne (1997a, pg. 46). Nesta figura, vê-se que o sistema faz comentários sobre diagnósticos do estudante, mostra inconsistências em tais diagnósticos, indica áreas importantes e dá prosseguimento ao diálogo com perguntas acerca de características anormais da imagem.

TUTOR: Give a diagnosys for this image. STUDENT : The brain is affected by the lesion. The lesion size is medium. TUTOR: Quite right, the brain is affected by the lesion. The lesion is of type meningioma. The lesion appears to be large. Which parts of the brain are affected by the lesion? STUDENT : The cortical grey is affected... TUTOR: Indeed, the cortical grey appears to be affected by the lesion. ... Do you notice any other indication of meningioma? STUDENT : The lesion is necrotic.	TUTOR: That is not a specific sign of meningioma. It can be associated with glioma, for example. The image opposite shows a case of glioma where the lesion is necrotic. What do you think the arrows above indicate? STUDENT : The lesion is in contact with the pia arachnoid. TUTOR: Correct, the lesion appears to be in contact with the pia arachnoid. What do you notice about the margin of the lesion? STUDENT : The margin of the lesion is mainly sharp. ...

Fig. 8 – Diálogo tipo Socrático utilizado no sistema RUI.

Para conseguir esta elaborada conversação o sistema utiliza quatro mecanismos: a) Visibilidade de características, b) Granularidade anatômica, c) Propagação de regras de ensino e de proteção específicas do domínio, e d) Ações padrão de ensino. A visibilidade de características usa informações da descrição da imagem – registradas pelo perito no nível de Produção – que são comparadas com os diagnósticos do aluno e, se necessário, provê uma retroalimentação imediata às ações do aluno. A granularidade anatômica permite detalhar o diagnóstico do aluno quando, apesar de não incorreto, exigir maiores detalhes para ser considerado correto. Esse mecanismo regula a percepção do aluno para indicar exatamente onde a característica anormal está situada. O mecanismo de propagação de regras de ensino e de proteção específicas do domínio utiliza as informações registradas nas classes de anormalidades obtidas no nível conceitual para controlar os outros três mecanismos. Na Fig. 9, apresenta-se um exemplo de como é definida uma classe de anormalidades e suas respectivas regras padrão de ensino e de proteção. O sistema realiza uma busca em profundidade no corpo de regras e, dependendo das cláusulas de uma regra, o interpretador de diálogo dispara um conjunto de ações que propagam seus efeitos podendo, ou não, disparar outras regras, e assim sucessivamente. Finalmente, as regras padrão de ensino são ativadas pelo interpretador de diálogo, dependendo do valor V, da característica F, no componente

anatômico C. Quando o interpretador de diálogo não encontra uma regra específica do domínio se referindo aos valores de V, F e C – porque essa regra não foi registrada pelo perito - o interpretador ainda consegue criar um discurso a partir desses valores através de um modo de preenchimento de espaços (slots).

```
Component heart Subpart_Of thorax
   Feature Dimensions
      width = {... }
      cardiothoracic_ratio = {... }
      size = {normal, slightly_enlarged, ..., grossly_enlarged}
   Feature Restriction
      h_size1:∀x∀y (heart(x) Λ cardiothoracic_ratio(x,y) Λ y<0.50 ⇔ size(x,"normal"))
      h_size2:∀x∀y(heart(x) Λ cardiothoracic_ratio(x,y) Λ y≥0.50 Λ y<0.60⇔ size(x,"slightly_enlarged"))
      h_size3:∀x∀y(heart(x) Λ cardiothoracic_ratio(x,y) Λ y≥0.60 Λ y<0.70⇔ size(x,"moderately_enlarged"))
      h_size4:∀x∀y(heart(x) Λ cardiothoracic_ratio(x,y) Λ y≥0.70 Λ y<0.80⇔ size(x,"enlarged"))
      h_size5:∀x∀y(heart(x) Λ cardiothoracic_ratio(x,y) Λ y≥0.80 Λ y<0.90⇔ size(x,"markedly_enlarged"))
      h_size6:∀x∀y (heart(x) Λ cardiothoracic_ratio(x,y) Λ y≥0.90 ⇔ size(x,"grossly_enlarged"))
      ct_ratio_mentioned: ∀x∀y (heart(x) Λ cardiothoracic_ratio(x,y))
   Teaching Rules
      ask_heart_size: if /*NONE*/ then
         Action
            ASK(heart, size, "What is the size of the heart?");
         End_Action
      tell_ct_ratio: if /*NONE*/ then
         Action
            TELL(heart, cardiothoracic_ratio, "The cardiothoracic ratio is roughly");
            TELL(NULL, NULL, ".");
         End_Action
      tell_heart_size: if ¬ct_ratio_mentioned then
         Action
            TELL(heart, cardiothoracic_ratio, "Actually, the cardiothoracic ratio is ");
            GRAP_LOC(heart, cardiothoracic_ratio);
            TELL(heart, cardiothoracic_ratio, size, ", giving the heart size ");
            TELL(NULL, NULL, ".");
         End_Action
      tell_heart_size: if ¬ct_ratio_mentioned then
         Action
            TELL(heart, cardiothoracic_ratio, "Actually, the cardiothoracic ratio is ");
            GRAP_LOC(heart, cardiothoracic_ratio);
            TELL(heart, cardiothoracic_ratio, size, ", giving the heart size ");
            TELL(NULL, NULL, ".");
         End_Action
      tell_h_size_without_ct_ratio: if ct_ratio_mentioned Λ (¬h_size1 ∨ ¬h_size2 ∨ ... ∨ ¬h_size6 ) then
         Action
            TELL(heart, size, "From the value of the cardiothoracic ratio we can infer that the size of the heart is ");
            TELL(NULL, NULL, ". The value of range for the cardiothoracic and the heart size are:
                                    0.5 to 0.6 ⇒ slightly enlarged, ...
                                    0.8 to 0.9 markedly enlarged and
                                    greater than 0.9 grossly enlarged."");
         End_Action
   Protection Rules
      /*NONE*/
End_Component
```

Fig. 9 - Descrição parcial do componente coração extraído de Direne 1997a

2.6 JOGOS

Num sentido bastante amplo, um jogo é uma atividade humana, envolvendo uma ou mais pessoas, que a partir de regras pré estabelecidas, buscam atingir uma meta, ao mesmo tempo que procuram impedir que o/os antagonistas consigam atingir as metas deles.

As motivações para o comportamento de jogar podem ser encontradas no entretenimento/lazer, no desenvolvimento de atividades físicas, no aprimoramento educacional, ou em necessidades psicológicas.

Os jogos podem ter alguma das seguintes características :

a) Jogos de habilidade: - É definido, principalmente, pela capacidade mental ou física dos competidores, ao invés de pela sorte, ou por algum tipo de planejamento de ataque ou de defesa. Por exemplo: Tiro ao alvo.

b) Jogos de estratégia: - É um jogo perfeitamente lógico, cujo resultado é influenciado pela interação com o ambiente e com os outros jogadores, sem que o acaso tenha alguma influência no resultado. Todos os jogadores tem o mesmo domínio das regras, dos elementos do jogo, e a mesma pontuação de saída. A diferenciação se processa, durante o jogo, na capacidade que um dos competidores tem de planejar a melhor forma de atingir a meta e de impedir que o adversário o faça. Por exemplo: Damas e Xadrez.

c) Jogos de azar: - É um jogo cujo resultado é totalmente dependente de algum processo randômico. Os processos randômicos podem ser gerados através de uma roleta, cartas de baralho, dados e etc.. Por exemplo: Jogo de dados.

Normalmente, os jogos possuem mais que uma das características acima. Por exemplo, o jogo de futebol envolve tanto habilidade física quanto estratégia, o poker envolve tanto estratégia quanto sorte, e etc.

Com exceção do futebol de robôs, os jogos que visam o desenvolvimento de habilidades físicas não atraíram muito interesse na comunidade de informática. Entretanto, muitos jogos que desenvolvem habilidades de estratégia e sorte têm sido implementados em computadores.

2.6.1 Videogames

Videogames são jogos controlados por computador cuja saída é realizada em unidades de vídeo ou em aparelhos de televisão. Quando utilizam um monitor ligado a um

computador pessoal, eles recebem o nome de jogo de computador, ou PC-game e, quando utilizam um aparelho de TV ligada a uma caixa específica para aquele jogo, são chamados de Console-games.

Os videogames podem ser classificados em gêneros como : Plataforma (Mario), Aventura (Zork, King's Quest), Computer role-playing game (Final Fantasy), First-person shooter (Battle zone), Third-person shooter (Max Payne, The Suffering), Esportes (Soccer, Golf), Corrida (Mario Kart), Shoot 'em up (Space Invaders), Luta (Kung Fu), Quebra cabeça (Tetris, Lumines), Simulação (Flight Simulator, SimCity, Civilization, The Sims), Jogos baseados em rounds (The Battle for Wesnoth) e Real-time strategy (Age of Empires).

Embora estudos recentes como os de Daphne Bavelier e Shawn Green (apud Riesenhuber, 2004) demonstrem que a utilização de videogames aumentam o desempenho em habilidades visuais relacionadas à detecção de objetos em curtos espaços de tempo, os videogames não tem finalidade educacional e seu objetivo principal é o entretenimento.

Um subgênero do tipo *Jogos baseados em round* são os tradicionais *Jogos de mesa*, tais como : Xadrez, Damas, Othello, Gamão, Go e Bridge. Esses jogos foram, e estão sendo, objetos de pesquisas com a finalidade de desenvolver programas competitivos em partidas contra seres humanos. Esses programas, atualmente, podem se equiparar ou vencer os melhores jogadores humanos em Damas, Othelo e Gamão, e estão um pouco atrás em Bridge. Um programa venceu um campeão mundial de Xadrez em plena atividade, enquanto que programas de computador ainda estão em nível de amador em Go. (Russel, Norvig, 2004)

2.6.2 Jogos e Sistemas Tutores Inteligentes

Uma classe de jogos bastante importante para a pesquisa em educação, são aqueles desenvolvidos no paradigma de Sistemas Tutores Inteligentes. O jogo WEST (Wenger, 1987), por exemplo, tem a finalidade de ensinar habilidades na aritmética. Os jogadores são envolvidos em uma corrida de carroças onde a jogada de cada competidor é obtida a partir de operações aritméticas simples com dois operadores fixos (4 operações) e 3 operandos (inteiros de 1 a 9 escolhidos aleatoriamente pelo computador). Os competidores – ora o aluno, ora o computador - devem criar expressões numéricas, cujo resultado corresponde ao número de casas que a sua carroça anda para frente. Caso uma carroça, após uma jogada, se sobreponha a do adversário, este deve voltar para casas especiais anteriores chamadas de *cidade*. Desta forma, nem sempre a melhor estratégia é utilizar expressões que maximizam o resultado. O

sistema cria um diálogo tutorial com o aluno, avaliando suas escolhas em relação ao objetivo do jogo (fazer o adversário voltar para trás, se proteger em uma casa do tipo *cidade*, utilizar um atalho, chegar ao final da trilha). O sistema QUEST - Qualitative Understanding of Electrical System Troubleshooting (White, 1985) é um sistema direcionado ao ensino de como resolver problemas em circuitos elétricos simples. É um programa que monitora todas as atividades do aprendiz, adaptando suas ações às do estudante, e guia-o na solução de problemas. Essas características são entendidas como uma limitação do sistema porque não permitem ao estudante explorar livremente o espaço de ensino que o programa aborda. O projeto WUSOR desenvolvido por Ira Goldstein e Brian Carr no MIT em meados dos anos 70 (ZELHART, 1994) é um tutor baseado no jogo de computador WUMPUS que objetiva exercitar probabilidade, raciocínio lógico, planejamento e decisão, enquanto os alunos se movem através de um labirinto, tentando anular os WUMPUS, e, ainda, evitar vários perigos existentes no labirinto.

2.6.3 STI e Xadrez

Pesquisas em Xadrez acompanham o desenvolvimento da Ciência da Computação[14]. Todavia, a grande maioria dessas pesquisas se caracterizam em desenvolver programas que jogam bem xadrez, e não programas que ensinam a jogar Xadrez. Só recentemente, alguns esforços foram desenvolvidos para produzir programas que ensinem a jogar xadrez. Um desses esforços é o programa UMRAO (Gadwal at al., 2000) que ensina a jogar finais entre Bispos e Peões. Nesse ambiente, o tutor propõe um tabuleiro com peões e Rei de uma cor, por exemplo, brancas, e Rei e Bispo de outra cor, por exemplo, pretas. O aluno então é instado a demonstrar que as brancas jogam e ganham a partida, normalmente com a promoção de um dos seus peões. As ações do aluno são acompanhadas e *feed-backs* automáticos são gerados quando o aluno comete desvios em relação aos objetivos propostos. Esta abordagem, do tipo *model-tracing* por excelência, é obtida através de dois módulos o EXPERT e o TUTOR. O módulo EXPERT é processado somente uma vez para cada problema, e cria um grafo de táticas – uma árvore de jogo (ver seção 3.1) - representando táticas plausíveis para o novato. Esse grafo é construído recursivamente a partir do tabuleiro inicial e das informações registradas na biblioteca de planos do perito e do aprendiz, as quais são registradas por meio de uma linguagem especial (AL1 de Mitchel). Todos os movimentos possíveis são então reduzidos para alguns poucos movimentos plausíveis, através da aplicação de predicados

[14] Consulte Russel, Norvig, (2004) pg. 179 para um resumo das pesquisas na área.

como aplicabilidade e possibilidade[15] associados aos planos do perito. Por outro lado, o módulo TUTOR é processado a cada vez que um novato quer trabalhar em um problema. O programa usa o grafo de táticas associado àquele problema, e que foi pré-compilado no módulo EXPERT, para acompanhar e predizer os movimentos do aprendiz. O estudante pode escolher o nível de *feed-back* que o tutor oferece: a) Ele pode jogar sem *feed-back*, b) Pode pedir uma dica, c) Pode pedir uma detalhada descrição do problema, d) Pode pedir que se apresente uma estratégia vencedora, ou e) Pode pedir ao tutor que retorne ao último ponto, onde ele ainda poderia ter optado por uma tática vencedora.

2.7 LINGUAGENS E FERRAMENTAS DE AUTORIA

Desde que Carbonell (1970a apud WENGER, 1987) apresentou sua dissertação e lançou o seu sistema Scholar, fato que marca o início das pesquisas em Sistemas Tutores Inteligentes (Wenger, 1987), muitos sistemas foram desenvolvidos nesse paradigma. Com a finalidade de sumarizar em profundidade a área, Murray (1999), a partir da análise de 24 sistemas, sugeriu 7 categorias para classificar os sistemas desenvolvidos sob esse paradigma as quais são sumarizadas a seguir.

2.7.1 Planejamento e Seqüenciamento do Currículo

Sistemas nesta classe se caracterizam por organizar o assunto a ser ensinado em partes e subpartes, tais como cursos, módulos, lições e exercícios. Estas unidades instrucionais são organizadas pedagogicamente em currículos, através de relações de dependência, tais como pré requisitos e dificuldade, para posteriormente serem apresentadas ao aprendiz. Ao contrário da organização orientada a quadros dos sistemas CAI, esses sistemas permitem a organização e a apresentação dos assuntos conforme o desempenho do aluno. Entretanto, as decisões pedagógicas são tomadas no nível "macro", isto é, no nível do seqüenciamento das unidades instrucionais. Esta categoria possui os recursos mais básicos presentes em um STI e por isso vários sistemas que se enquadram em outras classes contém essa capacidade mínima. As características desses sistemas torna-os mais apropriados para o

[15] Se são ou não possíveis (feasibility).

ensino de conhecimentos conceituais, declarativos e episódicos e menos para habilidades na solução de problemas e procedural.

2.7.2 Estratégias Tutoriais

Sistemas nessa categoria vão além das características da classe anterior, e apresentam um nível de instrução mais fina. Ações tutoriais são programadas dentro das unidades instrucionais, permitindo que o sistema defina quando e como dar explicações, que tipo de pistas e respostas, e que tipo de questões e exercícios oferecer ao estudante. Essa classe de sistemas ensina mais através da leitura e da reflexão do que da prática.

2.7.3 Simulação de Dispositivos e Treinamento em Equipamentos

Os sistemas nessa classe simulam equipamentos, ou alguma de suas partes, e apresentam perguntas ao aluno referentes a identidade de componentes, passos do processo de operação, passos do processo de manutenção, diagnóstico de falhas quanto a consertos ou troca das partes afetadas. Esses sistemas são fortemente dependentes do domínio e seu núcleo mais complexo é a construção do simulador do dispositivo, que deve reproduzir o comportamento do dispositivo de maneira profunda e fiel. Diferentemente das categorias anteriores, os estudantes só "aprendem fazendo" pois supõe-se que eles já estão familiarizados com o equipamento, e não necessitam de conhecimentos introdutórios ou conceituais. Esses sistemas variam desde aqueles que suportam mais de um domínio como o XAIDA, aqueles que suportam modelos superficiais como RIDES e aqueles que suportam profundos modelos cognitivos do funcionamento do dispositivo como SIMQUEST (Murray, 1999).

2.7.4 Sistemas Especialista e Tutores Cognitivos

Esses sistemas utilizam modelos cognitivos baseados em regras e possuem um profundo modelo da perícia. Eles observam o comportamento do estudante durante a solução de problemas e produzem uma retroalimentação imediata, a partir de regras que traduzem os erros mais comuns, caso as ações do estudante não estejam de acordo com aqueles previstos no modelo. Estudantes que utilizam esses sistemas resolvem problemas e subproblemas

associados dentro de um espaço de metas e quando “bloqueiam” podem solicitar que o sistema processe o próximo passo, ou que complete a solução do problema.

2.7.5 Múltiplos Tipos de Conhecimentos

O conhecimento pode ser categorizado em fatos, conceitos e procedimentos. Cada categoria tem uma forma particular de ensino. Fatos são ensinados com dispositivos práticos repetitivos e mnemônicos, conceitos são ensinados usando analogias de exemplos positivos e negativos progredindo do mais fácil para o mais difícil e procedimentos são ensinados passo a passo. A partir dessa observação, sistemas nessa categoria são construídos para permitir que, através da decomposição de complexas habilidades em componentes elementares de conhecimento e de links entre esses componentes, possam ser autorados e usados em instruções.

2.7.6 Sistemas para Propósitos Específicos

São sistemas muito especializados em uma tarefa ou domínio específico e menos geral. Esses sistemas são projetados para começar com um particular projeto de um tutor inteligente e generalizá-lo para criar uma estrutura para autorar tutores similares.

2.7.7 Hipermídia Inteligente ou Adaptativa

Murray (1999) cria uma categoria especial para sistemas baseados na Web porque esse tipo de sistema tem se tornado mais sofisticado e incorporado métodos e modelos do campo dos sistemas tutores inteligentes. Esses sistemas são, funcionalmente, semelhantes às categorias Planejamento e seqüencialmente do currículo e Estratégias tutoriais, dependendo se o foco da ação tutorial acontece no micro ou no macro nível.

3. CONCEITOS FUNDAMENTAIS

3.1 ÁRVORE E-OU

Normalmente, os algoritmos de busca em IA se utilizam de árvores OU. Nesse tipo de abordagem, um dado algoritmo que queira resolver um dado problema, ao atingir um nó arbitrário opta por um dos ramos de saída baseado em algum critério e, após um certo número de nós pesquisados, encontrando a solução, também definida por algum critério, pode encerrar a busca e parar. A correção dessa abordagem se justifica pois cada ramo de saída de cada nó é disjuntivo[16] (OU) e, se a solução é verdadeira quando percorremos um dos ramos, não precisamos verificar outros ramos da árvore de busca. Por outro lado, em algumas situações, tais como nos jogos, essa abordagem é insuficiente. Se a solução é encontrada em um ramo da árvore de busca, por exemplo a vitória de quem joga, deve-se continuar a busca, pois o adversário de quem joga pode escolher outro caminho em alguma bifurcação "acima" do nó solução – em busca da vitória dele - e não aquele ramo que quem joga optou seguir. Para garantir a correção para esses casos, torna-se necessário que os nós do adversário tenham todos as suas saídas extensivamente[17] exploradas e, portanto, estes ramos de saídas são conjuntivos (E). Nessa parte do trabalho conceitua-se as árvores E-OU a partir dos seus casos mais comuns, as árvores de objetivos e as árvores de jogos.

3.1.1 Árvore de Objetivos e Grafos E-OU

Uma árvore de objetivos descreve a situação na qual um objetivo principal pode ser atingido por meio da aplicação de um método de solução de problemas. Se houver mais de um método disponível, um algoritmo de solução de problemas pode ter que tentar vários deles de forma combinada. Isto se resume a um problema de busca tradicional, onde exploramos um ou outro método (OU). Entretanto, caso a execução dos métodos escolhidos necessite da execução de duas[18] ou mais etapas em série, então o nó que corresponde a esse método pode ser representado por uma subárvore com essas etapas como seus filhos e com a restrição adicional de que todos os nós precisam ser avaliados (E). Para exemplificar, considere o

[16] A operação lógica da disjunção (OR) tem a propriedade de que basta um caminho ser verdadeiro para que toda a disjunção também seja, independente do número de caminhos a inspecionar.
[17] A menos que possamos demonstrar a correção de algum mecanismo de poda.
[18] Se precisar de somente uma etapa então o problema se resume a um problema de busca tradicional (OU).

objetivo de ir de uma cidade A para uma cidade B[19] com o objetivo de gastar menos que 100 unidades monetárias (Ver Fig. 10). Suponha que há 3 métodos (caminhos) para atingir esse objetivo – C1, C2 e C3. Neste caso, o problema se restringe a utilizar, ou o método C1, ou C2, ou C3. Porém, suponha que no caminho C1 encontram-se as cidades a1, a2, e a3 então, para executar o método C1, é necessário realizar cada uma das etapas a1, e a2, e a3. Se a travessia da cidade a1 fosse suficientemente complexa para exigir uma nova abordagem por objetivo – neste caso um objetivo parcial do objetivo geral "Ir de A até B com menos de 100" – reaplica-se o processo, e assim por diante.

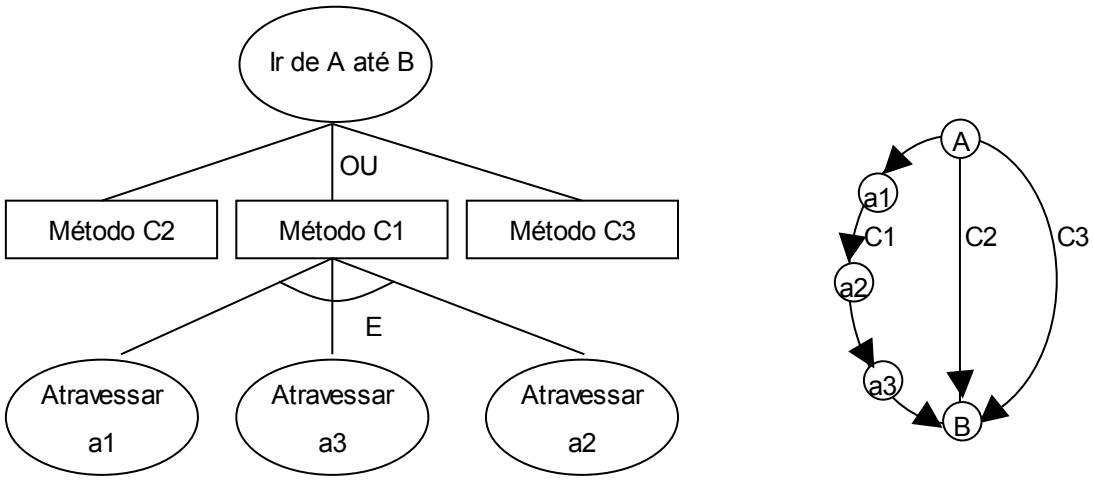

Fig. 10 – Objetivo de ir de A até B e sua representação em uma árvore E-OU

Árvores de objetivos, como a da Fig. 10, se caracterizam por nós do tipo "OU" representando objetivos, tais como os nós em forma de elipse da árvore acima, e por nós do tipo "E" representando opções de solução do problema, tais como os nós em forma de retângulo. Os nós do tipo "E", por definição, exceto quando folhas, sempre possuem uma restrição associada. Assume-se que filhos de um nó do tipo "E" serão sempre do tipo "OU" (e vice-versa) de forma que uma busca em profundidade alterna o exame de nós E com nós OU. Um descendente de um nó pode ser definido de forma recursiva como sendo, ou um dos filhos do nó, ou um dos descendentes dos filhos do nó. Suas folhas são, normalmente, nós do tipo "E", embora que, por razão de completude da definição, seja possível encontrar nós do tipo "OU" como folhas da árvore, mas serão considerados nós de fracasso, pois não há maneira concreta de satisfazer suas restrições.

Um nó do tipo "E" é dito resolvido se todos os seus nós filhos forem recursivamente resolvidos, e além disso, as soluções satisfizerem as restrições presentes no nó. Um nó do tipo

[19] Esse pode ser um problema difícil numa guerra, ou para uma transportadora ou em época de férias.

"OU" é resolvido se qualquer um de seus nós filhos forem recursivamente resolvidos. E, por fim, uma árvore de objetivos é dita resolvida se o seu nó raiz for resolvido.

Para o desenvolvimento de uma busca em árvores E-OU, pode-se embutir os nós E dentro de nós OU de forma a utilizar os algoritmos já desenvolvidos para as árvores OU. Os nós assim construídos são chamados de planos, e a árvore resultante é chamada de árvores de planos. No exemplo da Fig. 10, o Método C1 pode ser resolvido através da execução do plano "Atravessar a1 E Atravessar a2 E Atravessar a3". Dessa forma, pode-se conceituar plano como uma estrutura de árvore, que mostra a relação de condição (E-OU) de execução dos elementos, que o planejador organiza com a finalidade de atingir uma meta de mais alto nível. Sendo assim, efetuando a busca em um espaço de planos (ao invés de ações), pode-se reduzir a árvore de objetivos a um espaço de busca tradicional e, com isso, aplicar os algoritmos desenvolvidos para árvores OU.

Quando todas as restrições em uma árvore de objetivos forem vazias, isto é, são satisfeitas por qualquer solução, então a árvore é simplesmente um grafo E-OU puro.

3.1.2 Árvore de Jogos como Árvore de Objetivos

Uma árvore de jogos é uma árvore de objetivos na forma de uma árvore E–OU. Ela inclui opções de caminhos de movimentos, os quais configuram todas as partidas possíveis do jogo onde, alternadamente, cada jogador faz o seu movimento. Cada lance leva sempre a um conjunto finito de estados – configurações de peças - totalmente previsíveis (Ver Fig. 11).

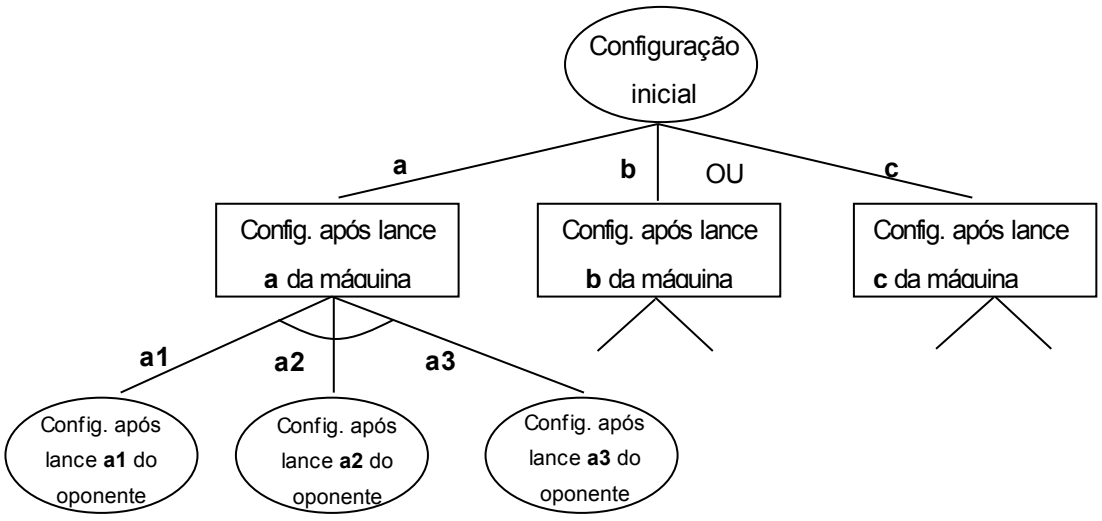

Fig. 11 – Uma árvore de jogos como uma árvore E-OU (Máquina x Oponente)

Na árvore da Fig. 11, os nós "OU" representam a visão do jogador *máquina* e os nós "E" do seu *oponente*. Em suas folhas, supondo que a árvore seja finita, estão as posições de final de jogo, onde a máquina ganhou, perdeu ou empatou. O subproblema mais imediato é o de decidir o próximo movimento da máquina. Entretanto, como o oponente ataca a máquina, ela terá que considerar todas as possibilidades de resposta do oponente para o seu próximo lance, incluindo como será o contra-ataque a cada uma destas respostas. Decorre daí, que o problema – expresso por uma árvore de jogos - passa a ser o de achar uma tática completa para ganhar a partida, e não só o de achar o próximo lance para máquina. Assim, enquanto que o propósito de uma árvore de objetivos é o de encontrar um plano, o propósito de uma árvore de jogos é o de encontrar uma "tática" para ganhar o jogo. Então, entende-se como tática completa, um conjunto de ações (jogadas) que A pode realizar em cada momento da partida, desde a situação atual até a situação final, que neutraliza qualquer das ações possíveis do seu oponente. Formalmente, uma tática completa é uma subárvore com a mesma raiz da árvore de jogo de tal forma que exatamente um ramo é incluído nos nós da máquina e todos os ramos são incluídos nos nós do oponente.

Entretanto, como será visto na seção 4.3., para jogos como o Xadrez, a árvore de jogo é muito grande para permitir que, no atual nível tecnológico, seja possível a montagem de táticas completas. Para superar esse problema, os lances são explorados até um certo nível de profundidade, e nesses pontos, uma função heurística é aplicada para estimar o valor da configuração obtida naquele ponto. Nessa condição, a árvore é chamada de árvore incompleta de jogo e uma tática é chamada de tática incompleta.

Em um jogo típico de tabuleiro, tal como o Xadrez, a função heurística é chamada de Função de Avaliação Estática (FAE). Rotulando a máquina de MAX e seu oponente de MIN, MAX deverá buscar configurações de valores altos da FAE, ao passo que MIN, os de baixo valor. Por isso, a FAE deve ser construída de tal forma que, quando aplicada a uma configuração, forneça uma estimativa da vantagem (FAE > 0) ou desvantagem (FAE < 0) para MAX. Um lance de MAX ou de MIN define um nível de alternância. Em FAEs perfeitas apenas um nível de alternância é necessário para decidir a melhor jogada, entretanto, como raramente elas são perfeitas, a melhoria da estimativa para a melhor jogada reside na tentativa de trabalhar com mais de um nível de alternância, na esperança que isso inclua ações – jogadas - de MIN não previstas na FAE. O número de níveis de alternância até a aplicação da FAE – para obtenção do valor daquela tática - é chamado de profundidade de visada. No caso de haver um estado terminal acima da profundidade de visada, a expansão daquele ramo da

árvore é suspensa e a FAE atribui um valor baseado nos seguintes critérios: um valor muito alto se o nível for de MAX, ou um valor muito baixo se o nível for de MIN, ou o valor 0 (zero) se for empate. Esse procedimento garante que todos os nós terminais – nós folha ou nó na profundidade de visada – sejam avaliados pela FAE e recebam um valor. Todavia, os próximos lances, após a profundidade de visada, continuam dependendo das ações de MAX e de MIN, e não sendo a FAE perfeita, deduz-se que, realísticamente, consegue-se como solução apenas o próximo lance de MAX que leve à melhor configuração no nível da profundidade de visada, supondo que MIN cause os maiores danos possíveis para MAX. Outrossim, define-se como valor representativo de uma tática o menor valor de qualquer nó terminal da subárvore que representa aquela tática, já que uma tática deixa aberta apenas as decisões de lances de MIN que sempre optará pelos piores valores para MAX. Entre todas as táticas possíveis, a melhor tática de toda a árvore de jogo seria a tática que tiver o maior valor representativo. Esse valor é também conhecido como o valor minimax da árvore de jogo.

3.2 O ALGORITMO MINIMAX

A partir do exposto na seção 3.1., é possível descrever um algoritmo que percorra, recursivamente, a árvore de jogo, e obtenha o valor minimax para um dado nó raiz n, da seguinte forma:

$$
\begin{aligned}
&\text{Valor-MiniMax}(n) = \\
&\quad \text{FAE}(n) && \text{se } n \text{ é um nó terminal} \\
&\quad \max_{s \in \text{Sucessores}(n)} \text{Valor-MiniMax}(s) && \text{se } n \text{ é um nó de MAX} \\
&\quad \min_{s \in \text{Sucessores}(n)} \text{Valor-MiniMax}(s) && \text{se } n \text{ é um nó de MIN}
\end{aligned}
$$

Assim, os principais componentes desse algoritmo, no jogo de xadrez são:

a) O estado inicial: - Indica o tabuleiro e o jogador que fará o movimento (corQueJoga).

b) Função sucessor: - Processo que retorna uma lista de pares (movimento, estado) cada qual indicando um movimento válido e o tabuleiro resultante.

c) Teste de término: - Um teste que determina quando a busca deve parar. Quando o jogo termina ou quando a profundidade de visada for alcançada.

d) Função avaliação estática: - Uma função que mede o valor numérico da configuração do tabuleiro que deve ser aplicada em um nó terminal.

O objetivo deste algoritmo é obter o movimento que gere o maior valor de utilidade, após as jogadas da máquina e as do seu oponente, considerando movimentos que maximizem as jogadas da máquina e minimizem o resultado do seu oponente.

Considere, como exemplo, a árvore de jogos de dois lances da Fig. 12. Esta árvore possui 3 táticas possíveis. A primeira, que corresponde ao lance *a* de MAX produz um valor representativo de 3, já que MIN vai escolher o melhor valor para ele ou seja o valor 3. Da mesma forma, a segunda tática tem o valor representativo de 2, e a terceira tática terá o valor representativo de 1. O valor minimax, conforme definido acima, será o maior valor representativo, ou seja 3.

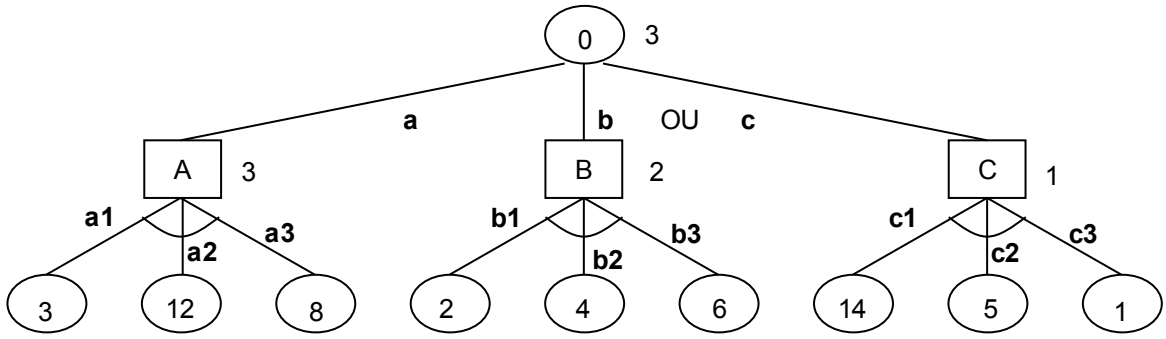

Fig. 12 – Uma árvore de jogos de duas jogadas

Um algoritmo com essas características percorre em profundidade todos os nós da árvore de jogos e calcula a FAE nos nós terminais. Esses valores são então propagados de volta e comparados nos nós pai com os valores provenientes do cálculo de seus irmãos para decidir-se qual o valor minimax – se nó MAX o maior, se nó MIN o menor - daquele nó. Entretanto, essa abordagem é extremamente cara considerando a explosão combinatória em árvores de jogos. Daí, são necessários mecanismos que eliminem parte da busca, mas que ainda garantam a correção da solução encontrada.

3.2.1 Algoritmo SSS*

O algoritmo SSS* (Stockman, 1979 apud Russel, Norvig, 2004) permite a realização de podas em táticas de árvores de jogo com o uso dos valores heurísticos já obtidos, tanto nos nós MIN como nos nós MAX.

Para exemplificar como se processa a poda em um nó MIN, considere a Fig. 13.

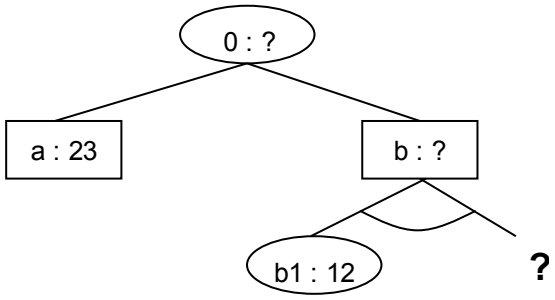

Fig. 13 – Qual o valor minimax do nó MAX 0 ?

Um irmão *a* do nó MIN *b,* que está sendo processado, obteve um valor minimax de 23. Caso um dos filhos de *b* obtiver um valor minimax de 12 – ou qualquer coisa menor que 23 – então não é necessário desenvolver e processar mais filhos de *b* – ou seja, poda-se todas as subárvores que tem como raiz um nó filho do nó *b* - pois : a) Se os nós podados tiverem valor minimax maior que 12, a decisão no nó de MIN vai resultar em 12 (min), b) Se os nós podados tiverem um valor minimax menor ou igual a 12 seu valor seria escolhido por MIN mas perderia na decisão de MAX porque 23 é maior que 12 e como o resultado é menor que 12 resulta que esse resultado é menor que 23. Algébricamente, tem-se que :

VlrMinMax(0)=max(23, min(12, x)). Daí, vem que:

a) Se x>12 então min(12,x)=12. Daí: VlrMinMax(0)=max(23,12)=23,

b) Se x<=12 então VlrMinMax(0)=max(23, z), como x<=12 e z =min(12,x) z é no máximo igual a 12, e ainda menor que 23. Daí: VlrMinMax(0)=23

Para exemplificar como se processa a poda em um nó MAX, considere a Fig. 14.

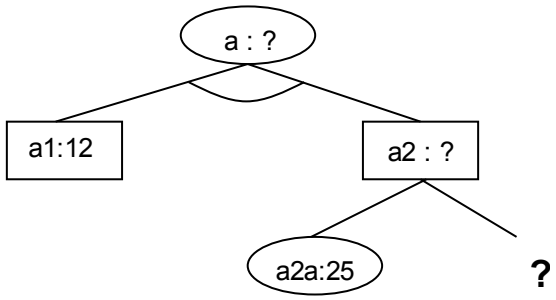

Fig. 14 – Qual o valor minimax do nó MIN a ?

Um irmão *a1* do nó MAX *a2*, que está sendo processado, obteve um valor minimax de 12. Caso um dos filhos de *a2* obtiver um valor minimax de 25 – ou qualquer coisa maior que 12 – então não é necessário desenvolver e processar mais filhos de *a2* – ou seja poda-se todas as subárvores que tem como raiz um nó filho do nó *a2* - pois : a) Se os nós podados tiverem um valor minimax menor ou igual a 25 a decisão no nó de MAX vai resultar em 25

(max), b) Se os nós podados tiverem valor minimax maior que 25 seu valor será escolhido por MAX mas será preterido na decisão de MIN porque 25 é maior que 12 e como esse resultado é maior que 25 resulta que esse resultado é maior que 12. Algébricamente tem-se que :

VlrMinMax(a)=min(12,max(25,x)). Daí, vem que:

a) Se x<25 então VlrMinMax(a)= min (12,25)=12,

b) Se x>=25 então VlrMinMax(a)=min(12, z), como x>=25 e z =max(25,x) z é no mínimo igual a 25 e ainda maior que 12. Daí: VlrMinMax(a)=12.

O algoritmo SSS* armazena todos os nós a espera de serem desenvolvidos, gerando uma carga computacional que torna impraticável sua utilização em árvores com muitos níveis.

3.2.2 Algoritmo com Poda Alfa-Beta

Um algoritmo minimax com poda alfa beta é diferente do algoritmo SSS* na medida em que caminha exclusivamente em profundidade na árvore de jogo, mas também utiliza valores da FAE para parar a busca. Por isso, muito do que foi apresentado para o algoritmo SSS* também vale para o algoritmo alfa-beta.

A poda alfa beta obtém seu nome a partir de dois parâmetros que descrevem os limites sobre os valores que são propagados de volta durante o percurso na árvore de jogo, a saber:

Alfa = o valor da melhor escolha para MAX - o de valor mais alto - que encontramos até o momento, em qualquer ponto de escolha ao longo do caminho. Inicialmente, esse valor é carregado com o menor valor possível (-Infinito).

Beta = o valor da melhor escolha para MIN - o de valor mais baixo - que encontramos até o momento, em qualquer ponto de escolha ao longo do caminho. Inicialmente esse valor é carregado com o maior valor possível (+Infinito).

No nós MAX se algum dos valores que retornam da chamada recursiva for maior que o valor de Beta então o algoritmo retorna o valor de Beta como o valor do nó, podando eventuais subárvores filhas restantes. Se isso não acontecer, então retorna o máximo entre os valores obtidos e atualiza Alfa como o máximo entre o próprio alfa e o valor retornado.

Semelhantemente, nos nó MIN se algum dos valores que retornam da chamada recursiva for menor que o valor de Alfa então o algoritmo retorna o valor de Alfa como o valor do nó, podando eventuais subárvores filhas restantes. Se isso não acontecer, então

retorna o mínimo entre os valores obtidos e atualiza Beta como o mínimo entre o próprio Beta e o valor retornado.

3.3 XADREZ

O jogo de Xadrez pertence a uma categoria de jogos que em IA é definido como um jogo determinístico, de revezamento de dois jogadores, de soma zero com informações perfeitas. Onde :

- Determinístico: - Não há nenhum elemento aleatório na jogada de cada jogador.
- Revezamento de 2 jogadores: - O jogo se desenvolve com jogadas alternadas entre os dois jogadores.
- Soma zero: - Significa que a soma dos valores de utilidade no final do jogo são sempre iguais e opostos. Isto é, se um ganha 1 ponto, o outro perde 1 ponto.

O termo determinístico implica, também, que é possível achar um algoritmo o qual, se aplicado, determina a tática que leva à vitória ou ao empate a quem iniciar o jogo, independente das táticas do antagonista. No jogo de Xadrez, no atual nível tecnológico, é impossível examinar todo o espaço de busca para achar o caminho para a vitória ou para o empate. A árvore de busca do Xadrez tem um fator de ramificação médio de 35 e as partidas podem chegar a 50 movimentos para cada jogador o que nos leva a uma árvore de busca de 35^{100} ou 10^{154} nós (embora o grafo de busca tenha "apenas" 10^{40} nós distintos). Apesar da existência de mecanismos de poda, como por exemplo poda alfa-beta visto na seção 3.2.2, é impossível, em um tempo razoável, uma máquina, na tecnologia atual, examinar todo o espaço de busca de um jogo de Xadrez. Para superar essa limitação, torna-se necessário a aplicação de uma função de avaliação estática (FAE) que avalie um nó intermediário de forma a indicar a proximidade com a vitória (heurística). Um algoritmo típico utilizado para examinar o espaço de busca e definir o melhor lance é o algoritmo MiniMax visto na seção 3.2.

Se a profundidade máxima da árvore de jogo for h e existirem b jogadas possíveis, em média, então a complexidade de espaço é $O(bh)$ para um algoritmo que gera todos os nós sucessores de uma vez, ou $O(h)$ para um algoritmo que gera um sucessor de cada vez. A complexidade de tempo desse algoritmo é $O(b^h)$ e pode ser bastante reduzida com a adoção de poda alfa-beta podendo atingir $O(b^{3h/4})$ no caso de seleção de nós aleatórios. Para aumentar a velocidade na pesquisa da árvore de busca, várias técnicas foram agregadas ao algoritmo Minimax.

3.3.1 Ordenação de Jogadas

Caso seja possível selecionar primeiro os os melhores, a complexidade de tempo pode ser reduzida até $O(b^{h/2})$. O valor de utilidade do nó a ser desenvolvido é desconhecido e, portanto, não podemos ordená-los previamente. Todavia, no jogo de Xadrez, uma ordenação bastante simples tal como: captura primeiro, depois ameaças, depois movimentos para frente, e depois movimentos para trás, leva a uma complexidade de tempo somente duas vezes maior que o melhor caso – $O(b^{h/2})$.

3.3.2 Tabelas de Transposição

Caso um movimento das brancas *a1* possa ser respondido pelas pretas por *b1,* e as brancas possuam um movimento não relacionado *a2*, talvez do outro lado do tabuleiro, que possa ser respondido pelas pretas com o movimento *b2*, as seqüências de jogadas (*a1*, *b1*, *a2*, *b2*) eqüivale[20] à seqüência (*a1*, *b2*, *a2*, *b1*) e a todas as outras permutações que se iniciam em *a1*. Então, vale a pena armazenar essa disposição em uma tabela de hash para não precisar recalculá-la em ocorrências subseqüentes. Essa tabela de tabuleiros já examinados é tradicionalmente chamada de tabela de transposição.

3.3.3 Busca de Quiescência

Movimentos quiescentes são aqueles que não mudam de forma significativa o andamento da partida. Por exemplo, uma captura favorável é não quiescente do ponto de vista de uma FAE que privilegie a captura de peças inimigas. Em algoritmos mais elaborados, a altura de visada da árvore de jogos não é constante, pois é mais vantajoso parar a busca quando os movimentos são quiescentes e investir o tempo remanescente em movimentos não quiescentes.

3.3.4 Tabelas de Aberturas

Uma das características mais interessante do Xadrez é que suas partidas são documentadas, através de padrões muito bem estabelecidos[21]. Essa documentação permite o

[20] Isso é terminarão com a mesma disposição de peças.
[21] Ver anexo I – Notações para o jogo de Xadrez.

estudo das partidas realizadas pelos grandes mestres, as quais revelam padrões nas táticas utilizadas no inicio do jogo – aberturas – a ponto de receberem nomes próprios, como por exemplo: *The Bird Opening*. Isso significa que, apesar de haver várias possibilidades de lances no inicio do jogo, opta-se sempre por padrões estabelecidos, os quais se revelaram mais eficientes do que a aplicação de táticas aleatórias. Assim, é comum que os programas desenvolvidos para jogar contra os humanos utilizem bancos de dados de aberturas – criados por peritos humanos - e as utilize para a definição do seu lance.

3.3.5 Tabelas de Finais

A prática utilizada com relação às aberturas são também empregada nos finais das partidas. No jogo de Xadrez, identifica-se finais típicos, os quais são analisados e as melhores táticas são registradas. Por exemplo, Rei e dois Bispos versus Rei, Rei e Torre versus Rei. Essas táticas são então recuperadas e utilizadas por programas que jogam Xadrez.

3.3.6 Avaliação Estática

A função de avaliação estática ou heurística é constituída normalmente de vários itens que são avaliados individualmente e depois somados para a obtenção de um único valor para aquele tabuleiro. Por exemplo, valor da peça, número de casas avançadas, quantas e quais peças ataca, xeque e etc.. Esses itens, que compõe a heurística, dependem do momento do jogo – inicio, meio ou fim – e de uma região do tabuleiro. A quantidade e qualidade dessas heurísticas itens fazem a diferença na avaliação de um tabuleiro e no resultado da partida.

3.3.7 Deep Blue

Um sistema computacional de nome *Deep Blue* foi a primeira máquina a vencer uma competição de Xadrez contra um campeão mundial de posse do título (Garry Kasparov) e nas mesmas condições existentes nos campeonatos. Na primeira competição, em Fevereiro de 1.996, ele venceu a 1ª Partida da série de 6. Contudo, Kasparov venceu as outras 3 e concedeu 2 empates, de forma que a competição terminou em 4-2 para Kasparov. *Deep Blue* foi então bastante modificado e jogou novamente contra Kasparov em Maio de 1997, vencendo-o por

3.5-2.5. Kasparov venceu a 1ª e *Deep Blue* venceu a 2ª e a 6ª partidas e houve empates nas partidas 3, 4 e 5.

Para ilustrar os conceitos apresentados neste capítulo resume-se as características de *Deep Blue* nesse último confronto, extraído de Russel e Norvig (2004). *Deep Blue* era um computador paralelo com 30 processadores IBM RS/600 executando busca por software, 480 processadores VLSI, especializado para Xadrez, que executavam a geração de movimentos, sua ordenação e busca por hardware nos últimos níveis da árvore de jogos. Este aparato buscava em média 126 milhões de nós por segundo[22], alcançando rotineiramente uma profundidade de 14. O algoritmo utilizava busca, com poda alfa-beta, de aprofundamento iterativo, com tabelas de transposição e pesquisava extensões para jogadas promissoras, podendo em alguns casos explorar até o nível 40. A função de avaliação estática tinha 8.000 características, um livro de abertura com 4.000 posições e banco de dados com 700.000 jogos de grandes mestres e um grande banco de dados de finais com 5 peças e muitas com 6 peças.

[22] Ao se contrapor essa fantástica capacidade de processamento aos 10^{40} nós da árvore de busca verifica-se que DeepBlue levaria 2,5 x 10^{24} anos para explorar toda a árvore. Lembrando que o Universo existe a apenas 10^{10} anos conclui-se que a força bruta, no atual nível tecnológico, não é uma boa estratégia para criar jogadores de Xadrez.

4. SISTEMA DE AUTORIA PARA ENSINO DE XADREZ

As pesquisas delineadas no capítulo 2 mostraram modelos da mente humana e seus mecanismos de aprendizagem de princípios e de perícia, além disso, enfatizou formas de organizar os conhecimentos que devem ser apresentados aos aprendizes, e também, conceitos de STI, jogos e jogos de Xadrez. No capítulo 3, apresentou-se algoritmos e técnicas utilizadas no desenvolvimento de jogadores automáticos com foco no jogo de Xadrez. Este capítulo tem por objetivo apresentar o sistema XadrEx – Expertise em Xadrez que procura materializar os conceitos apresentados. Este sistema, ainda num estágio de protótipo[23], aborda os seguintes aspectos:

1) Como um tutor humano pode registrar conjuntos de unidades instrucionais, estruturadas em cursos, módulos e lições para a aprendizagem de conhecimentos declarativos na área de domínio do jogo de Xadrez. Essas lições precisam ser pedagogicamente organizadas em um currículo, a partir de pré-requisitos e graus de dificuldade definidos pelo tutor.

2) Como estender as unidades instrucionais para que, além das lições, contenham práticas na forma de exercícios que permitam ao estudante a compilação, através da experimentação, do conhecimento declarativo obtido.

3) Como possibilitar que o estudante pratique por conta própria, independente de uma unidade instrucional, para desenvolver sua perícia inicial. Essa prática experiencial se dá, na área de domínio do jogo de Xadrez, através de partidas contra a máquina, do estudo de partidas clássicas, da continuação de partidas clássicas ou a partir de uma configuração definida pelo aluno. A expectativa é que sua exposição a "casos exemplares" desenvolva alguma perícia inicial no jogo de Xadrez.

4) Como as práticas desenvolvidas pelo aluno, seja nas unidades instrucionais, seja na prática por conta própria, podem ser avaliadas pelo sistema, durante todo o transcurso de uma partida, e como um diagnóstico de suas ações pode ser realizado imediatamente, através de um diálogo tutorial.

Baseado na categorização apresentada por Murray (1999) e explanadas na seção 2.6, o protótipo desenvolvido deve ser classificado como um sistema do tipo Hipermídia inteligente, com planejamento e seqüenciamento do currículo e estratégias tutorias.

---

[23] Rubrica: informática. Versão preliminar, geralmente reduzida, de um novo sistema de computador ou de um novo programa, para ser testada e aperfeiçoada. (Dic. Houaiss)

4.1 VISÃO GERAL DO SISTEMA

As funções identificadas para o sistema foram organizadas em quatro partes, como sugerido na seção 2.1, quais sejam:

a) Conhecimento do domínio, que estabelece o que deve ser comunicado.

b) Modelo do estudante, o qual modela aspectos do destinatário da comunicação.

c) Conhecimento pedagógico, o qual estabelece a habilidade de comunicar esse conhecimento.

d) Interface, que estabelece a forma de comunicar.

Na Fig. 15, apresenta-se um diagrama de blocos com essas principais funções, as quais descreve-se, com detalhes, na seqüência deste capítulo.

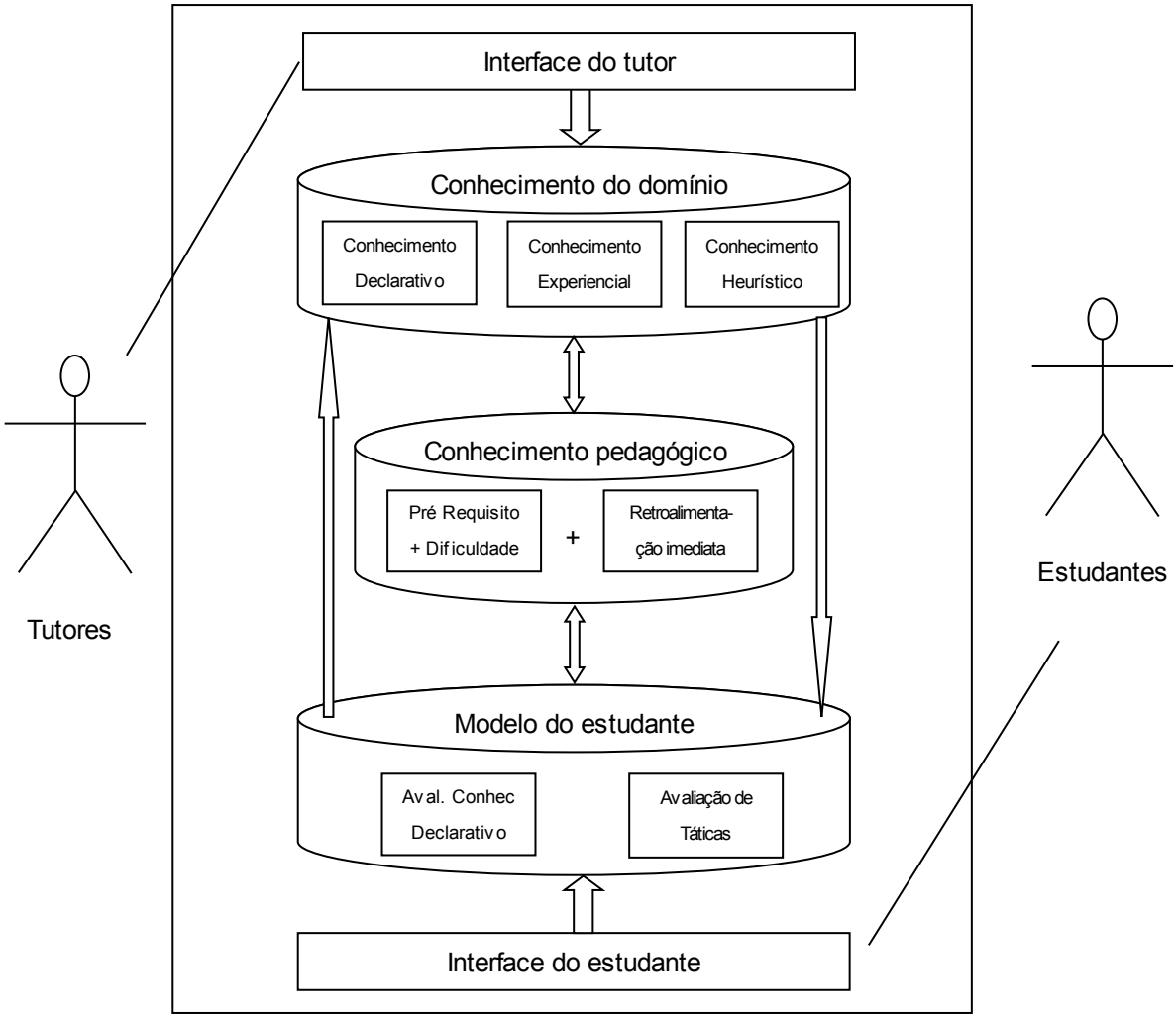

Fig. 15 – Arquitetura do XadrEx

As funções desenvolvidas para o sistema foram agrupadas, por ator[24], em módulos com suas respectivas atividades. Os módulos e atividades são os seguintes:

Módulo do Tutor agrupa as atividades Curso/Estilo, Comando/Heurística e Partida.

Módulo do Aprendiz agrupa as atividades Lições, Currículo e Jogo.

Módulo do Mediador agrupa as atividades de administração do sistema.

4.2 INTERFACE

Estas funções são apresentadas em uma interface padronizada constituída de uma tela dividida em 4 áreas, como mostrado na Fig. 16 e cujas finalidades descrevemos a seguir.

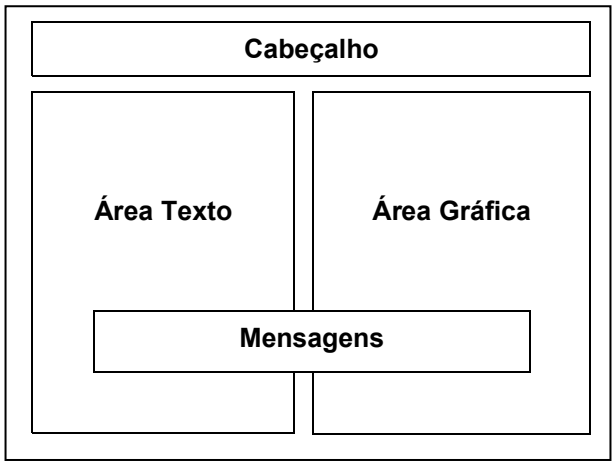

Fig. 16 – Lay out da interface

A área *Mensagens* só aparece quando o sistema tem mensagens para apresentar, às quais, depois de lidas, devem ser fechadas pelo usuário.

A área *Cabeçalho* sempre apresenta a sigla da UFPR, o nome do sistema, o ator -Tutor, Aprendiz ou Mediador-, seu userName, Atividades disponíveis, botão de OK e a sigla do CEX.

No módulo do tutor as áreas são utilizadas conforme a atividade:

a) Na atividade *Curso* a área texto é utilizada para registrar detalhes de uma lição, bem como título, dificuldade, pré-requisito e estilo de um curso, de um módulo e de uma lição. A área gráfica é utilizada para registrar o título e os detalhes de um exercício, além de título e detalhes de estilos. A área Cabeçalho lista opções que podem ser utilizadas pelo tutor, quais sejam, incluir, excluir, alterar, consultar, testar, aplicar estilos, e liberar para os

[24] Usuário do sistema no jargão da UML.

aprendizes os cursos, módulos, lições, exercícios, e estilos. É possível, também, a navegação para outras atividades do módulo do tutor.

b) Na atividade *Comando/Heurística* a área texto é utilizada para registrar detalhes de comandos, enquanto que na área gráfica é possível a especificação de detalhes de heurísticas. Na área cabeçalho existem as opções previstas para estas atividades, quais sejam, incluir, excluir, alterar e consultar.

c) Na atividade *Partida* a área texto é utilizada para registrar, comentar e importar partidas. A área gráfica é utilizada para a apresentação dos tabuleiros no caso de se optar por testar a apresentação da partida. Na área de cabeçalho, existem as opções previstas para essa atividade, quais sejam, incluir, excluir, alterar, consultar e testar partidas.

No módulo do aprendiz as áreas são utilizadas conforme a atividade:

a) Na atividade *Currículo* a área de texto apresenta a lista de lições disponível no XadrEx e que não foram selecionadas pelo Aprendiz, enquanto que a área gráfica apresenta a lista de lições disponível no sistema e que foram selecionadas pelo Aprendiz, e que compõe o seu currículo. A área cabeçalho apresenta as opções previstas para essa atividade, quais sejam, criar, excluir, alterar e consultar Currículo.

b) Na atividade *Lições* a área texto apresenta a parte textual – daí seu nome - da lição ou do exercício na ordem definida no currículo do aprendiz, enquanto que, na área gráfica apresenta-se a parte gráfica – daí seu nome - de lições ou exercícios, normalmente partidas constituídas de um ou mais tabuleiros. A área cabeçalho apresenta as opções previstas para essa atividade, quais sejam, ir para a próxima lição, ir para a lição anterior, ir para o exercício, caso exista algum previsto para essa lição, ou Sair dessa atividade.

c) Na atividade *Jogo* as áreas mudam conforme a opção selecionada pelo Aprendiz. Caso selecione Consultar, ou quando entra pela primeira vez, a área de texto apresenta a dados da partida selecionada como por exemplo sua descrição em formato PGN e na área gráfica apresenta os tabuleiros dessa partida. Se optar por Comentado então a área reservada para a descrição PGN é substituida pelos comentários associados a cada tabuleiro. Se optar por Novo, a área de texto permite o registro de uma disposição inicial, a seleção de uma cor das peças e o tipo de diálogo tutorial que deve ser gerado pelo sistema a medida que o aprendiz desenvolve uma partida contra a máquina; e, a área gráfica é utilizada para apresentação dos tabuleiros criados durante a partida e uma caixa de texto para o aprendiz digitar seu lance. A área cabeçalho apresenta as opções previstas para essa atividade, quais sejam, Novo, Lance, Comentado, Excluir, Consultar ou Sair.

O módulo do *Mediador* ainda necessita de um maior detalhamento e deverá ser objeto de trabalhos futuros.

4.3 CONHECIMENTO DO DOMÍNIO

O conhecimento do domínio é registrado pelo professor de Xadrez, que detém a perícia na área. Esse conhecimento é classificado como Declarativo, Experiencial ou Heurístico. Para cada um desses tipos foram criadas interfaces que permitem o seu registro.

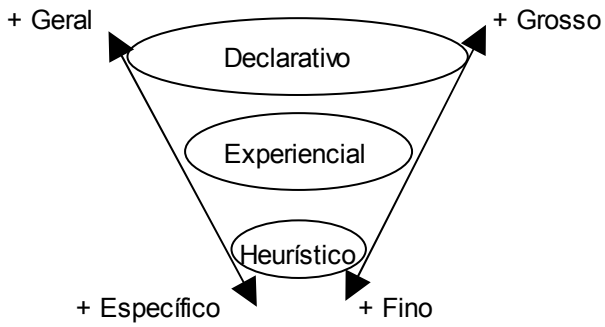

Fig. 17 – Tipos de conhecimento no XadrEx

4.3.1 Conhecimento Declarativo

A expressão do conhecimento declarativo representa o nível mais grosseiro entre os conhecimentos que darão entrada no sistema. Para permitir que o tutor registre esse conhecimento o sistema disponibiliza um tela conforme mostra a Fig. 18. Essas informações serão processadas pelo sistema e serão apresentadas ao aluno da forma como aparece na Fig. 19.

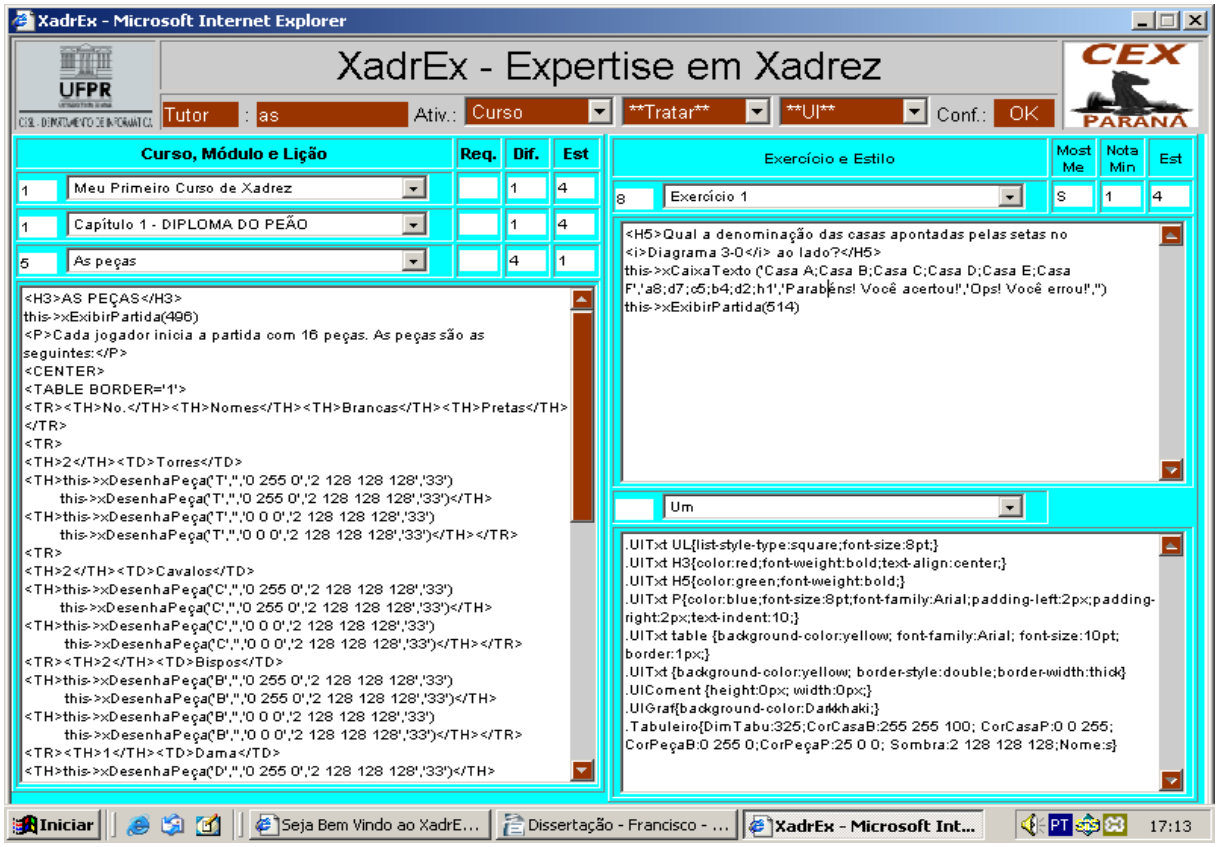

Fig. 18 – Interface para o registro de unidades instrucionais

Fig. 19 – Uma unidade instrucional e uma lição.

Na área texto (ver Fig. 18), o tutor registra seu conhecimento declarativo que será exibido ao aluno na área texto no módulo do aprendiz através de lições (ver Fig. 19). Para tanto, o tutor utiliza uma linguagem específica, a qual será, posteriormente, interpretada pelo sistema para gerar a lição. Esta linguagem mistura marcações HTML (*HyperText Markup Language*) – padrão da indústria - que controla basicamente os estilos de apresentação da lição, com a inserção de comandos – padrão do XadrEx - que controlam comportamentos específicos, tanto nas áreas texto quanto na área gráfica. Uma lição ou exercício, em tal linguagem, é analisada pelo sistema para extrair os comportamentos previstos para os comandos, por exemplo desenhar um tabuleiro com as características especificadas, cuja saída deve ser direcionada para a área gráfica. Esta análise envolve uma busca recursiva pelos textos a procura de comandos. Ao achar um comando, o sistema deve obter seu script no banco de dados, e interpretá-lo para buscar por subcomandos, e assim sucessivamente. Os *scripts* dos comandos obtidos formam a classe Comandos, que é dinamicamente construída, para cada lição ou exercício, e que deve estar disponível quando o interpretador[25] processar a página para enviar seu resultado ao navegador. Por exemplo, na lição *As Peças* apresentada na Fig. 18, o comando *desenhaTabuleiro* é invocado. Ao processar essa lição, o sistema deve obter o script do comando *desenhaTabuleiro* e de todos os subcomandos que contribui para o resultado de desenhar um tabuleiro. A Fig. 20 mostra as funções que colaboram com *desenhaTabuleiro*.

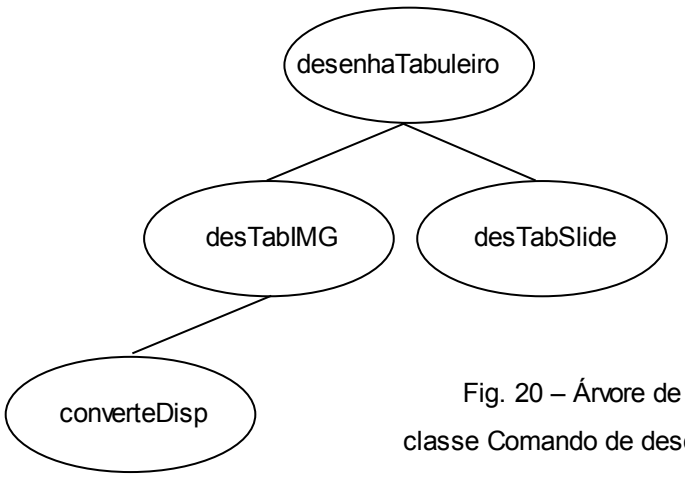

Fig. 20 – Árvore de comandos para a classe Comando de desenhaTabuleiro

Após a criação da classe Comando, essa lição é processada pelo sistema e exibirá para o aluno uma tela como a da Fig. 19. Nesta figura, os botões de navegação, abaixo do desenho do tabuleiro, são indicativos que o comando *desenhaTabuleiro* é geral o suficiente

[25] No nosso protótipo foi utilizado a linguagem PHP e portanto nos referimos ao interpretador PHP.

para permitir ao tutor construir vários tabuleiros para exemplificar suas idéias expressadas na na área de texto da lição.

O aluno, após estudar a lição, tanto através da leitura dos textos, quanto da inspeção dos tabuleiros relacionados, pode escolher no menu suspenso realizar os exercícios associados a essa lição e navegar para uma tela semelhante à da Fig. 21 onde, na área de texto são apresentados os exercícios, e na área gráfica os tabuleiros necessários para a realização do eexercício. Caso o aluno obtenha a pontuação esperada para esta lição, outra lição lhe será apresentada de acordo com o previsto em seu currículo.

Fig. 21 – Uma unidade instrucional e um exercício.

4.3.2 Conhecimento Experiencial

O conhecimento experiencial é registrado pelo tutor com o objetivo de "fixar" os conhecimentos adquiridos através de situações práticas. Essas práticas podem ser registradas

através de exercícios associados a uma unidade instrucional como aquele apresentado na Fig. 21, ou através de partidas clássicas comentadas.

A inserção de exercícios consiste de procedimentos semelhantes ao realizado para inserir uma lição. Observe na Fig. 18, que a linguagem utilizada é a mesma, isto é HTML + comandos. Neste exemplo, acompanhamos o comportamento do comando *caixaTexto* que recebe uma string e gera os exercícios apresentado na Fig. 21.

Os comandos (ou regras) representam vocábulos da linguagem do tutor e guardam dentro de si significados cognitivos mais finos que os textos das lições. Para permitir que esse vocabulário seja expandido o sistema possui um interface própria para o registro de comandos. A Fig. 22 mostra um exemplo para inserção do comando *caixaTexto*. Esses comandos devem ser escritos na linguagem de script PHP.

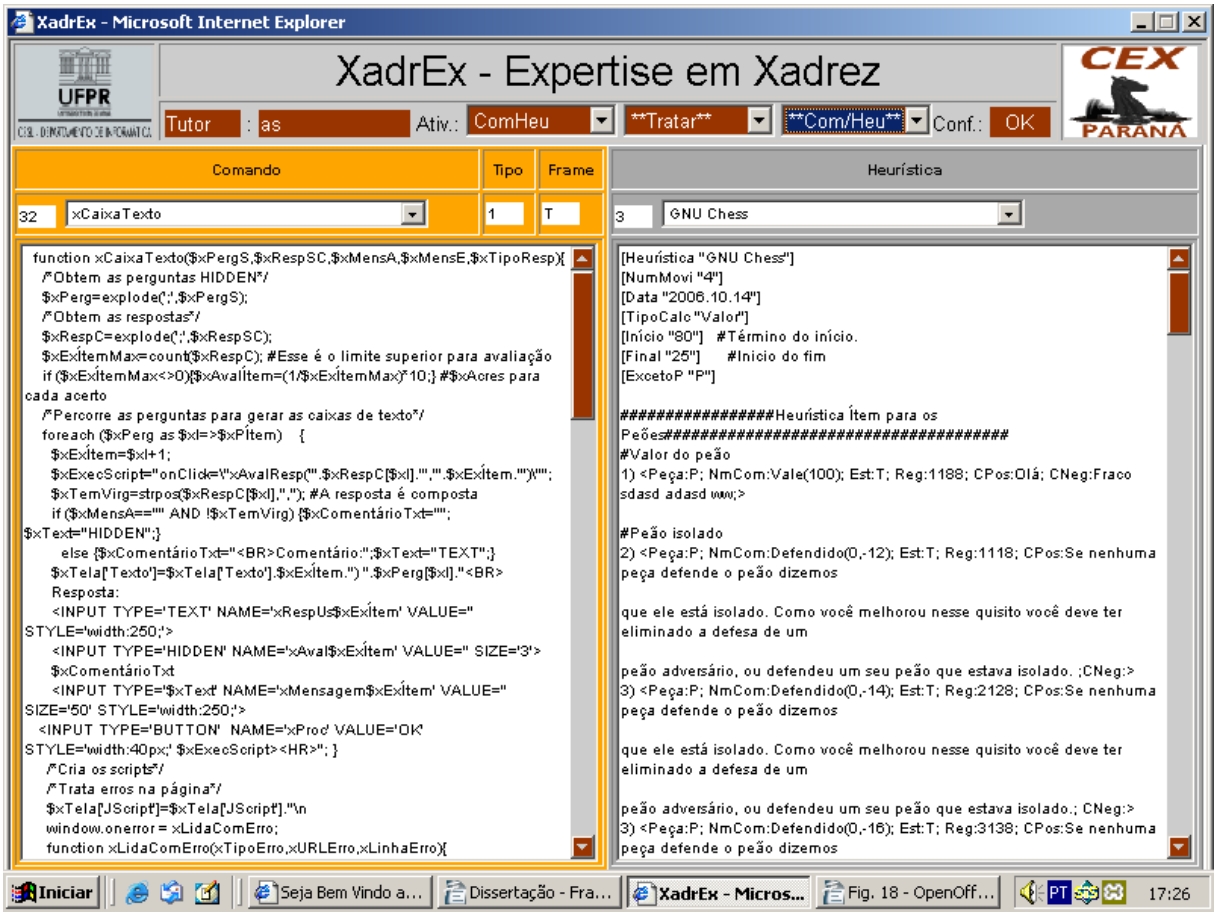

Fig. 22 – Registro de Comandos e Heurísticas

Uma outra forma de inserção de conhecimento experiencial é permitir que o tutor insira e comente partidas clássicas. A Fig. 23 – Registro de partidas mostra como inserir partidas através da digitação na caixa de texto ou informando um arquivo de partidas. A linguagem que o sistema está habilitado a interpretar é a PGN – Portable Game Notation. Esse padrão de

anotação é padronizado pelo FIDE[26] - Fédération Internationale des Échecs, e é possível encontrar na literatura especializada, ou na Internet, milhares de partidas registradas com esse padrão. A idéia aqui é que se o tutor pretende explicar uma abertura, ou um final, ou uma tática, ou até uma partida inteira, ele pode registrar a partida e comentá-la[27]. Posteriormente, seja através de um unidade instrucional, seja através de iniciativa própria, o aprendiz pode repassar os lances da partida e ler os comentários do tutor. A partida apresentada na Fig. 23 após processada é apresentada ao aluno como na Fig. 24 - Partidas clássicas comentadas pelo Tutor.

Fig. 23 – Registro de Partidas

[26]Em inglês, *World Chess Federation,* foi fundada em 20 de Julho de 1924, em Paris. É reconhecida pelo Comitê Olímpico Internacional (COI) como a responsável pela organização do Xadrez e dos campeonatos internacionais em níveis continentais. Em 1999, foi reconhecida pelo COI como uma federação de esporte internacional.

[27] Usando o padrão PGN.

Fig. 24 – Partidas clássicas comentadas pelo Tutor

4.3.3 Estilo[28]

As unidades instrucionais, quando escritas apenas com as marcações HTML, apresentam limitações quanto ao estilo de apresentação. Primeiramente, o controle de estilo desta linguagem é muito limitado, a ponto da comunidade ter criado a especificação CSS – Cascading Style Sheet. Em segundo lugar, essa linguagem exige que o tutor especifique o estilo para cada lição e exercício, o que nem sempre é desejável já que o tutor pode querer manter um padrão para uma dado módulo, ou até para todo um curso. Para facilitar a construção de estilos e permitir sua generalização por várias unidades instrucionais, o sistema permite o registro de estilos usando a linguagem CSS com as seguintes características:

a) Quando se referir a área *Texto,* o tutor escreve o nome padrão da classe *.UITxt*. Por exemplo: *.UITxt H3{color:red; text-align:center;}* significa aplicar os estilos especificados no título *H3* quanto o texto for direcionado para a área *Texto*.

[28] É entendido aqui como um conjunto de propriedades de formatação de texto.

b) Quando se referir a área *Gráfico,* o tutor escreve o nome padrão da classe *.UIGraf*. Por exemplo: *.UIGraf {background-color:Darkgray;font-size:10;font-family:Arial;}* significa aplicar os estilos especificados na área *Gráfico*.

c) Para controlar o aspecto do tabuleiro o tutor utiliza o nome de classe reservada *.Tabuleiro*. Por exemplo: *.Tabuleiro{DimTabu:325; CorCasaB:255 255 100; CorCasaP:99 76 32; CorPeçaB:0 120 0; CorPeçaP:25 0 0; Sombra:1.5 128 128 128; Nome:s}* significa aplicar os estilos especificados ao desenho do tabuleiro.

d) Para controlar o aspecto dos comentários o tutor utiliza o nome da classe *.UIComent*. Por exemplo: *.UIComent {height:140px; width:390px; left:0px; top:340; background-color:Darkkhaki;}* aplica os estilos especificados na área de comentário.

As especificações de estilo, definidas através desta linguagem, são associadas a uma unidade instrucional definida pela escolha de um Curso, um Módulo e uma Lição nas caixas de texto correspondente (Ver Fig. 18 – Interface para o registro de unidades instrucionais). Entretanto, para permitir a generalização de estilos, acrescenta-se a opção *Todos* tanto para Curso, quanto para Módulo, quanto para Lição. Isso dá grande flexibilidade ao tutor para generalizar suas especificações de estilo. Essa facilidade traz um esforço adicional ao sistema quando da associação de uma dada unidade instrucional ao seu estilo, pois, se não houver um estilo para ela, é necessário verificar se há estilos genéricos para o Módulo, ou Curso ou Tutor ou XadrEx a qual a UI pertence, antes de deixar por conta do navegador – como última instância - a aplicação do seu estilo padrão.

4.3.4 Conhecimento Heurístico

O conhecimento heurístico corresponde ao nível mais fino – mais específico – do sistema. É um tipo de conhecimento experiencial que, devido à sua importância, é destacado. O conhecimento heurístico, quando especificável, representa uma parte dos conhecimentos experienciais que levam o tutor a ser um perito em Xadrez. Sua quantificação e apropriação pelo sistema é fundamental, pois permite a criação de um jogo contra o aluno, serve de base para avaliação de suas jogadas e fornece argumentos para a criação de um diálogo tutorial. A codificação desse conhecimento, em sistemas computacionais que jogam Xadrez, é realizada no código do programa e, portanto, não pode ser mudada por ações do usuário do sistema. Essa abordagem não é muito útil em um STI cujo objetivo é permitir ao tutor especificar sua

própria heurística. Entretanto, a criação de interfaces para o registro de heurística não é muito fácil (Feitosa, 2006). Para contornar a dificuldade de criação dessa interface, o sistema disponibiliza uma linguagem textual para especificação de heurística[29]. Para maior clareza, a Fig. 25 reproduz um fragmento da heurística apresentada na Fig. 22 – Registro de Comandos e Heurísticas .

```
[Heurística "GNU Chess"]
[NumMovi "4"]
[Data "2006.10.14"]
[TipoCalc "Valor"]
[Início "80"]   #Término do início.
[Final "25"]      #Inicio do fim
[ExcetoP "P"]

######################Heurística Item para os Peões ########################
#Valor do peão
1) <Peça:P; NmCom:Vale(100); Est:T; Reg:1188; CPos:Olá; CNeg:Fraco sdasd adasd
ww;>

#Peão isolado
2) <Peça:P; NmCom:Defendido(0,-12); Est:T; Reg:1118; CPos:Se nenhuma peça
defende o peão dizemos que ele está isolado. Como você melhorou neste quisito você
deve ter eliminado a defesa de um peão adversário, ou defendeu um seu peão que
estava isolado. ;CNeg:>

..... Continua por mais 58 heurística item.
```

Fig. 25 – Exemplo de linguagem de construção de heurística do XadrEx - Fragmento

Para interpretar este texto, deve-se considerar apenas o que se encontra entre [ ] (colchetes) e entre as marcas < >, pois o texto restante é considerado como comentário com finalidade documentacional. Os textos entre [ ] (colchetes) são dados do corpo da heurística onde identifica-se pares *chave "valor"* que representam nomes de campos e seus respectivos valores. Para cada corpo da heurística, pode-se ter vários[30] itens heurísticos, cada um deles limitado por um par < >. Dentro destas marcas, encontra-se pares *chave:valor* separados por ponto e vírgula (;) os quais também representam nomes de campos dos itens heurísticos e seus respectivos valores. As tabelas 1 e 2 apresentam o significado desses campos.

[29] Assumindo o risco que suas idiossincrasias dificultem seu aprendizado pelos tutores.
[30] Como vimos anteriormente, o ícone dos jogadores automáticos – Deep Blue – tinha 8.000.

Campo	Valor	Significado
Heurística	GNU[31] Chess	Um texto para identificar a heurística.
NumMovi	4	Identifica o número de movimentos a pesquisar pelo algoritmo Minimax no cálculo da jogada da máquina ou na avaliação do aluno.
Data	2006.10.14	Data que a heurística foi registrada.
TipoCalc	Valor	Tipo de apuração do valor total da disposição inicial padrão. Caso igual a Valor então soma-se o valor das peças definidas nos itens heurísticos. Caso igual a Peça então conta-se o número de peças no tabuleiro. E, caso igual a Jogada considera-se o número de jogadas.
Início	80	Significa o estágio inicial do jogo. Caso TipoCalc = Valor ou Peça, identifica o percentual do total obtido em relação a disposição de peças inicial padrão. Caso TipoCalc = Jogadas indica até que número de jogada podemos considerar como Inicio do jogo.
Final	25	Significa o estágio final do jogo. Caso TipoCalc = Valor ou Peça, identifica o percentual do total obtido em relação a disposição de peças inicial padrão. Caso TipoCalc = Jogadas indica a partir de que número de jogadas podemos considerar como final do jogo.
ExcetoP	P	Lista de peças, separadas por vírgula, que não deve ser considerado nos cálculos para TipoCalc = Valor ou Peça.

Tab. 1 – Dados do corpo de uma heurística

Campo	Valor	Significado
Peça	P	Identifica a peça que será avaliada. Por exemplo: Peão.
NmCom	Vale(100)	Identifica o comando que deve ser obtido e avaliado caso a peça seja do tipo Peça, o jogo esteja no estágio Est, e a peça esteja posicionada na região Reg.
Est	T	Identifica em qual estágio do jogo o item heurístico deve ser aplicado. Pode ser : T(odos), I(nicio), M(eio) ou F(im).
Reg	1188	Identifica em qual região do tabuleiro o item heurístico se aplica. Por exemplo, o valor 1188 significa que vale da coluna 1, fila 1 até a coluna 8, fila 8 e, portanto, para todo o tabuleiro.
CPos	Excelente lance. Esse ganho vai ajudá-lo no final do jogo.	Comentário que o tutor quer agregar ao comentário do sistema quando houver melhora quanto a este item heurístico.
CNeg	A perda desse peão pode prejudicá-lo no final do jogo.	Comentário que o tutor quer agregar ao comentário do sistema quando houver piora quanto a este item heurístico

Tab. 2 – Dados dos itens de uma heurística

Especificações de heurísticas podem ser associadas a unidades instrucionais até o nível de exercício. Esse nível de detalhamento se justifica, pois um exercício que trata de finais de Bispo e Rei versus Rei, por exemplo, não necessita de item heurístico que trata de características do Cavalo ou da Torre; enquanto que, numa partida completa é importante todos os itens heurísticos que o tutor puder reunir. Por outro lado, é importante permitir ao tutor generalizar uma heurística para várias unidades instrucionais. Assim, de forma semelhante ao estilo, existe a opção Todos para Curso, Módulo, Lição e Exercício de forma que o tutor possa generalizar a aplicação da sua heurística para várias unidades instrucionais.

[31] Veja o anexo II – Heurística do GNU Chess

Em cada jogada do XadrEx, para possibilitar o cálculo da FAE nos nós terminais da árvore de busca, ou a cada lance do aluno para avaliação de sua tática, a heurística construída pelo tutor deve ser avaliada. Primeiramente, o sistema identifica e obtém a heurística associada àquela UI, traduz a especificação da heurística para uma matriz campo valor, e calcula o estágio da partida. Depois, faz uma busca recursiva, semelhante àquela vista na geração das lições e exercícios, de forma a criar a classe Heurística com os comandos necessários para o cálculo do valor de um tabuleiro. Assim, toda vez, durante uma transação[32], que for necessário o cálculo do valor da FAE, o sistema tem os elementos para fazê-lo.

Para efetuar o cálculo do valor do tabuleiro (FAE), o sistema percorre cada posição do tabuleiro e verifica se há peça e, havendo, qual é o tipo da peça. Daí, procura por comandos disponível na matriz Heurística cuja aplicabilidade coincida com essa peça, com o estágio do jogo, e com a região em que se encontra a peça. Se encontrar algum comando executa-o e seu resultado é somado ao valor da FAE. Como os valores bons para a corQueJoga são ruins para seu adversário, e vice-versa, os cálculos sempre consideram como positivo os valores para a corQueJoga e negativo para a cor contrária.

4.4 CONHECIMENTO PEDAGÓGICO

O conhecimento pedagógico diz respeito à forma como o conhecimento do domínio são ministrados ao aluno. Num nível global essa forma passa pela criação de um currículo, e num nível local, depende de uma retroalimentação automática às ações do aluno e de um diálogo tutorial efetivo. Esta parte do trabalho aborda como o XadrEx trata esses aspectos.

4.4.1 Currículo, Pré-requisito e Grau de Dificuldade

A seção 2.2.1 apresentou um currículo, na visão de Lesgold, constituído de 3 níveis: O nível do Meta-assunto, o nível da Rede de metas e o nível do Conhecimento.

No XadrEx o nível do Meta-assunto é chamado de Curso, o nível da rede de metas de Módulo e o nível do Conhecimento de Unidade Instrucional que agrega Lições e Exercícios.

[32] No caso da avaliação das táticas do aprendiz esse cálculo é feito 2 vezes; mas no cálculo de melhor jogada para o XadrEx ou para o Aprendiz esse cálculo pode ser feito milhares de vezes. Contudo, a geração da classe Heurística e o cálculo do estágio do jogo é feita só uma vez em cada transação.

Na implementação de tais conceitos, as relações de composição[33] e dependência entre identidades[34] foram simplificadas através das seguintes 3 hipóteses:

a) A relação de composição é uma árvore:- As relações de composição são exclusivas de tal sorte que um módulo pertence a um e só um curso e uma UI pertence a um e só um módulo. Não se admite, portanto, que um módulo seja parte de dois cursos ou uma UI seja parte de dois módulos.

b) A relação de dependência só existe entre irmãos na árvore de composição:- Ou seja, a lição 1 do módulo 1 pode depender da lição 2 do módulo 1, mas nunca depender da lição 2 do módulo 2.

c) A relação de dependência é uma árvore:- Ou seja, um curso, módulo ou UI deve depender somente de um e só um curso, módulo ou UI.

A hipótese simplificadora c) permite implementar a relação de dependência através de um simples atributo - Pré Requisito - que indica qual é o pré requisito para essa entidade porém, traz o inconveniente de não ordenar os vários filhos de uma dada entidade E na árvore de dependência. Para contornar esse inconveniente e permitir maior flexibilidade na organização do currículo, o sistema permite a inclusão do grau de dificuldade da entidade, e assume que entidades mais fáceis sejam apresentadas em primeiro lugar, possibilitando assim, uma ordenação pedagógica dos vários filhos da entidade E na árvore de dependência. A partir do exposto, o sistema define um conjunto de cursos, módulos e UI pedagogicamente ordenadas, a partir de pré requisitos e graus de dificuldade para serem apresentadas ao aprendiz, isto é, um seu currículo.

A partir da comparação entre as avaliações realizadas nas práticas com um valor objetivo, estabelecido pelo tutor, o sistema define a próxima lição para ser apresentada ao aluno, e assim sucessivamente até o término do currículo.

4.4.2 Retroalimentação Imediata

O sistema provê uma retroalimentação imediata às ações do aluno, através de um comentário de suas ações. A Fig. 26 mostra uma tela com um exemplo de um diálogo típico. O aprendiz pode escolher entre: a) Nenhum comentário, ou b) Comentário Tático, ou c)

[33] No sentido empregado na Análise Orientada a Objetos
[34] Um Curso, um Módulo ou uma UI.

Comentário do Lance, ou d) Os dois tipos de comentário. Abaixo, descreve-se as característica de cada um deles.

Fig. 26 – Partida Aprendiz x XadrEx com comentários sobre o lance

4.4.2.1 Comentário tático

Neste tipo de comentário, o sistema avalia o reflexo da tática do aprendiz para uma altura de visada igual a 1 e tece comentários, conforme a variação dos valores heurísticos do tabuleiro antes e após a jogada. O comentário tático é realizado através do preenchimento de espaços (*slots*) em um texto padrão, conforme o resultado da comparação dos valores obtidos para as heurísticas antes e após o lance. A Fig. 27 repete, para maior clareza, os comentários da Fig. 26, os quais são analisados a seguir.

Lance 12 das Brancas -> Da4: *ap* jogou: Da4 **Xeque**. Esse foi um bom lance, pois aumentou sua pontuação de 503 para 519 pontos (+3%). Aspectos que devem ser observados:

1) Peões *defendido* aplicável em quaisquer estágio do jogo e na região 3138 do tabuleiro, aumentou 16 pontos. Estude a(s) posição(ões): 34 (de -16 para 0).

Tutor: Se nenhuma peça defende o peão dizemos que ele está isolado. Como você melhorou nesse quisito você deve ter eliminado a defesa de um peão adversário, ou defendeu um seu peão que estava isolado.

-->O desdobramento a LP do seu lance revelou ser um péssimo lance(0%). Um lance mais adequado seria d3

Lance 11 das Pretas -> Cf4: *XadrEx* jogou: Cf4

Fig. 27 – Exemplo de comentário no XadrEx

Os textos "**Lance 11 das Pretas** -> Cf4: *XadrEx* jogou: Cf4" ou "**Lance 12 das Brancas** -> Da4: *ap* jogou: Da4 **Xeque**." são comentários padrão e não podem ser removidos, pois eles simplesmente descrevem o andamento da partida. O texto "*XadrEx*" é o nome do programa e "*ap*" é o userName escolhido pelo aprendiz no seu cadastramento. O texto "*Cf4*" é o movimento da peça, realizada pelo XadrEx, e significa que o Cavalo foi movido para a coluna *f*, fila *4*. O comentário tático, propriamente dito, compreende o texto, que vai de "Esse..." até "...estava isolado.". A Tabela 3, mostra a estrutura básica do comentário tático com os slots a serem preenchidos, os quais são apresentados entre os símbolos menor (<) e maior (>) e os seus respectivos significados.

Do ponto de vista sintático, comentários gerados através de preenchimento de *slots* são bem mais simples de implantar do que aqueles apresentados na seção 2.5.2. mas, entre outros, tem o inconveniente de gerar comentários previsíveis que se tornam, eventualmente, cansativos para o aprendiz. Entretanto, a abordagem adotada no XadrEx evita a previsibilidade através de variações no texto conforme as variações de valores das heurísticas, de variações no número e tipos de comandos associados aos itens heurísticos que foram afetados pelo lance do aprendiz, e o acréscimo ou não de comentários do tutor. Do ponto de vista semântico, observa-se que a tática do aluno não é avaliada no paradigma *model-tracing,* mesmo porque o sistema não tem como determinar com precisão o lance certo. Porém, ao apontar quais as posições do tabuleiro, sob que critério e com que intensidade foram afetadas por uma jogada, esses comentários atuam no sentido de "*provocar os estudantes a considerar e questionar as justificativas e implicações de suas crenças*". No Lance 12 das Brancas do

exemplo acima, observa-se que o aprendiz move sua Dama da posição *d1* para a posição *a4* com um Xeque[35], mas além disso, essa jogada defende o peão que estava sem defesa em *34* (c4). O sistema chama a atenção para esse fato, que talvez não fizesse parte das crenças do aprendiz quando ele fez aquela jogada.

Estrutura básica do comentário	
Esse foi <QualidadeDoLance> lance pois <AumentouDiminuiu> sua pontuação de <HeuAntes> para <HeuApós> pontos (<Delta>). Aspectos que devem ser observados: **<NumOrdem>)** <Peça> <Comando> aplicável <Estágio> do jogo e <Região> do tabuleiro, <DeltaItem> pontos. Estude a(s) posição(ões): <ListaPosEDeltaHeu>. <AcréscimoDoTutor>	
Significados	
Slots	**Comentário**
QualidadeDoLance	Pode valer “um bom”, “um ótimo” e etc.. conforme o valor de Delta.
AumentouDiminui	Pode valer ‘aumentou”, ou “diminuiu”. Caso Delta=0 então todo o texto até o ponto é : "Sua pontuação não variou. Foi de <HeuAntes> para <HeuApós>."
HeuAntes	Valor do tabuleiro antes do lance.
HeuApós	Valor do tabuleiro após o lance.
Delta	Valor em % da variação do valor do tabuleiro.
NumOrdem	Número de ordem dos comentários.
Peça	Tradução da abreviação da peça para seu nome. Por exemplo: P é substituído por Peões, D por Dama e etc.
Comando	Nome do comando indicado pelo tutor na heurística item. Exemplo: *defendido*.
Estágio	Indica o estágio onde aquele comando é aplicável e é obtido do item heurístico registrado pelo tutor. Pode valer, por exemplo: “em quaisquer estágio”, “no inicio” , “no meio”, “no final”.
Região	Indica a região onde aquele comando é aplicável e é obtido do item heurístico registrado pelo tutor. Pode valer “na região x” ou, no caso da região ser 1188, vale “em quaisquer região”.
DeltaItem	Variação do item heurístico obtido pela soma de todas as posições que coincidem com as especificações do item heurístico. Pode valer “aumentou n” ou diminuiu n”. A igualdade não gera comentário, pois a ação do aluno não afetou esse critério, ou se anularam.
ListaPosEDeltaHeu	Apresenta uma lista das posições e suas variações heurísticas que contribuíram para o valor de DeltaItem. Por exemplo: “34 (de -16 para 0)” significa que antes do lance a posição 34 contribuía com –16 e após o lance contribui com 0.
AcréscimoDoTutor	Se a variação do valor do tabuleiro for positiva então o sistema agrega o comentário positivo, se houver algum, registrado pelo tutor na especificação da heurística. Se for negativo, agrega o comentário negativo.

Tab. 3 – Detalhes da formação do comentário no XadrEx

[35] Não há um comentário para o Xeque porque o Tutor não criou uma Heurística item para o Xeque.

4.4.2.2 Comentário do lance

Neste tipo de comentário, o sistema estima a melhor jogada para o aprendiz, através da utilização do algoritmo Minimax e tece comentários, a partir da comparação entre o resultado obtido para o melhor lance e aquele realizado pelo aluno. Como frisado anteriormente, a árvore de jogos para o Xadrez é muito grande, inviabilizando uma busca exaustiva até um nó de fim de jogo para definir o melhor lance e, por isso, a definição da melhor jogada depende da estimativa do valor do tabuleiro na profundidade de visada estabelecida pelo tutor para aquela heurística. Portanto, a correção do lance estimado depende da qualidade da heurística criada pelo tutor e da profundidade de visada que, por sua vez, depende da capacidade de processamento do computador, o que nos leva sempre a uma imprecisão no cálculo do melhor lance. Ressalvadas essas imprecisões, o algoritmo Minimax foi modificado para devolver além do melhor lance, uma lista com todos os lances possíveis na altura de visada 1 e seu respectivo valor Minimax[36]. De posse desta lista, é possível criar o diálogo que se inicia em → apresentado na Fig. 27 que mostra a avaliação do afastamento quantitativo do melhor lance e também qual o melhor lance.

4.5 MODELO DO ESTUDANTE

As informações sobre o comportamento do aprendiz, para uma ação tutorial efetiva, são divididas, conforme o conhecimento avaliado em: a) Avaliação do conhecimento declarativo, e b) Avaliação de Táticas e Lances. Em qualquer dos casos, as avaliações serão realizadas sempre que as ações do aprendiz forem realizadas dentro de uma UI, pois seus resultados repercutirão no andamento do aluno ao longo do seu currículo.

4.5.1 Avaliação do Conhecimento Declarativo

As informações, que propiciam a avaliação do conhecimento declarativo do aluno, serão geradas no nível dos exercícios de fixação das lições. Esses testes do tipo múltipla escolha, preenchimento de caixa de texto, e etc., são definidos pelo tutor através da proposição do exercício, do resultado esperado, da respectiva ação tutorial e dos comandos

[36] Como o mecanismo de poda de um nó depende do valor obtido para o nó tio (irmão do pai) é imediato demonstrar que os nós na altura de visada 1 não serão podados e portanto sempre possuirão um valor minimax.

necessários. Uma ação tutorial típica nesse nível é a emissão de comentários pré-gravados, com felicitação pelo sucesso, ou recomendação de revisão dessa ou de outra lição do currículo. Na Fig. 21, por exemplo, o conhecimento do aluno é avaliado através de preenchimento de caixas de texto. Caso o aluno clique no botão Confirme e o valor que ele digitou na caixa Resposta esteje correto, uma mensagem de felicitação aparece na caixa Comentário. Após todas as respostas serem fornecidas, o sistema apura a proporção de acertos e, se estiverem acima do previsto para aquela lição, então o sistema apresenta a lição seguinte do currículo do aprendiz.

4.5.2 Avaliação de Lances

Caso o tutor inclua em uma UI um exercício que envolva um jogo contra a máquina, o sistema avaliará a ação do aprendiz automaticamente. O sistema utiliza o algoritmo Minimax modificado para obter a lista de lances possíveis para o aprendiz e seu respectivo valor Minimax. De posse destes valores, o sistema normaliza o valor da jogada do aprendiz para uma nota de 0 a 10 através da seguinte fórmula:

NotaAluno = (vlrJogadaA – vlrPiorJogada) / (vlrMelhorJogada – vlrPiorJogada) * 10

onde: vlrJogadaA = Valor Minimax para a jogada do aluno,
vlrPiorJogada = Menor valor Minimax entre as jogadas possíveis,
vlrMelhorJogada = Maior valor Minimax entre as jogadas possíveis.

Esta fórmula é suficientemente geral para não depender dos valores que cada heurística dá aos seus elementos. Por exemplo, uma heurística pode valorizar o Peão como 10, uma outra pode valorizá-lo como 100 e, considerando somente o valor do Peão, o resultado da primeira será menor que o da segunda para um mesmo tabuleiro, mas esses fatos não modificarão a avaliação do aluno. Como o tutor usará essas informações para compor a avaliação do aluno naquela UI visto que uma partida contra a máquina envolve vários lances ainda não está muito claro e dependerá das características dos comandos a serem criados pelos tutores.

4.6 OUTROS SERVIÇOS

Além dos serviços ligados ao registro e à comunicação do conhecimento prevê-se outras funções de finalidades administrativas, tais como : controle de acesso (senha), registro de aprendizes e tutores, pré-requisito e graus de dificuldade entre cursos de tutores diferentes, estatística de uso do sistema e etc..

4.7 CENÁRIO DE UTILIZAÇÃO

Os principais atores que interagirão com o XadrEx são: Mediador, Tutor e Aprendiz.

Na Fig. 28, apresenta-se uma visão geral dos principais atores e, a seguir, os seus papéis na interação com o sistema.

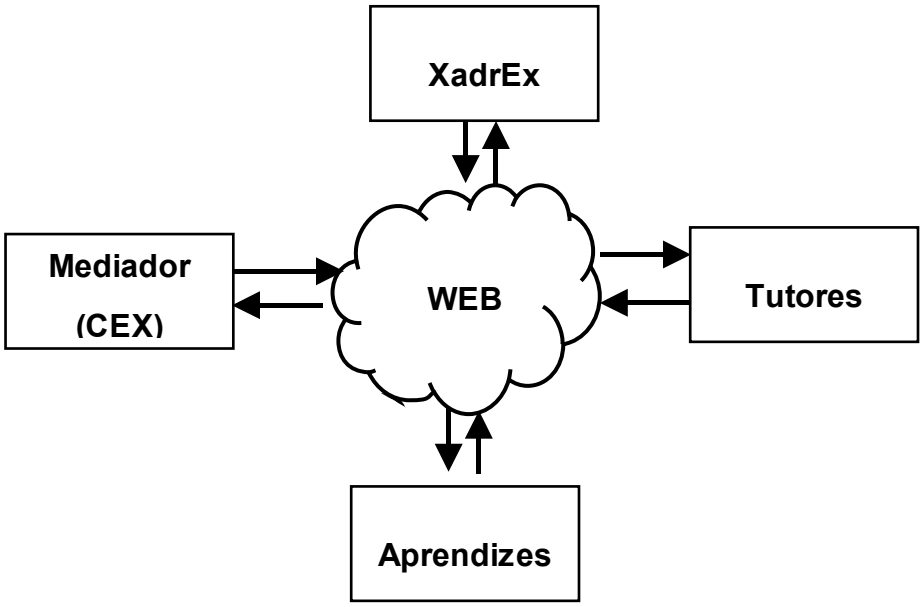

Fig. 28 – Os principais atores de uma rede de aprendizagem

O Mediador estará estabelecido no CEX – Centro de Excelência em Xadrez entretanto, devido ao modelo de processamento adotado, ele pode estar em qualquer lugar. A função do mediador é :

- Planejar e controlar o uso do sistema por Tutores e Aprendizes quanto a configuração do sistema, quantidade de alunos e tutores inscritos, volume da base de dados, expurgos e etc..

- Verificar a qualidade das contribuições oferecidas pelos tutores, apontando falhas, agradecendo a participação, sugerindo melhor estruturação de assuntos, sugerindo aos tutores integração de seus trabalhos no caso de assuntos comuns.
- Avaliar a qualidade dos cursos através de pesquisa junto aos alunos ou de consultas ao sistema sobre seus comportamentos, tais como: desistências, progressos, níveis de acerto e etc..

Os tutores estarão estabelecidos no CEX entretanto, devido ao modelo de processamento adotado, ele pode estar em qualquer outra instituição. A função do tutor é :

- Criar unidades instrucionais organizadas em Cursos, Módulos e Lições para serem utilizadas pelos aprendizes.
- Criar para cada lição os exercícios que permitam sua prática e a avaliação do conhecimento adquirido pelo aprendiz.
- Criar comandos que permitam o controle da área gráfica durante a exibição das lições e dos exercícios e expandam a linguagem do tutor.
- Registrar partidas clássicas ou de sua lavra, e comentar, se necessário, os seus movimentos.
- Registrar heurísticas para o algoritmo Minimax.

O aprendiz é o usuário final de todo o processo e o destinatário da informação. Espera-se que os aprendizes estejam estabelecido no CEX entretanto, devido ao modelo de processamento adotado, ele pode estar em qualquer outra instituição. A função do aprendiz é :

- Criar, manual ou automaticamente, um currículo para seu uso nas unidades instrucionais disponíveis para utilização.
- Estudar as unidades instrucionais do seu currículo e fazer os exercícios solicitados pelos tutores.
- Estudar partidas clássicas.
- Jogar contra a máquina.

5. RESULTADOS OBTIDOS

O sistema encontra-se em fase de desenvolvimento e testes piloto foram efetuados. Constatou-se que é possível criar unidades instrucionais organizadas em Cursos, Módulos, Lições e Exercícios, com rapidez, bom controle de estilo e com apresentação de slides. Também, verificou-se que o registro de heurísticas possibilitaram o desenvolvimento de um jogo do aprendiz contra a máquina com retroalimentação automática em toda a partida (inicio, meio e fim). Neste capítulo, aborda-se as possíveis repercussões do sistema, após a sua conclusão e implantação em entidades como o CEX e/ou escolas públicas ou privadas.

5.1 MAIOR DISPONIBILIDADE E RAPIDEZ NOS CURSOS DE XADREZ

Embora existam alguns programas para treinamento disponíveis, inclusive para download, no site do CEX, esses programas são versões de demonstração, que abordam situações por demais específicas, sem organização didática e sem ação tutorial. Desta forma, na maioria dos casos, os cursos de Xadrez são ministrados de forma presencial com uma relação aluno/professor bastante baixa, implicando em alto custo e indisponibilidade. O sistema apresentado permitirá que tutores criem novos conteúdos, aproveitando a capacidade de autoria desta ferramenta e após sua implantação definitiva, espera-se que a comunidade de tutores registre seus materiais de treinamento, aumentando a disponibilidade de cursos e acarretando uma relação aluno/professor maior. Outrossim, devido ao modelo de processamento de dados adotado, a nova ferramenta vai permitir que esses cursos sejam rapidamente construídos e rapidamente disponibilizados para os aprendizes.

5.2 MAIOR QUALIDADE DESSES CURSOS

O sistema proposto apresenta uma grande melhoria na qualidade dos recursos para aprendizagem de Xadrez por computador. No CEX, por exemplo, é possível o treinamento através de partidas on-line, mas isso só beneficia as pessoas mais experientes, não havendo nada para os novatos ou leigos. Além disso, o treinamento sem ação tutorial, mesmo para os mais experientes, pode originar analogias erradas. Contrariamente, um sistema que avalia, interpreta e comenta a jogada do aluno fornece uma carga cognitiva superior pois permite que

o aprendiz compile seus novos conhecimentos de forma correta, sem risco que analogias erradas, conduza-o a erros conceituais.

5.3 MAIS ADEPTOS E MAIOR PARTICIPAÇÃO NOS CAMPEONATOS

Devido a sua arquitetura ser orientada para a WEB, o sistema XadrEx permite uma total independência sobre a localização do servidor, dos tutores ou dos alunos. Isto é, um perito em Xadrez que queira registrar um curso, ou um aluno que queira aprender, não precisam estar lotados na instituição mantenedora do sistema, por exemplo o CEX. Esse fato permitirá que os benefícios de um curso possam ser aproveitados por vários alunos, criando mais adeptos do jogo, gerando maior participação nos campeonatos, e difundindo o jogo de Xadrez em todo o Brasil.

6. CONCLUSÃO

Esta dissertação apresentou um artefato de software que permite aos mestres enxadristas criarem cursos para o ensino de conhecimentos declarativos e experienciais na área de domínio do jogo de Xadrez.

6.1 RETROSPECTIVA

Esta proposta decorreu da constatação de que o jogo de Xadrez, em qualquer nível escolar e em qualquer idade, é útil para o desenvolvimento do raciocínio lógico e também como atividade de entretenimento haja visto as iniciativas governamentais no sentido de difundi-lo. Daí, pesquisou-se na bibliografia especializada produtos já desenvolvidos e testados. Essa pesquisa revelou que não há, mesmo a nível mundial – seja comercial, seja acadêmico - nenhum produto que possibilite a um tutor a construção de sistemas de aprendizagem do jogo de Xadrez. Além disso, verificou-se que diagnósticos automáticos do aluno, apesar de terem sido utilizados na programação de computadores e transformações algébricas, só foram testados em finais de jogos de Xadrez.

A partir destes resultados desenvolveu-se o XadrEx, um protótipo constituído de um conjunto de ferramentas e métodos para apoiar o ensino de Xadrez na fronteira entre os fundamentos e a perícia. Este sistema apresenta uma série de características diferenciadas, em relação aos existentes, destacando-se:

a) Independência do domínio : - A ferramenta possibilita que o tutor expanda os conteúdos cognitivos do sistema, seja no nível macro com a inclusão de novas lições, seja no nível intermediário com a inclusão de comandos, e seja no nível micro através da inclusão de heurísticas.
b) Retroalimentação imediata: - O sistema monitora as ações do aluno durante os exercícios práticos, ao longo de toda uma partida, e uma ação tutorial é prontamente acionada.
c) Grande abrangência : - A construção para ambiente Web habilita-o para uma grande faixa de classes de alunos e de tutores.

6.2 TRABALHOS FUTUROS

Entretanto, a construção de STI são complexos, caros e exigem grande dispêndio de tempo para sua realização. Durante o desenvolvimento deste protótipo, identificou-se quatro aspectos que não puderam ser explorados nesta pesquisa e devem ser objeto de trabalhos futuros.

6.2.1 Linguagem do Tutor

A linguagem disponível para o tutor incluir novos conteúdos é procedural e mais apropriada para programadores de computador do que para mestres enxadristas. Portanto, a maioria dos peritos enxadristas não sabem usá-la, e podem não querer aprendê-la, tornando impossível o registro do seu conhecimento no sistema por eles mesmos. Embora conteúdos possam ser criados por mestres enxadristas, e introduzidos por programadores, através de equipes multidisciplinares[37], é necessário novas pesquisas para desenvolver uma linguagem mais apropriada para o tutor que permita o seu trabalho autônomo, independente de suporte de programação.

6.2.2 Estruturação das Unidades Instrucionais

Embora no modelo atual, um tutor possa registrar um curso com só um módulo e uma só lição, a rígida estruturação das unidades instrucionais em curso, módulo, lição e exercícios, como implantada no XadrEx, pode inibir um tutor que queira participar apenas com uma unidade instrucional. Portanto, pesquisas adicionais devem ser realizadas para permitir que o tutor escreva uma única UI – menos trabalhosa - e agregá-la ao XadrEx.

Uma abordagem possível é pesquisar uma taxonomia[38] padronizada para o ensino de Xadrez e adotá-la no XadrEx. Há taxonomias padronizadas, por exemplo, para a área de Informática, tal como a criada pela ACM – Association for Computing Machinery com o nome de Computing Classification System, a qual permite a classificação de trabalhos na área de Informática. A existência de taxonomia semelhante para o Xadrez, e sua implantação no sistema, trará como benefício para o tutor uma maior facilidade na criação de unidades

[37] Semelhante às equipes de projeto tão comum no desenvolvimento de sistemas administrativos.
[38] Uma taxonomia acordada mutuamente pela comunidade de enxadristas que estabeleça uma terminologia e uma relação de subordinação de termos e para o qual os enxadristas aderem.

instrucionais atômicas, pois, após criá-la, ele pode inserí-la na organização de assuntos já preestabelecida pela taxonomia.

Outra abordagem possível é, através de uma profunda revisão em seus requisitos, posicionar o sistema como um LMS – *Learning Management System,* de forma que o tutor passe a escrever Objetos de Aprendizagem (Learning Objects), como vimos na seção 2.2.2, os quais interagem com o XadrEx através da comunicação em XML definidas dentro do padrão SCORM.- *Shareable Content Object Reference Model*. Nesse caso, ainda será necessário a criação de interfaces, que facilite a criação dos Objetos de Aprendizagem e permita a comunicação desses objetos com o XadrEx.

6.2.3 Interface com o Estudante

No módulo do estudante, a interface é muito despojada, sem efeitos multimídia por exemplo, o que pode ser desencorajador, principalmente para crianças e adolescentes. Então, torna-se necessário pesquisas mais aprofundadas sobre quais os elementos mais atraentes e eficazes para a construção da interface com o estudante.

6.2.4 Diálogo Tutorial

Embora o protótipo avalie as ações do aprendiz e apresente comentários críticos sobre suas táticas para "provocar os estudantes a considerar e questionar as justificativas e implicações de suas crenças", ele não permite a réplica do aprendiz. Entretanto, na seção 2.5 sugere-se que essa interação com o aluno reforça os aspectos cognitivos da lição ensinada como, por exemplo, quando empregamos diálogos do tipo Socrático semelhante àquele do sistema RUI. Assim, sugerimos novas pesquisas que verifique se o conjunto de informações obtidos a partir das ações do estudante e do conjunto de itens heurísticos fornecido pelo tutor, são suficientes para a geração de argumentos para o estabelecimento de um diálogo de longo prazo com o estudante.

REFERÊNCIAS BIBLIOGRÁFICAS

ADL Advanced Distributed Learning (ADL), Sharable Content Object Reference Model (SCORM®) 2004 3rd Edition Overview Version 1.0, 2006.

ANDERSON, J.R. Cognitive psychology and intelligent tutoring. Proceeding of the Cognitive Science Society Conference, Boulder, Colorado. 1984.

ANDERSON, J.R. The Architecture of Cognition. Harvard University Press, Cambridge, Massachusetts. 1983

ANDERSON, J.R., CORBETT, A. T., PATTERSON, E. G. Student Modeling and Tutoring Flexibility in the Lisp Intelligent Tutoring System. Conference on Intelligent Tutoring Systems, Quebec, Montreal. 1988. pag 83-106

BLESSING, B. B. A Programming by Demonstration Authoring Tool for Model-Tracing Tutors. International Journal of Artificial Intelligence in Education. 1997. 8, 233-261.

BOGATSCHOV, D. N. Jogos computacionais e de ação e a construção dos possíveis em crianças do ensino fundamental. Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Educação. 2001.

BURNS, B. D. The Effects of Speed on Skilled Chess Performance. Michigan State University. 2004

CAMPITELLIL, G., GOBET1, F., PARKER2, A. Struture and Stimulus Familiarity: A Sudy of Memory in Chess-Players with Functional Magnetic Resonance Imaging. The Spanish Journal of Psychology. 2005. Vol. 8, No. 2, 238-245.

CLANCEY, W. J. Knowledge-Based Tutoring – The Guidon Program. The MIT Press Cambridge, Massachusetts, London, England 1987.

De GROOT, A. D. (1978). Thought and choice in chess (2nd English ed.; first Dutch edition published in 1946). The Hague: Mouton Publishers.

DIRENE, A. Designing Intelligent Systems for Teaching Visual Concepts. International Journal of Artificial Intelligence in Education. 1997a. 8, 44-70.

DIRENE, A. Intelligent Training Shells for the Operation of Digital Telephony Stations. Universidade Federal do Paraná, Departamento de Informática. 1997

DIRENE, A., BONA, L., SILVA, F., dos SANTOS, G., GUEDES, A., CASTILHO, M., SUNYÉ, M., HARTMANN, C., de ANDRADE NETO, P., de MELLO, S., SUNYÉ NETO, J. e SILVA, W. Conceitos e ferramentas de apoio ao ensino de Xadrez nas escolas brasileiras. Em Anais do XXIV Congresso da Sociedade Brasileira de Computação : WIE - Workshop sobre Informática na Escola (Salvador, Brasil, Julho 2004), R. Macêdo, Ed., SBC, pp. 816–825.

DIRENE, A., SCOTT D. Identifying the component features of expertise in domains of complex visual recognition. Information Technology Research Institute Technical Report Series. Univ. of Brighton, Lewes Road, Brighton, UK. 2001

FEITOSA, A. R. M. Ensino e aprendizagem de estratégias do Xadrez por meio da autoria incremental de conceitos heurísticos. Proposta de dissertação (mestrado) - Universidade Federal do Paraná, Departamento de Informática. 2005

GADWAL, D., GREER, J. E., MCCALLA, G. Tutoring Bishop-Pawn Endgames: An Experiment in Using Knowledge-Based Chess as a Domain for Intelligent Tutoring. Department of Computacional Science. University of Saskatchewan, Saskatoon, Saskatchewan, Canada. 2000.

HARTMANN, C. M. Linguagem e ferramenta de autoria para promover o desenvolvimento de perícias em Xadrez. Dissertação (mestrado) - Universidade Federal do Paraná, Departamento de Informática. 2005

HYÖTYNIEMI, H., SAARILUOMA, P. Chess – Beyond the Rules. Control Engineering Laboratory. Helsinki Univ. of Tech.1999.

ISARD, S. What would you have done if...? Theoretical Linguistics, Vol 1, 1.974. pg. 233-256

LESGOLD, A. Toward a Theory of Curriculum for Use in Designing Intelligent Instructional Systems. Learning Issues for Intelligent Tutoring Systems, Springer-Verlag New York, Inc. New York, NY, USA, 1988. pag 114-37

LESGOLD, A., RUBINSON, H., FELTOVICH, P., GLASER, R., KLOPFER, D., WANG, Y. Expertise in a Complex Skill : Diagnosing X-Ray Pictures. Em M. Chi, R. Glasser, and M. Farr, editores, The Nature of Expertise. Lawrence Erlbaum. 1989

MARSLAND , T.A. Computer Chess and Search. Computing Science Department, University of Alberta, Edmon, Canadá. April 3, 1991

MARTINESCHEN, D. Conceitos e ferramentas para aquisição de conhecimento sobre heurística de jogos através de atividades competitivas. Proposta de dissertação (mestrado) - Universidade Federal do Paraná, Departamento de Informática. 2005

MURRAY, T. Authoring intelligent systems: An analysis of the state of the art. International Journal of Artificial Intelligence in Education, 10 (1999), 98–129.

PIANTAVINI, F. N. O. Jogo de regras e construção de possíveis : Análise de duas situações de intervenção psicopedagógica. Dissertação (mestrado) - Universidade Estadual de Campinas, Faculdade de Educação. 1999.

RIESENHUBER, M. An action video game modifies visual processing. TRENDS in Neurosciences Vol. 27 No. 2 February 2004.

RUSSEL, S., NORVIG, S. Inteligência Artificial. 2ª edição. – Rio de Janeiro : Elsevier, 2004.

SANTOS, G., DIRENE, A., Guedes A. L. P. Autoria e Interpretação de Soluções Alternativas para Promover o Ensino de Programação de Computadores. XIV Simpósio Brasileiro de Informática na Educação. 2003

SELF, J.A. Bypassing the Intractable Problem of Student Modeling. Conference on Intelligent Tutoring Systems, Quebec, Montreal. 1988. pg 107-123

SHARPLES, M. Computer-based tutoring of visual concepts: from novice to expert. Journal of Computer Assisted Learning. 1991 - 7. pg 123-132

TIRADO, A., SILVA, W. Meu Primeiro Livro de Xadrez. Curitiba : Expoente, 2005.

VANLEHN, K. Toward a Theory of Impasse-Driven Learning. Springer-Verlag New York, Inc. New York, NY, USA. 1988. Pag. 19-41. ISBN:0-387-96616-1

WENGER, E. Artificial Intelligence and Tutoring Systems : Computacional and Cognitive Appoches to the Communication of Knowledge. Morgan Kaufmann Publishers, Los Altos, CA, 1987.

WHITE, B. Y.; FREDERIKSEN, J. R. Quest : Qualitative Understanding of Electrical System Troubleshooting. Laboratórios BBN – ACM SIGART newsletter, Julho de 1985.

ZELHART, F., WALLINGFORD, E. A Survey of Intelligent Tutoring Systems and the Methods Used for Effective Tutoring. Intelligent Systems Laboratory. Department of Computer Science. University of Northern Iowa. Draft. 1994

ANEXO I –
NOTAÇÕES PARA O JOGO DE XADREZ[39]

[39] Baseado em http://en.wikipedia.org/wiki/Portable_Game_Notation e sites relacionados visitados em 22/11/2005

1) INTRODUÇÃO

O objetivo desse trabalho é o de descrever um padrão para a anotação de partidas, disposição de tabuleiros e descrição de jogadas durante uma partida de xadrez. Entre as notações possíveis, destaca-se a Portable Game Notation (PGN), a qual está entre as mais difundidas, além de ser extremamente popular entre os jogadores de xadrez. PGN está estruturada tanto para ser fácil de ler e escrever por usuários humanos, quanto para ser fácil para análise (parsing) e criação por programas de computador. Estas facilidades são conseguidas porque tanto os movimentos, quanto os dados relacionados são registrados em um formato constituido por textos ASCII, sem formatação, o que a torna acessível tanto para editores ASCII comuns, como para processadores de texto capazes de importar e exportar ASCII planos. Esses arquivos, gerados em texto plano, não possuem nenhum código de controle especial, envolvendo caracteres de escape, ou *carriage returns* e *linefeeds* para separaração de campos. Espaços embutidos (caracteres SPC) são usualmente desconsiderados quando da análise dos dados pelo computador. Esses arquivos recebem a terminação ".pgn".

A notação PGN descreve as partidas através dos seguintes tipos de dados:

1. Pares de marcas.
2. Descrição do movimento.
3. Resultado.

Vamos examinar cada um deles.

2) COMPONENTES DA NOTAÇÃO PGN

2.1 Pares de Marcas

Os pares de marcas x valor começa com um abre colchete ("[") inicial, seguido pelo nome da marca, e o valor da marca, para então terminar com um fecha colchete ("]"). Para distinguir o nome da marca do seu valor, o valor vem expresso entre aspas duplas. As sete primeiras são padronizadas, tanto quando a seu nome quanto quanto a ordem que devem aparecer noa arquivo texto. Juntas, estas sete marcas são conhecidas como STR (Regra das Sete Marcas). Essas sete marcas precisam aparecer antes de quaisquer outros pares de marcas que, eventualmente, sejam acrescentadas com o intuíto de expandir o padrão. As sete marcas são as seguintes e nesta ordem :

Event: O nome do torneio ou evento.

Site: A localização do evento, dado no formato "Cidade, Região PAIS", onde PAÍS é o código de 3 letras do Comitê Olímpico Internacional. Um exemplo é "New York City, NY USA".

Date: Começando com a data da partida, no formato YYYY.MM.DD. Para valores desconhecidos é utilizado "??".

Round: Número de ordem da partida entre os mesmos competidores durante esse torneio (Event).

White: O nome do jogador com as pedras brancas, no formato, "último nome, primeiro nome".

Black: O nome do jogador com as pedras pretas, no formato, "último nome, primeiro nome".

Result: O resultado do jogo. Pode assumir somente quatro valores: "1-0" (As Brancas ganharam), "0-1" (As Pretas ganharam), "1/2-1/2" (Empate), ou "*" (outro, por exemplo, o jogo está em andamento).

Com intuito de expandir o padrão, outros pares [Marca "valor"] podem ser acrescentados. Os pares mais comuns, os quais podem aparecer em qualquer ordem, são os seguintes:

Time: A hora local que o jogo começou, no formato "HH:MM:SS".

Termination: Dá mais detalhes a respeito do término do jogo. Pode ser: "abandonado", "julgamento" (resultado será determinado pelo julgamento de terceiros), "morto", "emergência", "normal", "infração de regras", "perda por tempo", ou "indeterminado".

FEN: A disposição inicial das peças no tabuleiro, em Forsyth-Edwards Notation (ver seção 3.). Pode ser usado para registrar uma partida que começou com uma disposição diferente da disposição inicial padrão. Se uma marca FEN é usada, um par de marca "SetUp" precisa aparecer e ter seu valor setado em "1".

2.2 Descrição dos Movimentos

Trata-se de um texto que descreve as jogadas realizadas durante o jogo. Esse texto é formado pelo número da jogada, o movimento das Brancas, o movimento das Pretas, e eventualmente um comentário.

Número da jogada

É o número de ordem das jogadas. Uma jogada é constituido pelo movimento das brancas (meia jogada ou um movimento) e pelo movimento das pretas (meia jogada ou um movimento).

Movimento

Para designar os movimentos das peças utiliza-se um texto na Notação Algébrica Padrão (SAN – Standard Algebric Notation). Esse padrão designa os movimentos da seguinte forma:

a) Uma letra de abreviação para o nome da peça que será jogada. Essas abreviações são K (King) para o Rei, Q (Queen) para a Dama, R (Rook) para a Torre, B (Bishop) para o Bispo, e N (Knight) para o Cavalo. Para o Peão foi convencionado uma abreviatura vazia. Por exemplo: Nf3 significa mover um cavalo para f3; c5 significa mover um peão para c5.

b) A letra "x" caso houver captura da peça adversária na casa de destino do movimento. Por exemplo: Bxe5 significa que um bispo captura a peça que estiver em e5.

c) Uma letra e um número para representar a casa de destino. Para estabelecer esta notação convenciona-se um tabuleiro com as pedras brancas na parte inferior e as pedras pretas na parte superior. Com essa disposição, numera-se, no sentido horizontal as colunas de a até h; e, no sentido vertical, numera-se as filas de 1 até 8. Assim, a casa mais a esquerda da primeira fila é a de número a1, enquanto que a mais a direita na oitava fila é a h8.

Se duas peças idênticas podem se mover para a mesma casa, a inicial da peça é seguida por:

a) Se ambas as peças estão na mesma fila, a coluna de saída;

b) Se ambas as peças estão na mesma coluna, a fila de saída.

c) Se ambas as peças estão em diferentes colunas e filas, o método (1) é escolhido.

Por exemplo, com dois cavalos um em g1 e outro em d2, ambos podem mover-se para f3, o movimento pode ser apropriadamente indicado como Ngf3 ou Ndf3. Com dois cavalos um em em g5 e outro em g1, os movimentos são N5f3 ou N1f3.

O roque do lado do rei (roque pequeno) é indicado pela seqüência "O-O"; o roque do lado da dama (roque grande) é indicado pela seqüência "O-O-O" (note que são O's maiúsculos e não os numerais 0's).

Promoção de peões são notados por um "=" seguido pela peça para o qual o peão foi promovido. Por exemplo: **f8=Q**. Entretanto, aceita-se sem a utilização do sinal "=". Por exemplo: **b8B**.

Um movimento que coloca o rei do adversário em xeque deve ter adicionado à sua notação o sinal de "+", ou *ch.* Da mesma forma, Xeque mate pode ser indicado com "#", ou "++".

Os movimentos podem ser listados em três colunas. Na primeira coluna, coloca-se o número da jogada, na segunda coluna o movimento das brancas, e na terceira o movimento das pretas. Por exemplo:

1. e4 e5
2. Nf3 Nc6
3. Bb5 a6

Opcionalmente, os movimentos podem ser representados sem a estruturação em colunas, e serem apresentados através de um texto contínuo. Por exemplo: 1. e4 e5 2. Nf3 Nc6 3. Bb5 a6.

Comentários

Após a especificação do movimento, pode-se, opcionalmente, inserir anotações que sugerem alternativas, ou comentam a jogada. Identifica-se 3 tipos possíveis para esses comentários:

a) Inseridos entre um abre chaves "{" e um fecha chaves "}". Opcionalmente, comentários que continuam até o final da linha podem se iniciar por um ";".

b) Comentários sem delimitadores. Caso o comentário se refira ao movimento das brancas o número do movimento deve ser reescrito e uma elipse (...) toma o lugar do movimento das brancas, por exemplo: 1. e4 e5 2. Nf3 As Pretas agora defendem seu peão 2. ... Nc6 3. Bb5 a6

c) Pontos de interrogação e de exclamação para classificar um movimento como bom ou ruim. Os símbolos, normalmente, usados são "?", "??", "!", "!!", "?!" and "!?". O símbolo é anexado imediatamente após o movimento. Por exemplo: Re7? ou Kh1!?. O uso dos símbolos de anotação são um tanto subjetivos, e diferentes anotadores, frequentemente, usarão símbolos diferentes. O significado mais comum dos símbolos são:

?: Erro : - Anexando um ponto de interrogação "?" após um movimento indica que o anotador acha que o movimento é pobre ou que não deveria ser jogado.

??: Erro mais forte : - O duplo ponto de interrogação "??" indica um erro mais forte. Movimentos típicos que podem receber duplo ponto de interrogação são: descobrir uma rainha que está sob ataque, ou se expor a um xeque mate.

!: Bom movimento : O ponto de exclamação indica um bom movimento. Movimentos típicos que recebem ! são novas aberturas, rupturas da defesa do adversário, sacrifícios e movimentos que evitam cair em armadilhas preparadas pelo adversário.

!!: Movimento brilhante : A dupla exclamação é usada para exaltar um movimento que o anotador acha que mostra a habilidade do jogador. Tais movimentos são dificeis de encontrar. Pode ser um sacrificio de grande quantidade de material, ou movimentos que a primeira vista parecem não ser intuitivo.

!?: Movimento interessante : O "!?" é um dos símbolos mais controversos. Diferentes livros fornecem variadas definições. Entre as definições, encontra-se: "interessante, mas talvez não seja o melhor movimento", "movimento digno de atenção", "movimento audaz" e "movimento arriscado". Movimentos típicos são sacrificios especulativos e a tentativa de um ataque perigoso que pode ser estrategicamente deficiente.

?!: Movimento dúbio : Esse símbolo é similar ao "!?" mas usualmente indica que o anotador acredita que o movimento seja objetivamente ruim, embora de dificil refutação. O "?!" é também frequentemente usado ao invés do "?" para indicar que o movimento não é de todo ruim. Um sacrifício que conduz a um ataque perigoso que o oponente deveria ser capaz de defender se ele jogasse bem pode receber um "?!".

2.3 Resultado da Partida

Após a enumeração dos movimento e seus eventuais comentários, e, caso o resultado do jogo definido no par [Resultado "valor"] for diferente de "*", o resultado é repetido no fim do texto.

2.4 Exemplo

Eis um exemplo de jogo no formato PGN :

```
[Event "F/S Return Match"]
[Site "Belgrade, Serbia JUG"]
[Date "1992.11.04"]
[Round "29"]
[White "Fischer, Robert J."]
[Black "Spassky, Boris V."]
[Result "1/2-1/2"]

1.e4 e5 2.Nf3 Nc6 3.Bb5 {This opening is called Ruy Lopez.} a6 4.Ba4
Nf6 5.O-O Be7 6.Re1 b5 7.Bb3 d6 8.c3 O-O 9. h3 Nb8 10.d4 Nbd7 11.c4 c6
12.cxb5 axb5 13.Nc3 Bb7 14.Bg5 b4 15.Nb1 h6 16.Bh4 c5 17.dxe5 Nxe4 18.Bxe7
Qxe7 19.exd6 Qf6 20.Nbd2 Nxd6 21.Nc4 Nxc4 22.Bxc4 Nb6 23.Ne5 Rae8 24.Bxf7+
Rxf7 25.Nxf7 Rxe1+ 26.Qxe1 Kxf7 27.Qe3 Qg5 28.Qxg5 hxg5 29.b3 Ke6 30.a3 Kd6
31.axb4 cxb4 32.Ra5 Nd5 33. f3 Bc8 34.Kf2 Bf5 35.Ra7 g6 36.Ra6+ Kc5 37.Ke1
Nf4 38.g3 Nxh3 39.Kd2 Kb5 40.Rd6 Kc5 41.Ra6 Nf2 42.g4 Bd3 43.Re6 1/2-1/2
```

3) FORSYTH-EDWARDS NOTATION

A seção 2. mostrou quais os componentes utilizados para a descrição de partidas de xadrez utilizando a notação PGN. Para representar uma particular disposição do tabuleiro em um jogo de xadrez, utiliza-se o padrão Forsyth-Edwards Notation (FEN).

O objetivo da notação FEN é prover toda informação necessária para reiniciar uma partida a partir de uma particular disposição. FEN é um sistema desenvolvido por Scottish, David Forsyth no século 19 e, posteriormente, aprimorada por Steven Edwards que a estendeu para uso em computadores. FEN é uma parte integrante da Portable Game Notation para jogos de xadrez, pois é utilizada para definir uma posição inicial no caso desta disposição ser diferente da padrão. Este padrão registra uma particular disposição de peças no jogo, usando uma linha de texto e um conjunto de caracteres ASCII. Um arquivo de texto com registros FEN deve ter a extensão ".fen".

3.1 Componentes da Notação FEN

Um registro na notação FEN contém 6 campos, separados por um espaço. Esses campos são os seguintes:

Disposição das peças

Imaginando o tabuleiro na perspectiva das brancas, isto é com as brancas na parte de baixo do tabuleiro, cada fila é descrita e separada das outra filas por uma barra (/). Essa descrição começa com a fila 8, na parte superior do tabuleiro, e termina com a fila 1, na parte inferior do tabuleiro.

O conteúdo de cada casa na fila é descrito usando letras maiúsculas ("KQRBNP") para as peças brancas e letras minúsculas ("kqrbnp") para as peças pretas. Casas desocupadas são anotadas usando dígitos de 1 para uma casa vazia até 8 para oito casas vazias. Por exemplo: ppB3T1 significa que nessa fila encontramos, da esquerda para a direita, dois peões pretos, 1 bispo branco, tres casas vazias, 1 torre branca e 1 casa vazia.

Cor ativa

Indica qual a cor que deve jogar. Se "w" significa que as brancas são as próximas a jogar, e se "b" significa que o próximo movimento é das pretas.

Disponibilidade do roque

Se nenhum dos dois lados pode efetuar qualquer dos dois tipos de roque, esse campo vale "-". De outro modo, esse campo pode ter uma ou mais letras com o seguinte significado: "K", as brancas podem rocar do lado do rei; e/ou "Q", as brancas podem rocar do lado da dama; e/ou "k", as pretas podem rocar no lado do rei; e/ou "q", as pretas podem rocar pelo lado da dama.

Casa alvo do En passant

Caso não seja possível o "en-passant", esse campo vale "-". Se o peão faz o movimento de 2 casas, registra-se nesse campo a posição “atrás do” peão, de forma a indicar que a cor que joga o próximo lance tem um "en-passant" para aquela casa.

Contador de movimentos (Halfmove)

Esse campo registra o número de movimentos, desde que o último peão avançou, ou desde a última captura. Caso esse número ultrapassar 50, qualquer dos adversários pode reclamar o empate baseado na regra do 50º movimento[40].

Número de jogadas (Fullmove)

O número de jogadas (full move). Ele começa em 1, e é incrementado após o movimento das pretas.

Exemplos de representação FEN

1. Representação FEN para a disposição inicial :

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - 0 1
```

2. Representação FEN após um movimento comum: 1. e4 :

```
rnbqkbnr/pppppppp/8/8/4P3/8/PPPP1PPP/RNBQKBNR b KQkq e3 0 1
```

3. E após : 1. ... c5 :

```
rnbqkbnr/pp1ppppp/8/2p5/4P3/8/PPPP1PPP/RNBQKBNR w KQkq c6 0 2
```

4. E após 2. Nf3 :

```
rnbqkbnr/pp1ppppp/8/2p5/4P3/5N2/PPPP1PPP/RNBQKB1R b KQkq - 1 2
```

[40]Regra segundo a qual qualquer competidor pode reclamar um empate caso não haja nenhuma captura de peça e não houve movimento de nenhum peão durante 50 movimentos.

4) OUTRAS ANOTAÇÕES PARA O JOGO DE XADREZ

4.1 Notação Algébrica Longa

Alguns programas de computador (e pessoas) usam uma variante da notação algébrica para xadrez, chamada *long algebraic notation* or *fully expanded algebraic notation*. Na notação algébrica completamente expandida, movimentos incluem a posição inicial e a posição final do movimento separado por hífen. Por exemplo "e2-e4", "Nb1-c3" ou "Rd3xd7". Essa notação toma mais espaço, e por isso não é comumente usada. Todavia, tem a vantagem da clareza, particularmente para os jogadores menos hábeis , ou para aprendizes do jogo.

4.2 Notação Numérica

No xadrez internacional por correspondência, o uso de notação algébrica pode causar confusão, pois linguas diferentes dão nomes diferentes (e portanto iniciais diferentes) para as peças. Daí, o padrão de transmissão adotado nesta forma de xadrez é a notação numérica (ICCF).

Na notação numérica, todas as casas são numeradas com um número de dois dígitos. Nesse sistema de coordenadas simples, o primeiro dígito descreve a coluna, e o segundo a fila. Um movimento é definido por um par de dois desses dois dígitos juntos. Por exemplo, o movimento que fosse escrito como *1. e4* em notação algébrica, seria escrito como *1. 5254* na numérica, significando que o peão que está na casa (5,2), move-se para a casa (5,4). Nem o tipo de peça, nem a captura são específicamente marcadas na notação numérica – todos os movimentos, exceto a promoção do peão, consiste de somente 4 dígitos.

Na promoção do peão, um quinto número é adicionado : 1 para a rainha, 2 para a torre, 3 para o bispo e 4 para o cavalo. Por exemplo, se um peão estiver na f7 e for movido para f8 com promoção para uma torre, a notação seria *67682*.

Para o roque, a posição de inicio e de fim do rei é registrado: para as brancas, *5131* (roque do lado da dama) e *5171* (roque do lado do rei); para as pretas, *5838* (roque do lado da rainha) e *5878* (roque do lado do rei).

ANEXO II –
HEURÍSTICA DO GNU CHESS

This file contains a description of GNU's heuristics.

-- requested by main author

Heuristic descriptions for CHESS.

Revision: 12-16-87

Modified 1991 by Mike McGann

Timing:

An initial quantum of time = Time Remaining/(2 * moves remaining +1) is alocated to the search. After this time is used up a check is made if the score has changed more than ZSCORE points between the last two search levels or if at least ZDEPTH has not been reached or if the choosen move has changed between the last two search levels the time is doubled.

Evaluation:

Here is a brief description of the heuristics used in the positional evaluator of the GNU Chess program. Many heuristics are functions of the stage of the game which is based on the total non-pawn material remaining for both sides.

PAWNS

The material value of a pawn is 100 points. Isolated pawns get a penalty depending on which file they occupy: (12,14,16,20,20,16,14,12) for files (a..h).

Doubled pawns (which are not also isolated) get a penalty of 12 points. Backward pawns (defined simply as not being defended by a pawn with the square in front also not defended by a a pawn) are penalized 6 points. A 4 point penalty is also invoked for each attack by the opponent to a backward pawn and for a backward pawn on a half-open file. Pawn Advancement in the centre is given a bonus of about 4 points per rank in the opening increasing to about 8 points per rank in the ending. Advancement on the edges is given a lower bonus. Pawns on the e and d files and on the 2nd rank are given a 10 Point penalty. An additional penalty of 15 points is invoked if these pawns are also blocked. Pawns within 2 squares of the king are given a 10 point bonus. Passed pawns are given a bonus for increasing rank which is a function of stage of the game and of whether the opponent blocks or attacks one or more squares in front of the pawn or if the opponents king is in the square of the pawn. This bonus ranges from about 15 points for a pawn on the second rank up to about 300 points for a passed pawn on the 7th rank which can't be stopped from queening.

KNIGHTS

The material value of a knight is 330 points. The main heuristic for knights is a bonus for proximity to the centre. This varies from 0 points in the corners to 30 points in the centre. Knights are also given a bonus for being within 2 squares of each enemy piece. This bonus is a function of the stage of the game, equalling 4 points in the end game. A penalty of 1 point per square is given for distance from either king. A bonus of up to 8 points (depends on stage) is given for knights which can't be driven away by enemy pawns.

BISHOPS

The material value of a bishop is 330 points. Bishops are given a bonus as material falls off the board equalling 10 points in the end game. Bishops get a bonus for mobility and Xray mobility thru pieces but not pawns. This bonus ranges from -4 points for a totally blocked bishop up to 18 points for a bishop attacking 12 or more squares.

Xray attacks on an enemy R,Q,K or any undefended piece are given an 8 point bonus. Bishops are given a bonus of 14 points if they lie on the edge of the board up to 22 points if

the lie in the centre. A bishop is given a bonus of up to 5 points for each attack to a square adjacent to the enemy king.

ROOKS

The material value of a rook is 520 points. Rook mobility is handled similiarly to bishops with a bonus of 0 points if blocked up to 20 points if attacking 12 squares or more. A bonus of 8 points for Xray attacks is handled as it is for bishops. Rooks are given a bonus of 10 points for occupying a file with no friendly pawns and a bonus of 4 points if no enemy pawns lie on that file. After the opening Rooks are penalized slightly depending on "taxicab" distance to the enemy king.

QUEENS

The material value of a queen is 980 points. The only heuristic for a queen is that after the opening it is penalized slightly for "taxicab" distance to the enemy king.

KINGS

Kings are given a penalty for proximity to the centre in the opening and a bonus for proximity to the centre in the endgame. The penalty is about 24 points for being in the centre in the opening with a bonus of about 36 points for being in the centre in the endgame. Except when the otherside has only pawns. Then the bonus is turned off. Kings are penalized for lying on an open or half-open file or if the adjacent file closest to the corner is open or half-open. This penalty is up to 23 points in the opening and goes to zero in the end game. The King is penalized up to 8 points if there are no pawns immediately adjacent. A penalty is invoked depending on the number of "safe" checks available by the opponent. This penalty ranges from 6 points for one such check to 50 points for 4 or more. Depending on game stage, Kings are given up to 10 points for castling and a penalty of up to 40 points for moving before castling.

SPECIAL

If more than one piece is "hung" (attacked and not defended or attacked by an enemy piece of lower value) an extra penalty of 10 points is invoked for that side and the search may be extended one ply. Pinned or trapped pieces are treated similarly. A special mating routine is used if one side has only a king and the other has mating material.