

Universidade Federal de
Santa Catarina

Programa de Pós-
Graduação em Ciência
da Computação

ppgcc.posgrad.ufsc.br

Campus Reitor João
David Ferreira Lima -
Trindade

Florianópolis- SC

Dissertação apresentada ao Programa de Pós-
Graduação em Ciência da Computação,
Departamento de Informática e Estatística, do
Centro Tecnológico da Universidade Federal de
Santa Catarina, como requisito para obtenção do
título de mestre em Ciência da Computação

Orientadora: Prof^{fa}. Jerusa Marchi, Dr^a.

Coorientador: Prof. Elder Rizzon Santos, Dr.

Florianópolis, 2017

Arquitetura de Ambiente Inteligente de Aprendizagem Utilizando Lógica Temporal
Darlan Anschau

Arquitetura de Ambiente Inteligente de Aprendizagem
Utilizando Lógica Temporal

Darlan Anschau

Esta dissertação apresenta uma arquitetura multiagente de paradigma Agentes & Artefatos chamada Smart ITS. Nela, o Agente Tutor se destaca por utilizar de operadores de Lógica Temporal para representar crenças e sequenciar Objetos de Aprendizagem, provendo adaptatividade e estrutura robusta no domínio de conhecimento em que efetua tutoria.

Orientadora: Prof^{fa}. Jerusa Marchi, Dr^a.

Coorientador: Prof. Elder Rizzon Santos, Dr.



Darlan Anschau

**ARQUITETURA DE AMBIENTE INTELIGENTE DE
APRENDIZAGEM UTILIZANDO LÓGICA TEMPORAL**

Dissertação submetida ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Santa Catarina para a obtenção do Grau de Mestre em Ciência da Computação.

Orientadora

Universidade Federal de Santa Catarina:
Prof^ª. Jerusa Marchi, Dr^ª.

Coorientador

Universidade Federal de Santa Catarina:
Prof. Elder Rizzon Santos, Dr.

Florianópolis

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Anschau, Darlan
Arquitetura de Ambiente Inteligente de
Aprendizagem Utilizando Lógica Temporal / Darlan
Anschau ; orientadora, Jerusa Marchi, coorientador,
Elder Rizzon Santos, 2017.
97 p.

Dissertação (mestrado) - Universidade Federal de
Santa Catarina, Centro Tecnológico, Programa de Pós
Graduação em Ciência da Computação, Florianópolis,
2017.

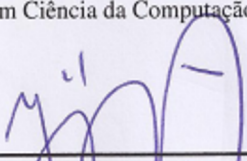
Inclui referências.

1. Ciência da Computação. 2. Lógica Temporal. 3.
Sistemas Tutores Inteligentes. 4. BDI. 5.
Planejamento Automático. I. Marchi, Jerusa. II.
Santos, Elder Rizzon. III. Universidade Federal de
Santa Catarina. Programa de Pós-Graduação em Ciência
da Computação. IV. Título.

Darlan Anschau

**ARQUITETURA DE AMBIENTE INTELIGENTE DE
APRENDIZAGEM UTILIZANDO LÓGICA TEMPORAL**

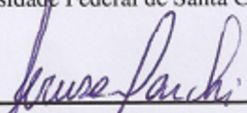
Esta Dissertação foi julgada aprovada para a obtenção do Título de Mestre em Ciência da Computação, e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação.



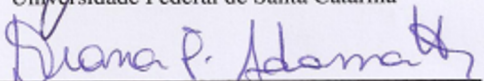
JOSÉ LUÍS ALMADA GÜNTZEL
Subcoordenador do PPGCC/UFSC
Portaria Nº 1882/2015/GR

Prof. Elder Rizzon Santos, Dr.
Coorientador
Universidade Federal de Santa Catarina

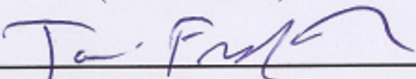
Banca Examinadora:



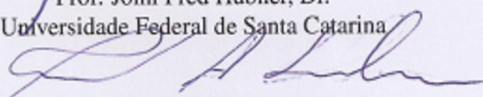
Prof.ª Jerusa Marchi, Dr.ª
Orientadora
Universidade Federal de Santa Catarina



Prof.ª Diana Francisca Adamatti, Dr.ª
Universidade Federal do Rio Grande
(Videoconferência)



Prof. Jomi Fred Hübner, Dr.
Universidade Federal de Santa Catarina



Prof. Ricardo Azambuja Silveira, Dr.
Universidade Federal de Santa Catarina

Dedico este trabalho ao meu pai, Dari, à minha mãe, Maria, e ao meu irmão, Maicon.

AGRADECIMENTOS

Agradeço em primeiro lugar a Deus. À minha orientadora, Jerusa Marchi, que considero como uma segunda mãe e amiga, por ter sido imprescindível para o meu amadurecimento acadêmico e por considerar o lado humano da pesquisa. Ao meu coorientador, Elder Rizzon Santos, por suas inestimáveis contribuições. Aos colegas do laboratório IATE, em especial ao professor Ricardo Azambuja Silveira e ao Thiago Ângelo Gelaim.

“A verdadeira viagem de descobrimento não consiste em procurar novas paisagens, e sim em ter novos olhos”.

(Marcel Proust)

RESUMO

Nos últimos tempos, o sequenciamento de atividades educacionais ocorre por meio de recursos na forma de Objetos de Aprendizagem (OAs), os quais podem ser selecionados para personalizar atividades conforme características específicas do Aprendiz. Apresenta-se aqui a hipótese de que características dinâmicas do Aprendiz podem ser representadas em função de operadores de Lógica Temporal, e mais especificamente, pode-se estabelecer uma relação deste com o Domínio de Conhecimento em um curso. Portanto, estes aspectos temporais somam-se a outras características na identificação do estado corrente do Aprendiz, servem de amparo para a definição do escopo de curso e auxiliam no planejamento de ações futuras. A partir disto, propõe-se um sistema híbrido entre Sistemas Tutores Inteligentes (STI) e Sistemas Hipermídia Adaptativos Educacionais (SHAE), resultando em um sistema, denominado Smart ITS, constituído de uma arquitetura que contém agentes e artefatos. O Agente Tutor do Smart ITS destaca-se por fazer uso de operadores de Lógica Temporal para representar crenças e garantir tanto a flexibilidade, provida com a adaptatividade de OAs, quanto a estrutura robusta do Domínio de Conhecimento em que efetua tutoria. As crenças e as respectivas semânticas temporais que são utilizadas pelo Agente Tutor servem de modelo para planejamento, visto que as mesmas ditam o sequenciamento ordenado de ações através de proposições que representam estados de mundo e que seguem um modelo conceitual com a finalidade de elaboração de planos.

Palavras-chave: Lógica Temporal. Sistemas Tutores Inteligentes. BDI. Planejamento Automático.

ABSTRACT

In the last years, the sequencing of educational activities occurs through resources in the form of Learning Objects (LOs), which can be selected to personalize activities in conformity with the learner specific characteristics. We present as hypothesis that dynamic features can be represented in accordance with Temporal Logic and, more specifically, we can establish a relation from the learner to the course knowledge domain. Therefore, these temporal aspects are added to other characteristics for the current Learner's state identification, supporting the course's scope definition and aiding for planning future actions. Thus, we propose a hybrid Intelligent Tutoring Systems (ITS) and Adaptive Educational Hypermedia Systems (AEHS), resulting in a system, named Smart ITS, consisting of an architecture that contains agents and artifacts. The Tutor Agent in Smart ITS stands out by utilizing Temporal Logic operators to represent beliefs and by granting both flexibility, provided by the adaptability with LOs, and the robust structure of domain knowledge wherein tutoring takes effect. The temporal beliefs and the respective temporal semantics used by the Tutor Agent serve as model for planning, since those same dictate the ordered sequencing of actions through propositions that represent states of the world and follow a conceptual model with the finality of plans elaboration.

Keywords: Temporal Logic. Intelligent Tutoring Systems. BDI. Automated Planning.

LISTA DE FIGURAS

Figura 1	Evolução dos Sistemas de Ensino Utilizando o Computador. .	38
Figura 2	Arquitetura Clássica de um STI.....	40
Figura 3	Arquitetura MATHEMA.....	45
Figura 4	Arquitetura de um Sistema Tutor.....	46
Figura 5	Visão Geral da Arquitetura Smart ITS.....	65
Figura 6	Conjuntos de Domínio de Aprendiz e Conhecimento.....	70
Figura 7	Modelo Conceitual MATHEMA Usando OAs.....	72
Figura 8	Algoritmo de Profundidade para o Sequenciamento Simples .	76

LISTA DE PSEUDO-CÓDIGOS

Pseudo-código 1	Ciclo de um Agente METATEM	52
Pseudo-código 2	Ciclo de BDI no Jason	56

LISTA DE QUADROS

Quadro 1	Trabalhos Derivados do Sistema MATHEMA.....	48
Quadro 2	Operadores da Lógica Temporal PML.....	53
Quadro 3	Comparativo de Lógicas com Ênfase na Temporalidade.....	58
Quadro 4	Operadores Temporais dos Tempos Verbais Passado e Futuro	69
Quadro 5	Modelo Conceitual do Curso “Métodos Algorítmicos”.....	71

LISTA DE ABREVIATURAS E SIGLAS

IA	Inteligência Artificial	27
AVA	Ambientes Virtuais de Aprendizagem	27
EaD	Ensino a Distância	27
IA-ED	Inteligência Artificial na Educação	28
CAI	<i>Computed Assisted Instruction</i>	28
STI	Sistema Tutor Inteligente	28
AIA	Ambientes Interativos de Aprendizagem	28
CSCL	<i>Computer Supported Collaborative Learning</i>	28
SHAE	Sistema Hiperfídia Adaptativo Educacional	28
SMA	Sistema Multiagente	28
RDP	Resolução Distribuída de Problemas	29
POA	Programação Orientada a Agentes	29
A&A	Agentes & Artefatos	29
BDI	<i>Belief-Desire-Intention</i>	29
RdP	Redes de Petri	30
OA	Objetos de Aprendizagem	31
PRS	<i>Procedural Reasoning System</i>	36
dMARS	<i>distributed MultiAgent Reasoning System</i>	36
ICAI	<i>Intelligent CAI</i>	39
ITS	<i>Intelligent Tutoring System</i>	39
SOPHIE	<i>SOPHisticated Instructional Environment</i>	39
PLATO	<i>Programmed Logic for Automatic Teaching Operations</i>	39
AHAM	Adaptive Hypermedia Application Model	42
LMS	<i>Learning Management System</i>	43
LD	<i>Learning Design</i>	43
SS	<i>Simple Sequence</i>	43
SCORM	<i>Sharable Content Object Reference Model</i>	43
ADL	<i>Advanced Distributed Learning</i>	43
SN	<i>Sequencing & Navigation</i>	43
LOM	<i>Learning Objects Metadata</i>	43
SCO	<i>Sharable Content Object</i>	43
LRS	<i>Learning Record Store</i>	43

UP	Unidades Pedagógicas	47
UI	Unidades de Interação	47
JESS	<i>Java Expert System Shell</i>	49
RPO	Redes Petri Objeto	49
LTL	<i>Linear Temporal Logic</i>	49
UB	<i>Unified system of Branching Time</i>	49
CTL	<i>Computer Tree Logic</i>	49
PTL	<i>Propositional Temporal Logic</i>	50
SNF	<i>Separated Normal Form</i>	50
PML	<i>Propositional MetateM Logic</i>	52
fbf	fórmula-bem-formada	52
<i>LORA</i>	<i>Logic Of Rational Agents</i>	56
KARO	Knowledge and belief, Abilities, Results and Opportunities ..	57
SMC	Sistema Multicontexto	62

LISTA DE SÍMBOLOS

○	<i>Next</i>	53
◇	<i>Sometime</i>	53
□	<i>Always</i>	53
●	<i>Weak last</i>	53
●	<i>Strong last</i>	53
◇	<i>Was</i>	53
■	<i>Herefore</i>	53
U	<i>Until</i>	53
W	<i>Unless</i>	53
S	<i>Since</i>	53
Z	<i>Zince</i>	53
A	para todos os futuros	55
E	para algum futuro	55

SUMÁRIO

1	INTRODUÇÃO	27
1.1	INTELIGÊNCIA ARTIFICIAL NA EDUCAÇÃO	28
1.2	INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA	29
1.3	MOTIVAÇÃO	30
1.4	CONTRIBUIÇÕES	31
1.4.1	Objetivo Geral	31
1.4.2	Objetivos Específicos	31
1.5	ORGANIZAÇÃO DO TRABALHO	32
2	FUNDAMENTAÇÃO TEÓRICA	33
2.1	SISTEMAS MULTIAGENTES (SMA)	33
2.1.1	Agentes com Estados Mentais	35
2.1.2	Modelo BDI	36
2.2	SISTEMAS TUTORES INTELIGENTES (STI)	37
2.2.1	Histórico dos STIs	38
2.2.2	Arquitetura Clássica De Um STI	39
2.2.3	Correntes Psicológicas	40
2.3	SISTEMAS HIPERMÍDIA ADAPTATIVOS EDUCACIONAIS (SHAE)	41
2.3.1	Padrões para o Sequenciamento de OAs	43
2.4	MATHEMA	44
2.4.1	Modelo Conceitual MATHEMA	45
2.4.1.1	Projeto MATHNET	47
2.5	LÓGICA TEMPORAL	49
2.5.1	Linear Temporal Logic (LTL)	50
2.5.1.1	Sintaxe de uma Lógica Temporal Proposicional	50
2.5.1.2	Agentes Deliberativos METATEM	52
2.5.1.3	Semântica de Lógica Temporal	54
2.5.2	Computer Tree Logic (CTL)	55
3	TRABALHOS RELACIONADOS	59
3.1	TRABALHOS RELACIONADOS COM OBJETOS DE APRENDIZAGEM	59
3.2	TRABALHOS DA ÁREA DE SISTEMAS TUTORES INTELIGENTES	61
3.3	LÓGICA MODAL	61
3.3.1	Trabalhos em Lógica BDI	62
4	SMART ITS	63

4.1	ARQUITETURA DO SISTEMA MULTIAGENTE DE TUTORIA	66
4.2	PROPOSTA DE AGENTE TUTOR	68
4.3	ESTRUTURA DO DOMÍNIO DE CONHECIMENTO	70
4.4	ESTRUTURA DE PLANO	72
4.4.1	Crenças do Agente Tutor Sobre o Domínio de Conhecimento	73
4.4.2	Semântica de Passado	77
4.4.3	Semântica de Ação	77
4.4.4	Semântica de Plano	78
4.5	AGENTE TUTOR: SEQUENCIANDO OBJETOS DE APRENDIZAGEM UTILIZANDO OPERADORES TEMPORAIS ...	79
4.5.1	Exemplo	80
4.5.1.1	Um Curso Simples	80
5	CONCLUSÃO	85
5.1	TRABALHOS FUTUROS	86
	REFERÊNCIAS	87

1 INTRODUÇÃO

“Em tudo o que fazemos, temos em vista alguma outra coisa.”

(Aristóteles)

Atualmente, as tecnologias de informação providas por sistemas computacionais são difundidas ao grande público e cada vez mais crescem em popularidade. Em boa parte, estes fenômenos acontecem devido ao barateamento e crescente inovação de *hardware* que garantem baixo custo e possibilita grande poder de processamento e espaço para armazenamento em memória. Por parte do *software*, diversas aplicações computacionais tornam possível aprimorar e criar novas funcionalidades, além de lidar com paradigmas antes intransponíveis.

Por conta disso, diversas áreas de conhecimento usufruíram da computação para o crescimento de suas pesquisas. Em particular, a área da educação recebeu contribuições significativas da Inteligência Artificial (IA), por meio dos Ambientes Virtuais de Aprendizagem (AVA) e da Inteligência Artificial na Educação, em especial da Inteligência Artificial Distribuída. Para situar o tema abordado neste trabalho, esta seção contempla conceitos da área educacional que serviram de base para o ensino com suporte tecnológico (MOORE; DICKSON-DEANE; GALYEN, 2011):

- **Distance Learning ou Ensino a Distância (EaD):** é uma modalidade de ensino voltada a aprendizes que encontram-se geograficamente distantes. Pode-se nela, mas não necessariamente, fazer uso do computador ou mídias digitais. Por isso, o termo evoluiu para outras formas de aprendizagem como *e-learning* e Educação *On-line*;
- **e-learning ou aprendizagem eletrônica:** é uma família de tecnologias educacionais que faz uso de tecnologias de informação. Caso contenha aulas presenciais e a distância é denominado de *blended learning*. O *e-learning* caracteriza-se bastante pelo uso de meios de comunicação novos, fornecendo mídias novas como áudio, vídeo e conteúdo instrucional para aprendizagem através da *Web*;
- **Educação *On-line*:** a sua definição é a mais controversa dentre as já citadas, pois não está claramente definida na literatura. É uma modalidade da EaD que visa conectividade, comunicação e interatividade, e pode fazer uso de correio, rádio, televisão, entre outros.

A Seção 2.3 confere maior importância aos recursos educacionais dos ambientes de *e-learning*, mas primeiro trataremos da IA inserida na Educação, que evoluiu em conjunto com os ambientes virtuais para torná-los inteligentes.

1.1 INTELIGÊNCIA ARTIFICIAL NA EDUCAÇÃO

A área da Inteligência Artificial na Educação (IA-ED) emprega técnicas de IA para a construção de modelos úteis em programas computacionais educacionais, visando melhorias na aprendizagem (MONTEIRO, 2006; FRIGO et al., 2007). A IA-ED inclui vários paradigmas, Cardoso et al. (2004) expõem dentre estes: Sistemas de Instrução Assistida por Computador, *Intelligent Computer Aided Instruction* (ICAI); micromundos; Sistemas Tutores Inteligentes (STI); Ambientes Interativos de Aprendizagem (AIA); e *Computer Supported Collaborative Learning* (CSCL).

Para Grubišić, Stankov e Žitko (2015), no entanto, os melhores sistemas de *e-learning* adaptativos são os STIs e os Sistemas Hiperídia Adaptativo Educacionais (SHAEs).

A grande preocupação de um STI é a de que o conhecimento de um domínio (o quê ensinar) seja repassado (como ensinar) ao aprendiz (a quem ensinar) para que este ganhe conhecimentos. Neste sentido, o Modelo de Sobreposição, do inglês *Overlay*, assume que o Aprendiz possua um conhecimento insuficiente, porém correto pertencente a um determinado domínio (VICTORIO-MEZA; MEJÍA-LAVALLE; RODRÍGUEZ, 2014). Geralmente é necessário, para um sistema educativo, que o usuário Aprendiz adquira conhecimento, mas a abordagem usada em um STI difere-se daquela usada em um SHAE Grubišić, Stankov e Žitko (2015). Este, tipicamente, se preocupa com a apresentação de conteúdo e navegabilidade (de que maneira apresentar). Nesta vertente, surgiu do conceito de Objeto de Aprendizagem (OA): qualquer entidade utilizada e referenciada na aprendizagem com suporte tecnológico (ELECTRICAL; COMMITTEE, 2002). Qualquer recurso educacional pode se tornar um OA se corretamente anotado, com padronização, qualidade e garantia de reuso.

A Seção 2.2 é dedicada aos STIs. No que tange a IA-ED, os STIs vêm se destacando por usufruir de benefícios advindos da divisão de tarefas. Especificamente, Sistemas Multiagentes (SMA) possibilitam maior abstração para o tratamento de problemas complexos e distribuídos (POZZEBON et al., 2008). Com a finalidade de definir SMAs, primeiramente são consideradas suas origens e heranças conferidas dentro da área da Inteligência Artificial Distribuída (IAD).

1.2 INTELIGÊNCIA ARTIFICIAL DISTRIBUÍDA

Avanços tecnológicos tornaram os computadores máquinas capazes de trabalhar concorrentemente e de se conectar a redes. A partir da IAD, também foi possível tirar a exclusividade de uma única aplicação ou de um único agente de IA para, em troca, resolver problemas através de múltiplas ações concorrentes, distribuídas de maneira coordenada, de forma parecida com o que pessoas realizam em atividades de grupo (BOND; GASSER, 1988). A IAD se subdivide em três ramos distintos, conforme Ishida (1995):

- **Resolução Distribuída de Problemas (RDP):** interessa-se pela decomposição de problemas entre um número de módulos que cooperam entre si e dividem conhecimento. São projetados especificamente para obtenção da solução de um problema particular;
- **Sistemas MultiAgentes (SMA):** caracteriza-se pela existência de um certo número de agentes autônomos, heterogêneos e potencialmente independentes, trabalhando juntos para resolver problemas que eventualmente surgem. Estes agentes são capazes de se adaptar ao meio ambiente em que estão contidos, reagindo e provocando mudanças nele;
- **IA Paralela:** interessa-se mais pelo bom desempenho do que por avanços conceituais, preocupando-se principalmente em desenvolver linguagens e algoritmos de computação paralela.

O desenvolvedor que utiliza de uma arquitetura baseada em agentes tem maior liberdade para programar por não precisar se ater em demasiado com problemas das áreas de redes ou computação paralela. Além disto, o desenvolvedor recebe amparo de conceitos, tais como os contidos em paradigmas de Programação Orientada a Agentes (POA), de Shoham (1993), Agentes & Artefatos (A&A), de Ricci, Viroli e Omicini (2007), comunicação por via de Atos de Fala (SEARLE, 1969), dentre outros. A Seção 2.1 trata mais de agentes, enfatizando os agentes com lógicas modais que representam computacionalmente o modelo mentalista de crenças, desejos e intenções, do inglês *Belief-Desire-Intention* (BDI).

No paradigma de A&A, agentes são entidades ativas, que podem ter a capacidade de deliberação e de comunicação com outros agentes da sociedade artificial. Enquanto isto, os artefatos são entidades com funções simples e servem como ferramentas, ou utensílios, para uso pelos agentes. A integração destas ferramentas à arquitetura de STI aqui desenvolvida possibilita a representação temporal proposicional do sequenciamento de OAs adaptado ao aprendiz.

1.3 MOTIVAÇÃO

O sistema MATHEMA está posicionado, dada a definição das áreas gerais apresentadas na Introdução, na intersecção entre *e-learning*, STI e SMA. O STI MATHEMA possui um modelo conceitual que estratifica o Domínio de Conhecimento do sistema de tutoria em três dimensões. Esta representação estruturada, chamada de modelo conceitual MATHEMA, é utilizada pelo Agente Tutor, o especialista do Domínio de Conhecimento do curso, para suprir conteúdo adequado ao Aprendiz.

Em sua grande maioria, as especificações de sistemas que agregam o modelo conceitual MATHEMA foram realizadas na forma de Redes de Petri (RdP). Bastante conhecidas por sua representação gráfica, as Redes de Petri destes trabalhos descrevem o comportamento distribuído, mas não representam a temporalidade internamente no agente, ou seja, não representam explicitamente aspectos temporais na forma de crenças. Assim sendo, os trabalhos envolvendo Redes de Petri apenas utilizam uma notação de lugares/transições para descrever redes ou sistemas.

Por outro lado, uma maneira de formalizar agentes em SMA utiliza de lógica modal para a representação da dinamicidade do ambiente multiagente. Uma lógica modal considerando aspectos de temporalidade é denominada de Lógica Temporal. Uma forma de representar a temporalidade via Lógica Temporal utiliza operadores modais para descrever os tempos verbais de passado e presente/futuro. Neste tipo de Lógica Temporal unem-se os paradigmas declarativo e imperativo, fornecendo uma percepção dinâmica de ambiente, de acordo com a mudança nas proposições do mundo percebido pelo agente na forma de crenças, e também sendo utilizada para ações no mundo, desencadeadas por regras que consideram estes aspectos percebidos de temporalidade.

Com o surgimento de novas mídias nos sistemas educacionais, o sequenciamento de recursos educacionais se tornou, em suma, uma adaptação do currículo por sequência de Objetos de Aprendizagem (OAs) disponíveis. A criação, manutenção e atualização das informações de domínio e de adaptação ao Aprendiz em OAs é feita por descrição em metadados elaborados por especialistas humanos.

Assim, a escolha e seleção de OAs de acordo com a identificação das preferências do Aprendiz pode ser aprimorada com os aspectos temporais para o diagnóstico da situação do Aprendiz no curso, e com isto, seu histórico passa a ser mais uma característica relevante para decidir as próximas atividades do Sistema Tutor. Para isto, representar o Domínio de Conhecimento de forma consistente auxilia neste diagnóstico e na seleção dos OAs para as atividades de interação pedagógicas futuras, em um curso consistente.

1.4 CONTRIBUIÇÕES

A proposta deste trabalho apresenta uma arquitetura de STI, o Smart ITS, e um Agente Tutor que propiciam a elaboração de planos ao usar de Lógica Temporal (LT) para representar atividades suportadas com Objetos de Aprendizagem (OA) por meio de proposições. Como representação interna do agente, a estrutura de crenças e a semântica de operadores servem como base para o sequenciamento. Este trabalho é uma solução para o problema de adaptatividade do conteúdo, adicionando aspectos temporais à identificação das necessidades do Aprendiz em relação ao curso atual.

1.4.1 Objetivo Geral

Desenvolver uma arquitetura contendo agentes e artefatos, chamada Smart ITS, com a finalidade de representar o conhecimento do Aprendiz em Lógica Temporal no modelo conceitual MATHEMA por meio de um novo Agente Tutor BDI que promove o sequenciamento de Objetos de Aprendizagem na produção de um curso personalizado e adaptado.

1.4.2 Objetivos Específicos

- Analisar trabalhos relacionados por meio de revisão bibliográfica;
- Propor os elementos de uma arquitetura de STI multiagente, propiciando a identificação de aspectos temporais do Aprendiz em relação ao Domínio de Conhecimento;
- Especificar a representação do Modelo de Sobreposição entre Domínio de Conhecimento e Domínio de Aprendiz, de forma que contemple a identificação de aspectos temporais;
- Apresentar um Agente Tutor BDI com a capacidade de planejar o sequenciamento das ações a partir de crenças temporais, descrevendo os operadores utilizados para a proposta e a semântica empregada;
- Implementar a proposta e validar a arquitetura.

1.5 ORGANIZAÇÃO DO TRABALHO

Este capítulo, Capítulo 1, conteve uma introdução geral das áreas em que este trabalho se encontra, dispôs sobre a motivação, problema de pesquisa e contribuição, bem como de seus objetivos geral e específicos.

O Capítulo 2 apresenta a Fundamentação Teórica, endereçando as áreas e subáreas abrangentes à esta pesquisa, bem como conceitos e tecnologias, que embasam o trabalho elaborado.

No Capítulo 3 é realizada uma comparação de trabalhos recentes nas áreas abordadas no Capítulo 2, evidenciando diferenças para com outros desenvolvimentos recentes e ressaltar os pontos importantes desta dissertação.

O Capítulo 4 dedica-se à proposta desta dissertação, apresentando a arquitetura seguindo o paradigma de Agentes & Artefatos, o Smart ITS, as capacidades do novo Agente Tutor BDI, suas crenças e a semântica temporal da estrutura hierárquica do MATHEMA. Neste Capítulo também é descrito o ciclo de raciocínio do Agente Tutor e o desenvolvimento da proposta, descrevendo na qual um exemplo da implementação é apresentado.

O Capítulo 5 expõe a conclusão do trabalho, com uma breve discussão do tema tratado e recomendação de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

“*Nunca chegamos aos pensamentos.
São eles que vêm.*”

(Martin Heidegger)

Neste capítulo serão apresentadas as áreas de interesse deste trabalho e os conceitos que embasaram o desenvolvimento desta pesquisa.

Sistemas Tutores Inteligentes (STIs) recentes têm sido desenvolvidos baseados na ideia de sociedades de agentes, de forma a prover tutoria individualizada para o Aprendiz. Nestes sistemas as tarefas são decompostas em agentes de *software* que se comunicam para resolução de problemas. Portanto, a Seção 2.1 aborda a área de Sistemas Multiagentes (SMA) para fornecer características e algumas definições de tipos de agente. Na mesma seção, os agentes deliberativos de estados mentais recebem destaque, e em especial, o Modelo BDI. Na Seção 2.2, os STIs são introduzidos considerando a visão histórica das inspirações educacional, psicológica e computacional.

Com a evolução dos recursos educacionais para aprendizagem surgiram os Sistemas Hipermedia Adaptativos Educacionais (SHAES), abordados na Seção 2.3. Ainda apresenta-se como os SHAES influenciaram no desenvolvimento de Objetos de Aprendizagem (OAs), como esta área lida com adaptação e sobre os metadados usados para o sequenciamento de OAs.

A Seção 2.4, apresenta o sistema MATHEMA, que faz uso de SMAs para criação de STIs utilizando um modelo conceitual aproveitado em diversos trabalhos acadêmicos. Logo após, é realizada uma descrição de Lógicas Temporais na Seção 2.5 e das melhorias alcançadas ao se adicionar o parâmetro temporal a agentes.

2.1 SISTEMAS MULTIAGENTES (SMA)

A palavra “agente” provém do latim *agens*, e significa aquele que opera, produz, atua. O modelo de “ator”, de Hewitt e Jr. (1977), contempla boa parte desta definição a partir de pesquisas em Inteligência Artificial Distribuída. O ator seria tal como um agente, com endereço e comportamento, comunicando-se por vias de troca de mensagens com outros atores. Logo, agentes de *software* agregaram vantagens da computação distribuída. Correia (2011) aponta algumas características herdadas da IAD: adaptabilidade, cooperação, especialização contextual, custo, desenvolvimento paralelo, eficiência, estendibilidade, confiabilidade e qualidade dos resultados.

A Inteligência Artificial está diretamente vinculada aos sistemas compostos por agentes artificiais inteligentes, estes podendo apenas e simplesmente serem denominados de agentes. O conceito de agente não possui um fator consensual para que haja uma propriedade única que a distinga e defina. Portanto, a definição de um agente é abstraída a soma de determinadas características para qualificação ou desqualificação. Um conceito simples de agente segundo Genesereth e Nilsson (1988) é o de uma entidade, *hardware* ou *software*, com finalidade de intervir no mundo (ou ambiente, o local onde encontra-se inserido) adequadamente, e para isto, torna-se necessária a existência de sensores (para percepção do mundo) e atuadores (para realização de ações).

Para Russell e Norvig (2003), a qualidade intrínseca de um agente para uma performance ideal, na qual ele age de forma a realizar a melhor ação possível com as informações que detém, pode ser denominada de racionalidade. Para isto, são necessários os campos da matemática e engenharia. Seguindo nessa direção, os autores diferenciam a interação do agente com o tipo de ambiente, em que encontra-se, nos seguintes aspectos: observável, total ou parcialmente; de agente único ou multiagente, no último caso competindo ou cooperando; determinístico ou estocástico, em relação aos próximos estados do ambiente; episódico ou sequencial, em relação à ação do agente e suas influências; estático ou dinâmico, em relação às mudanças do ambiente enquanto o agente delibera; discreto ou contínuo, da maneira em que o agente lida com o tempo; e conhecido ou desconhecido, em relação às crenças do agente.

No contexto dos SMAs, existem diversas abordagens para a classificação de agentes distribuídos. Nwana (1996) menciona as diferenciações por agentes estáticos ou móveis, agentes reativos ou deliberativos, e por fim, pelas características que aspira-se que apresentem, julgando fundamentais: cooperação, aprendizagem e autonomia. Destas derivam os tipos de agentes: agentes colaborativos, agentes de interface, agentes móveis, agentes de informação/internet, agentes reativos, agentes híbridos e agentes heterogêneos.

Wooldridge e Jennings (1995) distinguem os agentes de SMA em agentes de noção fraca e agentes de noção mais forte. Na noção fraca, os agentes têm as seguintes propriedades:

- **autonomia:** operam sem intervenção direta humana, ou de outros, possuindo determinado controle sobre suas ações e estado interno;
- **habilidade social:** interagem com outros agentes (e possivelmente humanos) por uma determinada linguagem de comunicação;
- **reatividade:** percebem o ambiente e respondem em tempo adequado quando mudanças ocorrem;

- **proatividade:** têm iniciativa e não dependem exclusivamente de resposta ao ambiente, por meio de objetivos.

Já na noção mais forte, os agentes possuem as propriedades supracitadas mais propriedades inerentes aos seres humanos, desde simples caracterizações de aspectos humanos até estados mentais e, num futuro possível para alguns pesquisadores, emocionais (WOOLDRIDGE; JENNINGS, 1995).

Para Padgham e Winikoff (2005), as características que diferenciam agentes de objetos são: autonomia, proatividade e robustez. Sendo esta última, a robustez, uma característica de flexibilidade e recuperação, fazendo com que o agente busque executar o plano atual e, quando for conveniente, mudar para um plano melhor no intuito de alcançar os seus objetivos.

2.1.1 Agentes com Estados Mentais

Como já citado, embora brevemente a partir de Nwana (1996), agentes também podem ser classificados em dois tipos de agentes conforme o raciocínio da sua arquitetura:

- **agentes reativos:** em um agente deste tipo não existe uma representação explícita, ou seja, não há armazenamento em memória. O comportamento se deve a ações reflexivas de estímulo/resposta. Temos como exemplo a arquitetura proposta por Brooks (1986), na qual o fluxo decorre diretamente dos sensores para os módulos de competência e atuadores;
- **agentes cognitivos ou deliberativos:** a representação em crenças que transcorre de forma explícita por meio de representação interna, ou seja, a memória retrata a visão do agente sobre o ambiente onde está localizado. Guiam-se através do processo de deliberação: o processo de gerar e escolher opções/planos para satisfazer determinados objetivos.

Dos agentes de arquiteturas cognitivas ainda é possível fazer uma classificação entre agentes funcionais, que operam otimizando a resolução de um problema específico; e agentes baseados em estados mentais, carregando consigo influências da filosofia para a reprodução de um sistema intencional humano, como no trabalho pioneiro de (DENNETT, 1971), onde a intencionalidade de um sistema abrange crenças e desejos.

Uma classe intermediária entre reatividade e deliberação é a classe dos agentes dedutivos, que é melhor desenvolvida na Seção 2.5.1.2.

2.1.2 Modelo BDI

Dentre os agentes mentalistas, uma categoria de agentes representa crenças, desejos e intenções, do inglês *Belief-Desire-Intention* (BDI), tendo como base os pressupostos da filosofia de Bratman (1987). Um agente BDI planeja a partir do raciocínio prático e futuro-dirigido em estados mentais provindos de inspiração psicológica, componentes filosóficos, de arquitetura de *software* e lógica. Os primeiros sistemas com esta abordagem foram *Procedural Reasoning System* (PRS), de Ingrand, Georgeff e Rao (1992), e *distributed MultiAgent Reasoning System* (dMARS), de d'Inverno et al. (1998).

Um agente proativo tem como característica o uso de objetivos, enquanto que o reativo muda seu comportamento de acordo com sua resposta às mudanças do ambiente. As mudanças significativas do ambiente são os eventos, e para isso o agente deve responder de alguma forma (PADGHAM; WINIKOFF, 2005). Por isso, na área de agentes artificiais é compartilhado o conceito de atitudes: comportamentos humanos que visam propósitos, como acreditar, querer, ter esperança, medo, entre outros. *Information attitudes* são as informações do agente sobre o mundo e as *pro-attitudes* orientam as ações do agente (NWANA, 1996).

O modelo BDI pode ser descrito por diversas arquiteturas, nas quais os conjuntos computacionais são representados de formas diferentes. Por exemplo, o sistema intencional de Cohen e Levesque (1990) formaliza intenções, que abrangem desejos, e objetivos, que abrangem planos. Outra forma de representar estas definições em uma arquitetura BDI:

- **crenças (*beliefs*):** são o que na IA clássica é chamado de conhecimento. A partir da constatação de que não existe uma verdade definitiva e que o agente cria uma concepção da realidade, a crença se torna a visão do agente sobre o ambiente;
- **desejos (*desires*):** são as crenças de estados futuros, mesmo que sejam estados irreais. Para limitar o que é intencionado e factível, restringe-se este conjunto aos **objetivos (*goals*)**, que são os desejos concebíveis, reais e não conflitantes;
- **intenções (*intentions*):** são o conjunto de objetivos para os quais o agente possui recursos para realização. Uma intenção é uma escolha seguida de comprometimento. Da composição de intenções surgem os **planos (*plans*)**, que são as combinações das intenções do agente em unidades consistentes, para tanto, uma sequência de ações.

Algumas funções computacionais utilizadas para tratar estes conjuntos no modelo BDI, são segundo Wooldridge (2000):

- **função de revisão de crenças:** com base nas crenças e percepções, tem como produto as crenças atualizadas;
- **função de geração de opções:** com base nos conjuntos de crenças e intenções, tem desejos como produto;
- **função de filtro de intenções:** com base nos conjuntos de crenças, desejos e intenções, tem intenções como produto;

Rao (1991) ressalta alguns aspectos gerais dos agentes BDI para ajustes no desenvolvimento: processo deliberativo BDI, onde ocorre a geração de opções e filtragem; do raciocínio meio-e-fim, o qual concretiza os planos; das recomendações de intenções, ousado ou cauteloso; e das estratégias de comprometimento:

- **cega:** mantém a intenção até ser realizada;
- **obcecada:** mantém a intenção até a realização ou caso a conclusão não seja possível, e para;
- **liberal:** mantém a intenção enquanto o agente crê ser realizável.

Um exemplo de área que faz uso de SMAs em implementações é a área de Sistemas Tutores Inteligentes (STIs). Os STIs lidam com tarefas complexas que podem ser divididas entre agentes dentro de SMAs e por isso recebem destaque na próxima seção.

2.2 SISTEMAS TUTORES INTELIGENTES (STI)

Na década de 1920, o psicólogo Sidney Pressey desenvolveu diversas variantes daquilo que ele veio a chamar de “máquina de testes”. Antes dela, não existiam máquinas com o intuito de “ensinar”, existiam apenas dispositivos que repassavam material. Pressey visionava uma educação industrializada, porém a máquina dele não se tornou popular em sua época. Já nos anos 1950, Skinner (1958) criou sua própria máquina influenciado pela predecessora máquina de testes, e foi além disso, ao empregar seu novo modelo de ensino: o condicionamento operante (um processo vertente do behaviorismo) e a aprendizagem por reforço, obtendo breve popularidade.

Comparativamente, segundo Benjamin (1988), a máquina de Pressey dependia de tentativa/erro nas questões de múltipla escolha ou de conhecimento prévio do assunto, enquanto a máquina de Skinner não mostrava alternativas que estivessem erradas, mas prosseguia com pequenos passos, de

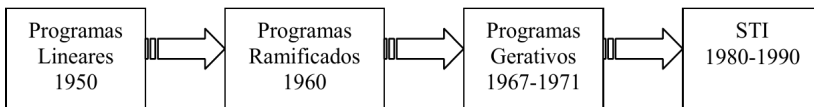
maneira construtiva sistemática e lógica. Ao fim dos anos 1960, com o descrédito de muitos em relação às máquinas de aprendizagem, o *Teaching Machine Project* do centro de pesquisas da IBM adaptou a abordagem de Skinner para desenvolvimento da arquitetura *Computed Assisted Instruction*(CAI) .

As máquinas não chegavam perto de ter comportamentos parecidos com a dinâmica apresentada por professores. Por isto, até este período de evolução tecnológica, ainda não haviam sido criados sistemas inteligentes para tutoria.

2.2.1 Histórico dos STIs

Para Nwana (1990), um Sistema Tutor Inteligente (STI) integra três áreas e suas respectivas subáreas: Psicologia (Cognitiva), Ciência da Computação (IA) e Educação e Treinamento (CAI). A Figura 1 apresenta um sucinto histórico, em ordem cronológica, das eras com programas de ensino.

Figura 1 – Evolução dos Sistemas de Ensino Utilizando o Computador.



Fonte: Gavidia (2003).

Os primeiros CAIs eram “programas lineares” baseados no princípio do condicionamento operante, ressaltando a orientação passo-a-passo e almejando um comportamento específico através de “frames”. Depois deles vieram os “programas ramificados”, nos anos 1960, que possibilitaram o sequenciamento dos *frames* para dar suporte, no fim dos anos 1960 e início dos anos 1970, à *generative CAI* (programas gerativos ou também chamados de sistemas adaptativos). Nwana (1990) afirma que os programas gerativos foram um grande avanço, pois estes sistemas foram os precursores dos principais STIs, porém ainda eram em sua maioria limitados a exercícios específicos de domínios bem estruturados.

O sistema SCHOLAR, de Carbonell (1970), foi uma das primeiras experiências em direção à área dos STIs, apesar de que ainda não tratava-se de um. De acordo com o objetivo educacional, o sistema visava oferecer material a partir do perfil do aprendiz, dialogando em linguagem natural, tanto perguntando quanto respondendo e informando sobre a exatidão da resposta do aprendiz. O SCHOLAR tinha limitadas capacidades de inferência, estratégias pedagógicas pouco desenvolvidas e necessitava melhorias no modelo

do aprendiz. Surgido de sistemas com arquitetura CAI, continuava a seguir a filosofia proposta por Skinner (1953).

Posteriormente, na década de 1980, surgiu uma nova geração, a dos STIs, a partir da adaptação *Intelligent CAI* (ICAI) e descrita por Sleeman e Brown (1982), criando o termo *Intelligent Tutoring System* (ITS). Por um longo período de tempo o ICAI foi considerado sinônimo de STI. Nessa arquitetura ocorre a simulação do conhecimento de um determinado domínio a ser ensinado para corresponder com estratégias de resolução de problemas. Porém, os ICAIs não contêm modelo do aprendiz e não seguem a arquitetura clássica de um STI.

Historicamente, outros STIs receberam notoriedade: SOPHIE (*SOPHisticated Instructional Environment*), de Brown e Burton (1974), situado em um ambiente de aprendizagem reativo, se diferencia do SCHOLAR por não ser sempre necessária uma fala para uma ação; GUIDON, Clancey (1983), faz uso do sistema especialista MYCIN de diagnóstico e seleção de tratamento para doença infecciosas; e o WEST, dos mesmos autores do SOPHIE, por diversas vezes nomeado de técnico por ser diferente de um tutor convencional, ao simular partidas de jogos de tabuleiro, em turnos, auxilia na tomada de decisões no sistema matemático PLATO (*Programmed Logic for Automatic Teaching Operations*).

Logo, um STI tenta imitar o processo de tutoria de um professor em relação a um aprendiz ou grupo de aprendizes a partir de uma arquitetura própria. Wenger (1987) define que um STI lida, em seu cerne, com as questões *do quê* e de *como* ensinar, em outras palavras, ele é dito inteligente por ser **especialista** em determinado domínio do processo de aprendizagem e por ser **adaptativo**, de acordo com a identificação das preferências do aprendiz.

2.2.2 Arquitetura Clássica De Um STI

Na arquitetura clássica básica de um STI, mostrada na Figura 2, se encontram os componentes funcionais (WENGER, 1987):

- **Modelo do Aprendiz:** responsável por responder *para quem ensinar?* Contém informações específicas de cada aprendiz de forma individual. O aprendiz é a razão da existência de um sistema tutor, logo, o STI deve criar um modelo fidedigno às características inerentes a ele, chamadas de preferências, assim como atualizá-las. É desejável, por exemplo, que destas preferências mantenha-se um histórico do progresso do Aprendiz, seu estilo cognitivo e perfil de aprendizagem.
- **Módulo Pedagógico** (também denominado de módulo tutor): demons-

tra aptidão na questão de *como ensinar?* Oferece uma metodologia para o processo de aprendizagem. Possui estratégias pedagógicas para selecioná-las em função das características do aprendiz;

- **Módulo do Domínio** (também denominado módulo do especialista): mantém conhecimento para responder a questão *do que ensinar?* Nele apresenta-se armazenada a representação sobre a área do Domínio de Conhecimento na qual o tutor está ensinando;
- **Interface**: intermedeia a interação entre o tutor e o aprendiz.

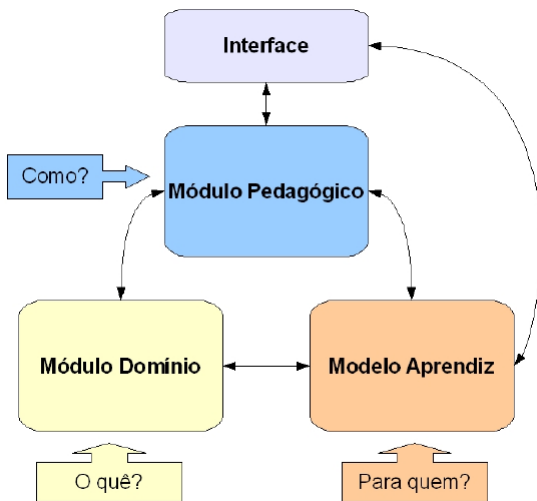


Figura 2 – Arquitetura Clássica de um STI
Fonte: Sibaldo (2010).

2.2.3 Correntes Psicológicas

Conforme citado na perspectiva histórica dos STIs, apresentada na seção anterior, a psicologia corroborou com o progresso e as suas correntes são usadas de diversas formas. Fleet (2012) destaca as três correntes da psicologia mais fortemente ligadas à criação de conteúdo educacional:

- **behaviorismo**: com raízes advindas dos famosos experimentos de Pavlov no condicionamento animal, prevaleceu nos anos 1960, sendo

Skinner seu maior representante na área da educação. Esta corrente tem como vantagens a clareza dos objetivos e por ser possível aprender por meio de respostas automáticas a estímulos. Em contraponto, dependendo do estímulo, o comportamento esperado pode não ocorrer e não há representação do conhecimento em nível abrangente;

- **teoria cognitiva:** a filosofia de Chomsky (1956) desafiou a perspectiva behaviorista, fazendo com que a informação possa ser vista como uma estrutura em esquemas processuais definidos por diferentes modelos: no registro sensorial, na memória de curto prazo e na memória de longo prazo. A meta a se alcançar são comportamentos consistentes: aqueles que são armazenados na memória de longo prazo. O maior problema, a partir desta finalidade, é a restrição imposta a uma única forma de realizar tarefas;
- **construtivismo:** seus principais representantes são Vygotsky e Piaget. O foco educacional é voltado para o indivíduo e na construção do conhecimento por experiência em diversas perspectivas de realidade. Nela, ao contrário das abordagens convencionais, o aprendiz é apresentado a realidades diferentes, onde é propício o desenvolvimento de habilidades para novas situações e desafios. Para tarefas comuns, a abordagem individual pode não ser a mais apropriada.

Após este pequeno compêndio sobre STIs, a próxima seção trata do que ocorreu nos últimos anos em prol da autoria para a criação ou anotação, no sentido de compartilhamento e reusabilidade de recursos educacionais em novas mídias. Como já visto, o advento de Objetos de Aprendizagem (OA) usados em aplicações *Web* abriu caminho para o surgimento da área de Sistemas HiperMídia Adaptativos Educacionais (SHAE).

2.3 SISTEMAS HIPERMÍDIA ADAPTATIVOS EDUCACIONAIS (SHAE)

Os SHAEs propõem uma solução diferente daquela apresentada por um STI ao usar de recursos de hiperMídia, atualmente predominantes para fins de aprendizagem. A ponto de Objetos de Aprendizagem estarem presentes em quase todas as grandes iniciativas de *e-learning* (DE-MARCOS et al., 2009). No caso dos SHAEs, o foco principal é a navegação e preferências do Aprendiz, fazendo dele um vetor para flexibilização do conteúdo.

Esta flexibilização em demasido, no entanto, pode causar ausência de compromisso com o conteúdo do domínio a ser apresentado. Para evitar esta falha, novos modelos surgiram para torná-los mais parecidos com os tradicionais STIs. Segundo (KARAMPIPERIS; SAMPSON, 2005), o estado-da-arte

em SHAEs baseia-se no modelo Adaptive Hypermedia Application Model (AHAM) (BRA; HOUBEN; WU, 1999; HENZE; NEJDL, 2004), que por sua vez, é bastante utilizado em aplicações de sistemas educacionais adaptativos usando hipermídia na *Web*.

A área de STI compartilha de dois grandes problemas com a área de Sistemas Hipermídia Adaptativos Educacionais (SHAE): a adaptação e a personalização da aprendizagem para com o Aprendiz.

No STI MATHEMA, que é apresentado na Seção 2.4, os papéis realizados por agentes humanos no sistema são: um motivador externo, a pessoa que informa ao Agente Aprendiz sobre o sistema STI; o Aprendiz, usuário e alvo do sistema; e os especialistas humanos, cujos papéis podem abranger desde a criação do *currículo*, a manutenção do sistema quando existe alguma falha ou dados insuficientes, até a provisão de atualizações. Na arquitetura clássica de STIs, apresentada na Seção 2.2.2, os produtos resultantes da atuação nestes papéis são respectivamente inseridos no Módulo do Domínio, Módulo Pedagógico e Modelo do Aprendiz. Sistemas de autoria, como o MathTutor, dependem de especialistas humanos, mas são pouco reutilizáveis por prenderem-se a ontologias que restringem-se ao domínio. No entanto, os SHAEs possuem papéis mais sofisticados para serem realizados por agentes humanos no sistema, de forma distribuída e utilizando de hipermídia.

De maneira parecida com os módulos da arquitetura clássica de STIs, o AHAM possui estes elementos em duas camadas principais: camada de execução, consistindo de um motor de inferência adaptativo; e a camada de armazenamento, consistindo de um Espaço de Mídia, um Modelo de Usuário e um Modelo Adaptativo. Porém, diferente de grande parte dos STIs, estes três módulos principais recebem contribuição dos seguintes papéis realizados por agentes humanos para projetar os recursos usados pelo sistema (KARAM-PIPERIS; SAMPSON, 2005):

- O Especialista do Domínio: é a pessoa que define e delimita o Domínio de Conhecimento no curso.
- O *Designer* de Conteúdo: é a pessoa responsável pelo desenvolvimento ou montagem de recursos educacionais. Neste processo, descreve os objetos e o uso dos mesmos com atributos na forma de metadados de OA, tais como de pré e pós-condições.
- O *Designer* Instrucional: é a pessoa responsável por relacionar o Domínio de Conhecimento com o Domínio de Aprendiz. Descreve as características individuais do modelo cognitivo e preferências do Aprendiz.

No caso desta dissertação, o Domínio de Conhecimento, a estrutura dos recursos educacionais (na forma de OAs) e o modelo de Aprendiz são

criados por estes respectivos papéis para serem armazenados em artefatos da arquitetura e fornecem dados ao Agente Tutor. A proposta, no Capítulo 4, mostra quais são os artefatos e o que eles precisam para a criação de currículos com a ajuda dos *designers* através de padrões existentes e com uma estrutura de modelo conceitual MATHEMA, usada para o sequenciamento de OAs provido por operadores de Lógica Temporal.

2.3.1 Padrões para o Sequenciamento de OAs

No contexto de SHAEs, os sistemas de gestão de aprendizagem, do inglês *Learning Management System* (LMS), utilizam padrões de especificações para gerenciar OAs:

- **IMS Learning Design (LD)**: é um LMS em si e possui pacotes de dados padronizados. Tem como intuito compreender todas as atividades e papéis que envolvam aprendizagem (IMS Global Learning Consortium, 2003). Para sequenciamento simples contém o IMS *Simple Sequence* (SS) com vasta documentação (IMS, 2003a; IMS, 2003b; IMS, 2003c). A implementação ocorre por meio de declarações XML no arquivo “imsmanifest.xml” do IMS *Content Packaging*.
- **Sharable Content Object Reference Model (SCORM)**: é um padrão especificado pela *Advanced Distributed Learning* (ADL) (ADL, 2009) que foca na interação individual entre o aprendiz e o conteúdo instrucional. Para o sequenciamento dos recursos, a sua árvore de atividades contém uma versão mais enxuta do IMS SS, o SCORM *Sequencing & Navigation* (SN). Essencialmente, este faz uso de Objetos de Aprendizagem no padrão de metadados IEEE *Learning Objects Metadata* (LOM), empacotado no denominado *Sharable Content Object* (SCO).
- **API Tin Can**: também conhecida como *Experience API* ou xAPI, é uma nova especificação da ADL para substituir o SCORM. Em ADL (2013), foi lançada a primeira versão completa e documentação, em conjunto com uma implementação em JSON. A especificação Tin Can provê melhorias em relação ao antigo padrão SCORM, nos quais os grandes diferenciais são:
 - **expressividade**: fazendo uso de declarações (*Statements*) na forma “Eu fiz isto” (RUSTICI, 2013);
 - **Learning Record Store (LRS)**: responsável pela gravação, coleta e monitoramento do conteúdo de aprendizagem;

- **compartilhamento:** o projeto Tin Can fornece interoperabilidade de padrões, além de permitir que o conteúdo seja disponibilizado de forma aberta.

Mesmo em um modelo como o AHAM, que lida melhor com conceitos pertencentes a um Domínio de Conhecimento, o *Designer* Instrucional se importa mais com os recursos para a anotação de metadados sobre os OAs, o que tende a isolar os recursos e conceitos.

A proposta desta dissertação cria cursos robustos, isto é, consistentes mas flexíveis, com a ajuda de OAs criados por *designers* de conteúdo, através de padrões existentes. O sequenciamento é facilitado com o uso de uma estrutura em lógica, onde o tutor pode raciocinar por meio de operadores de Lógica Temporal.

2.4 MATHEMA

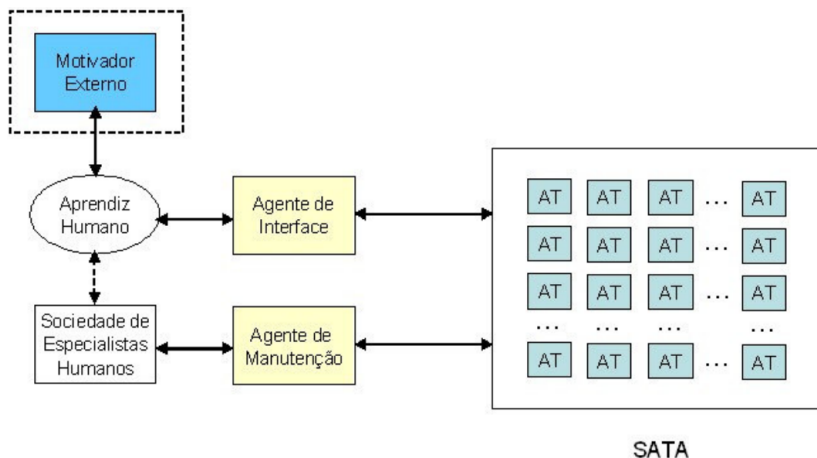
O sistema MATHEMA, tema da tese de doutorado de Evandro de Barros Costa (COSTA, 1997), foi inicialmente concebido como um protótipo de STI desenvolvido na linguagem de programação Common LISP. Nesta tese, havia sido apresentado um ambiente multiagente de tutoria tendo em vista viabilizar a comunicação interativa, de maneira cooperativa, entre agentes artificiais e humanos com fins de aprendizagem por meio de resolução de problemas. A Figura 3 mostra a arquitetura do STI MATHEMA, a *Arq_Mat*.

Definição 1 *O ambiente multiagente MATHEMA configura-se em uma 6-upla de entidades artificiais e humanas, a saber, a arquitetura $Arq_Mat = \langle AH, SATA, SEH, AI, AM, ME \rangle$, onde: AH é o Aprendiz Humano; SATA, a Sociedade de Agentes Tutores Artificiais; SEH, a Sociedade de Especialistas Humanos; AI, o Agente de Interface; AM, o Agente de Manutenção; e ME, um Motivador Externo.*

O Aprendiz Humano é motivado a utilizar o sistema MATHEMA por uma entidade humana, o Motivador Externo. Quando pronto para a interação, o Aprendiz Humano tem acesso ao sistema MATHEMA por meio do Agente de Interface, que serve de canal de comunicação entre o Aprendiz Humano e a Sociedade de Tutores Artificiais, na qual Agentes Tutores se organizam. Ou seja, é a partir do Agente de Interface que ocorrem as interações de aprendizagem entre Agente Aprendiz e sistema.

Por outro lado, o sistema é alimentado de conhecimento com especialistas do domínio do STI, através da Sociedade de Especialistas Humanos. O canal de comunicação entre um especialista humano e o sistema MATHEMA ocorre por meio do Agente de Manutenção.

Figura 3 – Arquitetura MATHEMA.



Fonte: Costa (1997).

Neste canal, ocorre a alimentação do sistema com atualização e correção de eventuais falhas não previstas. Ou ainda, o sistema, por meio da SATA, quando está ciente de não ter condições de resolver o problema por conta, requer assistência humana depois de identificar a ausência de informação necessária para auxiliar na resolução de determinados problemas.

Na arquitetura do STI MATHEMA, como apresentado na Figura 4, um agente é uma entidade composta por três sistemas: Sistema Tutor, Sistema Social e Sistema de Distribuição.

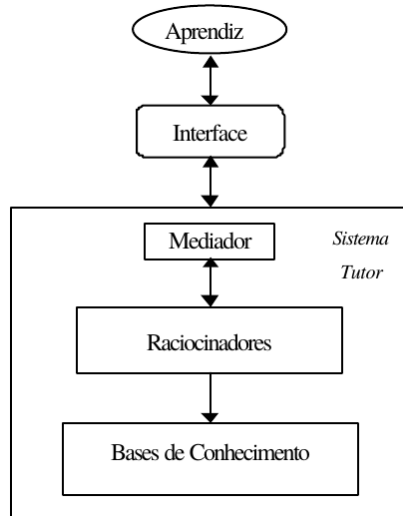
O modelo conceitual MATHEMA, parte das bases de conhecimento da proposta de STI em Costa, Lopes e Ferneda (1995) ganhou maior importância posteriormente com a sua reutilização por vários trabalhos na área de STIs. Este modelo conceitual trata da estrutura de dados concernente ao Domínio de Conhecimento de forma multidimensional.

2.4.1 Modelo Conceitual MATHEMA

A modelagem do conhecimento do domínio no MATHEMA é organizada em duas formas de visualização: a *visão externa* e a *visão interna*.

O modelo conceitual MATHEMA na sua *visão externa* consiste na separação de um domínio específico em subdomínios, representados em três dimensões de conhecimento: o **Contexto (C)**, caracteriza diferentes pontos

Figura 4 – Arquitetura de um Sistema Tutor



Fonte: Costa (1997).

de vista sobre o domínio e é responsável pela definição de diferentes interpretações; a **Profundidade (P)**, refere-se a um contexto particular do qual é realizada a estratificação em diferentes níveis epistemológicos; e a **Lateralidade (L)**, diz respeito ao conhecimento de suporte a um dado objeto do domínio.

Seja um domínio particular de conhecimento D aquele em que o STI tem especialidade, de acordo com (COSTA, 1997), associa-se a este domínio D um conjunto de contextos distintos C :

$$D \rightarrow \{C_1, C_2, \dots, C_n\}, \quad (2.1)$$

onde, a cada um destes contextos com um índice fixo i , associa-se um conjunto de profundidades diferentes de índice j :

$$C_i \rightarrow \{P_{i1}, P_{i2}, \dots, P_{ij}\}, \quad (2.2)$$

onde cada par contexto e profundidade possui um conjunto de lateralidades L de índice t :

$$\langle C_i, P_j \rangle \rightarrow \{L_{ij1}, L_{ij2}, \dots, L_{ijt}\}. \quad (2.3)$$

Na *visão interna*, cada uma destas dimensões é levada em consideração como um plano: plano pedagógico, plano de problemas e plano de

suporte, respectivamente. Cada conhecimento associado a um sub-domínio é organizado em um ou mais currículos. O currículo leva o nome de *curriculum*. Cada *curriculum* consiste de um conjunto de Unidades Pedagógicas (UP), onde cada *up* é associada a um conjunto de problemas, e leva o seguinte formalismo:

$$\text{curriculum} = \{up_1, up_2, \dots, up_n\} \quad (2.4)$$

Cada Problema contém Unidades de Interação (UI) que são unidades mais refinadas para apresentar os problemas ao estudante e que são efetivamente responsáveis pela interação com o estudante.

Novos sistemas fazendo uso do modelo conceitual MATHEMA têm sido propostos em diversos sentidos, com iniciativas que fizeram uso de novas e diferentes formas de autoria de cursos, de criação de modelos de aprendiz, de interação adaptativa, entre outros. Nisto, surgiu o Projeto MATHNET e destacaram-se arcabouços e sistemas de autoria, em especial o Mathtutor.

2.4.1.1 Projeto MATHNET

O Mathnet (LABIDI, 2000) foi inicialmente concebido como um ambiente de sistema multiagente para o uso em STIs, fazendo uso do modelo conceitual MATHEMA.

O projeto MATHNET, no entanto, foi uma iniciativa que abrangeu as universidades federais de Alagoas (UFAL), do Maranhão (UFMA) e de Santa Catarina (UFSC), com o incentivo do CNPq (TEIXEIRA; LABID; NASCIMENTO, 2002). Posteriormente, mesmo com o término do incentivo, uma razoável quantia de outros trabalhos acadêmicos expandiram o desenvolvimento alcançado pelo projeto. Assim sendo, houveram trabalhos visando incluir aprimoramentos ao sistema e sugestões de novos modelos e soluções com aplicações no decorrer dos avanços nas diversas áreas envolvidas no projeto, dos quais o Quadro 1 mostra alguns trabalhos que baseiam-se no MATHEMA.

Quadro 1 – Trabalhos Derivados do Sistema MATHEMA

Autoria	Sistema	Contribuição
(COSTA, 1997)	MATHEMA	Protótipo de STI (LISP), Modelo Conceitual
(LABIDI, 2000)	MATHNET	Ambiente de STI, Projeto de STIs
(GÓIS, 2000)	MATHEMA	Especificação por Redes de Petri (RdP)
(ASSUNÇÃO, 2001)	MATHEMA	STI de Harmonia Tradicional
(FARIA et al., 2001)	MathTutor v1	STI com frames
(BORGES, 2002)	Ext. MATHNET	Solucionador de problemas
(COSTA, 2002)	Ext. MATHNET	Ferramenta de autoria
(FILHO, 2002)	Ext. MATHNET	Agente Estratégico
(FRIGO et al., 2002)	MathTutor v2	STI focado no modelo MATHEMA
(CARDOSO et al., 2004)	Ext. MathTutor	Redes Petri Objeto (RPO)
(ALMEIDA, 2004)	COMPOR Ext. MATHEMA	Arcabouço de STI (Metodologia, Modelo de Componentes e Framework)
(PINTO, 2006)	MATHEMA	Plataforma para Construção de AIA
(YAMANE et al., 2006)	Ext. MathTutor	Ferramenta de Autoria
(SECCO, 2007)	Ext. MATHEMA	Um AIA em Fração
(FRIGO et al., 2007)	FAST Ext. MathTutor	Modelo de Autoria para STIs Adaptativos
(POZZEBON et al., 2008)	ASTI Ext. MathTutor	Arcabouço de STI e ferramenta de autoria utilizando RPO
(SIBALDO, 2010)	Ext. MATHEMA	Interações Cooperativas e Adaptativas
(OLIVEIRA et al., 2012)	Ext. MATHEMA	Modelo integrando MATHEMA a jogos MMORPG
(TEIXEIRA, 2013)	MACSA Ext. MATHNET	Modelagem do aprendiz em ambientes cooperativos

Para fins de esclarecimento sobre o uso da nomenclatura encontrada na literatura de STIs envolvendo o MATHEMA, diferenciam-se: MATHEMA, com distinção entre modelo conceitual, arquitetura de ambiente multiagente, e STI; Mathnet, para um ambiente multiagente baseado em específico no modelo conceitual MATHEMA; MATHNET, para o projeto de STI e trabalhos posteriores inspirados pelo Mathnet; e MathTutor, arcabouço de STI integrante do projeto MATHNET, em conjunto com os trabalhos posteriores inspirados neste arcabouço. Na literatura, ainda é possível encontrar os mesmos termos designados de formas diferentes, com variantes como Math_Net e MATHTUTOR. Além disto, frequentemente encontra-se o termo “MATHEMA” como sendo exclusivamente um “arcabouço”, referindo-se à abreviação de “arcabouço do modelo conceitual MATHEMA”.

A definição do termo “arcabouço”, no entanto, é um pouco nebulosa. Murray (1999) diferencia os arcabouços de STIs (*shells*, em inglês), dos sistemas de autoria (também chamados de ferramentas de autoria): um arcabouço é um *framework* genérico para construção de STIs, enquanto que um sistema de autoria é um arcabouço STI junto com uma interface de usuário que permite a não programadores formalizar e visualizar o seu conhecimento. Posteriormente, Murray (2003) fez uma revisão de sistemas de autoria, classificando aplicações.

Apesar disto, não é incomum encontrar na literatura o uso de mais de

um “arcabouço” desempenhando alguns papéis para diferentes módulos do sistema, como no caso do COMPOR (ALMEIDA, 2004) que apresenta modelagem por componentes no arcabouço de modelo conceitual MATHEMA. Outro exemplo é o MathTutor (FRIGO et al., 2002), que integra parte do projeto MATHNET. Mathtutor é um STI provido de um sistema de autoria a partir de Redes de Petri Objeto que são traduzidos para a ferramenta *Java Expert System Shell* (JESS). O JESS é um arcabouço para sistemas especialistas que serve para a criação de regras. No Mathtutor, ele torna o sistema capaz de gerar inferências sobre as informações contidas no arcabouço de modelo conceitual MATHEMA.

O MathTutor recebe uma atenção especial por ser uma proposta de STI bem desenvolvida, tendo como base o Projeto MATHNET. Os trabalhos do MathTutor tiveram início na UFSC, a partir da iniciativa do professor Guilherme Bittencourt, que ministrava aulas sobre o MATHEMA, em conjunto com alguns de seus aprendizes.

O arcabouço de STI Mathtutor, do projeto MATHNET, visto na Seção 2.4.1.1, utiliza Redes de Petri Objeto (RPO) para a autoria de cursos a partir de uma tradução para o JESS. Contudo, a área de agentes possui uma grande quantidade de trabalhos desenvolvidos com Lógica Temporal, principalmente envolvendo a formalização de crenças em agentes.

2.5 LÓGICA TEMPORAL

Lógica Temporal é toda a lógica que lida com o tempo, como as lógicas modal, deôntica, de tempos verbais, intervalos, entre outros. Ainda dentro deste escopo, Fisher (2011) define a Lógica Temporal como uma extensão da lógica clássica, na qual o tempo torna-se um parâmetro a mais, modificando a verdade das declarações lógicas.

Em Ciência da Computação, desde Pnueli (1977), a Lógica Temporal demonstra-se uma boa forma de especificação e verificação da corretude de modelos em programas na sua versão linear: *Linear Temporal Logic* (LTL). Posteriormente, Ben-Ari, Manna e Pnueli (1981) formalizaram uma versão ramificada dela, a *Unified system of Branching Time*(UB), que mais uma vez estendida deu origem à *Computer Tree Logic* (CTL). Apesar de ter uma grande complexidade computacional, característica da área de agentes, a CTL é bem expressiva e constitui um fragmento da Lógica de Primeira Ordem ainda decidível.

2.5.1 Linear Temporal Logic (LTL)

A LTL faz menção a um conjunto de Lógicas Temporais que são lineares, ou seja, apresentam apenas um caminho de fórmulas para o seu futuro. Em uma Lógica Temporal Proposicional, segundo Emerson (1990), a porção não temporal (ou seja, não modal) da lógica é apenas lógica clássica proposicional, correspondente ao nível mais abstrato de raciocínio. A LTL pode, no entanto, ser estendida em expressões construídas de variáveis, constantes, funções, predicados, e quantificadores, até chegar à Lógica Temporal de Primeira Ordem, que é indecidível.

2.5.1.1 Sintaxe de uma Lógica Temporal Proposicional

Inicialmente, apresenta-se uma Lógica Temporal Proposicional, do inglês *Propositional Temporal Logic* (PTL), como descrita em Fisher (2011), com a finalidade de construir uma Lógica Temporal a partir de uma sintaxe básica. A forma normal desta lógica é descrita como *Separated Normal Form* (SNF). A PTL caracteriza uma sequência linear, discreta e tem passado finito, ou seja, contém um início. Fórmulas em PTL são construídas a partir dos seguintes elementos segundo a sua sintaxe formal (FISHER, 2011):

- Um conjunto finito de símbolos proposicionais, PROP, tipicamente representado por cadeias de caracteres alfanuméricas minúsculas, como por exemplo, relativas ao aprendiz, oa, domínio, entre outras, usadas na implementação desta dissertação. Os símbolos φ e ψ representam proposições arbitrárias para uso na semântica da lógica.
- Conectivos proposicionais: $\neg, \vee, \wedge, \Leftrightarrow, e \Rightarrow$.
- Conectivos temporais: $\bigcirc, \diamond, \square$, **início**, \mathcal{U} e \mathcal{W} .
- Parênteses, “(” e “)”, geralmente para evitar ambiguidade.

Exceto pelos conectivos (ou operadores) temporais, os outros elementos são convencionais em lógica clássica, portanto, apenas os operadores lógicos temporais necessitam de espaço para explicação.

O operador **início** delimita o valor em que, no passado finito, o tempo começa a ser considerado pelo sistema.

O operador \bigcirc referencia o próximo estado. A fórmula temporal $\bigcirc\varphi$ significa, de forma simples, que o agente possui uma crença de que no momento imediatamente posterior ao atual a proposição φ será verdadeira. Portanto, se o tempo atual t está em um modelo que considera a transição de

estados/momentos/mundos em relação aos números naturais (este é o caso considerado neste trabalho), a crença $\bigcirc\varphi$ é satisfeita, afirmando que φ é satisfeito no momento $t + 1$.

O \square é, desde a lógica modal, o operador de necessidade. Ele garante a proteção de propriedades no sistema, importantes para a manutenção de condições do ambiente. Serve para garantir que determinadas proposições ditas seguras não mudam ou o agente deve se assegurar que continuem iguais durante a temporalidade considerada.

O \diamond é, desde a lógica modal, o operador de possibilidade. O operador de possibilidade garante uma propriedade denominada de vivacidade. Isto é, $\diamond\varphi$ garante que, em dado momento do presente e/ou futuro, uma ou mais vezes, φ deverá acontecer. Ele é, de forma simples, um objetivo no sistema.

Os operadores temporais de próximo momento \bigcirc , possibilidade \diamond e necessidade \square são usados nesta lógica agregados de apenas uma proposição e por isto são denominados operadores unários. Os operadores \mathcal{U} e \mathcal{W} , no entanto, são operadores binários porque uma proposição é ligada a outra.

O *until* \mathcal{U} é um operador binário que induz um objetivo em uma duração, por exemplo, em $\varphi\mathcal{U}\psi$, a proposição φ acontece agora e sempre até um determinado futuro ($\diamond\psi$) até que a condição ψ seja satisfeita, ou seja, a proposição φ é satisfeita em todos os momentos discretos do atual ao futuro até ψ acontecer. Esta semântica apenas descreve este período de duração e nada afirma sobre as proposições após ψ acontecer.

Já o *unless* \mathcal{W} é um operador binário em que $\varphi\mathcal{W}\psi$ afirma que φ é satisfeito em todos os momentos discretos do atual ao futuro até que ψ for verdade, mas não há garantia de que ψ ocorra. Em outras palavras, esta duração pode acontecer infinitamente durante o tempo em que é considerada a temporalidade do sistema, pois as tentativas de ψ acontecer também podem falhar infinitamente nesta temporalidade. Por isto, existe a chance de que a proposição ψ nunca seja satisfeita.

Os operadores temporais podem ser categorizados pelas suas propriedades como a sequência determinística no operador de *próximo momento* \bigcirc (como em autômatos finitos determinísticos), o uso em operadores de proteção no operador de *necessidade* \square e vivacidade no operador de *possibilidade* \diamond (como em autômatos de Büchi), e os operadores de duração como o *até* \mathcal{U} e o *ao menos que* \mathcal{W} .

Os elementos da PTL são operadores lógicos (ou conectivos lógicos) de futuro, no entanto, também é possível utilizar de operadores lógicos temporais em agentes dedutivos para representar passado, servindo de parâmetro para inferências dado um determinado histórico.

Pseudo-código 1 Ciclo de um Agente *METATEM*

- 1: Atualizar história recebendo mensagens;
 - 2: Checar regras disparadas;
 - 3: Executar conjuntamente as regras com comprometimento sobre ciclos prévios;
 - 4: Voltar ao passo 1.
-

2.5.1.2 Agentes Deliberativos *METATEM*

Os agentes de raciocínio dedutivo representam simbolicamente fórmulas lógicas e a manipulação sintática corresponde à dedução lógica, ou prova de teoremas (WOOLDRIDGE, 2001). Seus dois componentes principais são: uma interface, responsável por interagir com o ambiente e um motor computacional, responsável por definir atos. A interface deste tipo de agente consiste de três componentes conjuntivos:

- um identificador;
- conjunto de símbolos definindo que mensagens aceitar - proposições de ambiente;
- conjunto de símbolos definindo que mensagens enviar - proposições de componente.

É o caso do *METATEM*, de Barringer et al. (1990), que recebe este nome por fazer uma especificação de lógica *TEMP*oral com *META*-nível de raciocínio. A execução do agente *METATEM* obedece a um ciclo de tentativas de acordo com regras em sua história, onde seus antecedentes correspondem ao passado e os gatilhos acionam os consequentes de futuro, como pode ser mostrado no Pseudo-código 1.

A sintaxe da *Propositional MetateM Logic* (PML), de (BARRINGER et al., 1990), possui os seguintes elementos: alfabeto de símbolos proposicionais \mathcal{A}_P , que é a união dos conjuntos disjuntos de componente \mathcal{A}_C e ambiente \mathcal{A}_E . Ela usa operadores da PTL que já estendem os operadores proposicionais da lógica clássica, mais operadores temporais de passado. Uma fórmula-bem-formada (fbf) desta lógica contém as subclasses:

- $\text{fbf}_=$ é o conjunto de proposições p de \mathcal{A}_P e é fechado sobre os operadores proposicionais clássicos;

- $\text{fbf}_{<}$ é o conjunto de fórmulas que estão fechadas sobre proposições e os operadores do passado (estrito), onde as proposições estão em fbf_{\leq} , ou seja, na união dos conjuntos $\text{fbf}_{=}$ e $\text{fbf}_{<}$;
- fbf_{\geq} é o conjunto de fórmulas que estão fechadas sobre proposições e os operadores de futuro (não-estrito).

O METATEM promove o encontro das abordagens declarativa e imperativa, onde a PML declara proposições de passado e ações futuras. Portanto, a PML possui passado declarativo e futuro imperativo a partir de uma extensão da lógica proposicional clássica através de seus operadores temporais, sendo linear e de tempo discreto.

No Quadro 2 são listados os símbolos dos operadores temporais de passado e futuro da PML. Na proposta, Capítulo 4, estes operadores são utilizados na definição de conjuntos de semântica específica adaptados ao contexto temporal dos artefatos.

Quadro 2 – Operadores da Lógica Temporal PML

Símbolo	Nome	Semântica Genérica
<i>Operadores Unários de Futuro</i>		
$\bigcirc\varphi$	<i>Next</i>	φ é satisfeito no próximo momento
$\diamond\varphi$	<i>Sometime</i>	φ é satisfeito agora ou num futuro momento
$\square\varphi$	<i>Always</i>	φ é satisfeito para todo o futuro
<i>Operadores Unários de Passado</i>		
$\bullet\varphi$	<i>Weak last</i>	φ é satisfeito no momento inicial
$\odot\varphi$	<i>Strong last</i>	φ é satisfeito no momento anterior
$\diamond\varphi$	<i>Was</i>	φ é satisfeito em algum momento no passado
$\blacksquare\varphi$	<i>Herefore</i>	φ sempre satisfeito no passado
<i>Operadores Binários de Futuro</i>		
$\varphi\mathcal{U}\psi$	<i>Until</i>	φ é satisfeito até que ψ ocorra
$\varphi\mathcal{W}\psi$	<i>Unless</i>	φ é satisfeito mesmo que ψ não ocorra
<i>Operadores Binários de Passado</i>		
$\varphi\mathcal{S}\psi$	<i>Since</i>	φ foi satisfeito desde que ψ ocorreu
$\varphi\mathcal{Z}\psi$	<i>Zince</i>	φ foi satisfeito mesmo que ψ nunca ocorreu

Os operadores de lógica do futuro são definidos semanticamente na PTL, já os de passado são parecidos: o operador *Weak last* ($\bullet\varphi$) é equivalente ao **início** do PTL. O operador *Strong last* ($\odot\varphi$) significa que φ é satisfeito por acontecer no tempo imediatamente anterior ao atual, $t - 1$. O operador *Was* ($\diamond\varphi$) significa que φ é satisfeito por ter ocorrido ao menos uma vez no

passado. O operador *Herefore* ($\blacksquare\varphi$) significa que φ é satisfeito por sempre ocorrer na temporalidade considerada do passado. Para o operador binário *Since*, a duração faz com que em $\varphi\mathcal{S}\psi$, φ tenha sido satisfeito até o momento anterior desde que ψ ocorreu ($\blacklozenge\psi$), e com o operador *Zince* em $\varphi\mathcal{L}\psi$, φ foi satisfeito até o momento anterior, mesmo que ψ nunca tenha ocorrido.

Uma implementação mais recente do METATEM é o CONCURRENT METATEM (FISHER, 1994), e compreende um sistema que executa agentes de maneira concorrente e capazes de comunicação.

Para a proposta desta dissertação faz-se uso de um conjunto da terminologia dos operadores lógicos PML em Barringer et al. (1995), com semântica própria para o sequenciamento de Objetos de Aprendizagem, como visto no Quadro 4, do Capítulo 4.

2.5.1.3 Semântica de Lógica Temporal

A PML possui uma Estrutura de Kripke (KRIPKE, 1963) na qual a relação de acessibilidade ocorre a partir dos valores booleanos das proposições, que configuram o seu alfabeto, ou seja, com as possibilidades de verdadeiro ou falso nas proposições para o caminho da árvore computacional: $2^{\mathcal{A}_p}$.

Lógicas temporais, assim como nas lógicas modais, são tipicamente Estruturas de Kripke na forma:

$$\mathcal{M} = \langle S, R, \pi \rangle, \quad (2.5)$$

onde:

- S é o conjunto de momentos no tempo;
- R é uma relação de acessibilidade temporal;
- $\pi : S \mapsto \mathbf{P}$ (PROP) é uma função de transição que mapeia cada momento/mundo/estado para um conjunto de proposições.

No caso do trabalho aqui proposto, para o sequenciamento através de índice, o mapeamento é feito para o conjunto dos números naturais \mathbb{N} ,

$$\mathcal{M} = \langle \mathbb{N}, \pi \rangle, \quad (2.6)$$

onde $\pi : \mathbb{N} \mapsto \mathbf{P}$ (PROP) mapeia um número para um momento discreto:

$$(\mathcal{M} \times \mathbb{N} \times \mathbf{fbf}) \rightarrow \mathbb{B}. \quad (2.7)$$

Para acessar o valor verdade de uma fórmula-bem-formada **fbf** em um modelo \mathcal{M} , cria-se uma relação de interpretação, permitindo acessibilidade a

outros mundos possíveis, a partir de um índice temporal i , e fórmula φ :

$$\langle \mathcal{M}, i \rangle \models \varphi, \quad (2.8)$$

intuitivamente significando que no modelo \mathcal{M} , no instante i , φ é verdadeira.

2.5.2 Computer Tree Logic (CTL)

A árvore lógica computacional, *Computational Tree Logic* (CTL) em inglês, é uma extensão da LTL que adiciona quantificadores de caminho. Isso deve-se pelos operadores de símbolo A ("para todos os futuros") e E ("para algum Futuro"). Ela foi criada por Clarke e Emerson (1982) que posteriormente a estenderam para uma lógica mais completa, denominada CTL*.

A lógica BDI proposta por Rao (1991), usa a modalidade temporal CTL*. A implementação de um conjunto executável da lógica BDI proposta resultou no AgentSpeak (RAO, 1996a). Em Rao (1996b), no entanto, o autor também se preocupa especificamente com o aspecto linear para correteude e em um estudo anterior aborda diferenças assimétricas entre lógicas temporais ramificadas e lineares (RAO; GEORGEFF, 1991).

A lógica BDI_{CTL} é uma CTL* com conjuntos de crenças, BEL, desejos/objetivos, GOAL, e intenções, INTEND. Os principais operadores temporais são: próximo \bigcirc , eventualmente \diamond , até \mathcal{U} , e inevitável A. Para fins de comparação, o problema de teste de satisfazibilidade da LTL é de complexidade computacional PSPACE-completo, enquanto que a CTL* é completa, para tempo determinístico, em 2EXP (EMERSON, 1990).

O Jason, de Bordini, Hübner e Wooldridge (2007), é uma ferramenta de programação, contendo um interpretador de AgentSpeak(L), escrita em Java e com sintaxe parecida com PROLOG. A estratégia de um agente BDI no Jason, conforme Seção 2.1.2, segue o comprometimento liberal.

Para Wooldridge (2000), um agente BDI que alcança o balanço entre as recomendações das intensões ousada e cautelosa, efetuará o Pseudo-código 2, onde: B_0 são as crenças iniciais, I_0 são as intenções iniciais, brf é a função de revisão de crenças e π é um plano.

Pseudo-código 2 Ciclo de BDI no Jason

```

1:  $B \leftarrow B_0$ ;
2:  $I \leftarrow I_0$ ;
3: Enquanto verdade faça
4:   receba a percepção  $\rho$  pelos sensores;
5:    $B \leftarrow bnf(B, \rho)$ ;
6:    $D \leftarrow opções(B, I)$ ;
7:    $I \leftarrow filtro(B, D, I)$ ;
8:    $\pi \leftarrow plano(B, I)$ ;
9: Enquanto não (vazio( $\pi$ ) ou satisfeito( $I, B$ ) ou impossível( $I, B$ )) faça
10:    $\alpha \leftarrow$  primeiro elemento de  $\pi$ ;
11:   execute( $\alpha$ );
12:    $\alpha \leftarrow$  cauda de  $\alpha$ ;
13:   observe o ambiente para a próxima percepção  $\rho$ ;
14:    $B \leftarrow bnf(B, \rho)$ ;
15:   Se reconsiderar( $I, B$ ) então
16:      $D \leftarrow opções(B, I)$ ;
17:      $I \leftarrow filtro(B, D, I)$ ;
18:   Se não plausível( $\pi, I, B$ ) então
19:      $\pi \leftarrow plano(B, I)$ ;

```

No artigo (WOOLDRIDGE, 1996), a execução do METATEM é comparada à BDI, onde na sua *fbf* as cláusulas de crenças são fechadas para o passado, as cláusulas dos desejos são fechadas para presente e futuro, e as cláusulas de intenções são fechadas para regras de passado-futuro, respectivamente: PML^- , são as fórmulas de histórico, apenas passado; PML^+ , são comprometimentos; e PML^\pm , são regras de gatilho *passado* \Rightarrow *futuro*.

Outros usos de Lógica Temporal em sistemas BDI:

- **sistema intencional de Cohen e Levesque**: Cohen e Levesque (1990) utilizaram LTL agregando como modalidade um conjunto de desejos e um conjunto intenções;
- **LORA** (*Logic Of Rational Agents*): formalizada por Wooldridge (2000), **LORA** combina quatro componentes distintos:
 1. **um componente de primeira-ordem**: Este componente permite representar as propriedades dos objetos num sistema sobre estado, e como estes objetos se relacionam uns com os outros;
 2. **um componente BDI (similar ao de Cohen e Levesque)**: Tem por finalidade expressar crenças, desejos e intenções de agentes no sistema;

3. **um componente temporal (CTL)**: Permite representar os aspectos dinâmicos do sistema - como eles variam conforme o tempo;
 4. **um componente de ação (CTL)**: Mapeia as ações que os agentes realizam e os estados esperados resultantes dessas ações.
- **KARO**: O KARO (MEYER; HOEK; LINDER, 1999) (Knowledge and belief, Abilities, Results and Opportunities) expande a lógica de ação com lógica epistêmica (doxástica), usando operadores de conhecimento e crença. Nela, os operadores de desejo e intenção são baseados na Lógica Temporal CTL*. Adicionam-se a estes, operadores de Habilidades, Resultados e Oportunidades de lógica dinâmica. É uma lógica multi-modal que está no sistema normal S5.

O Quadro 3 contém uma sucinta comparação entre as lógicas temporais e suas aplicações, vistas nesta seção.

Com base neste capítulo de Fundamentação Teórica, evidencia-se a existência de diferentes áreas nas quais a proposta desta dissertação se encontra inserida. Dentro destas áreas, o Capítulo 3 compara trabalhos relacionados, iniciando com os usos de padrões de sequenciamento de OAs na literatura, Seção 3.1, para a posterior comparação com os STIs recentes que fazem uso do modelo conceitual MATHEMA, na Seção 3.2, e, por fim, a Seção 3.3 compara os trabalhos relacionados à Lógica Modal.

Quadro 3 – Comparativo de Lógicas com Ênfase na Temporalidade

Sigla¹	Nome	Aplicação	I/R²	Autor
LTL	Linear Temporal Logic	Teórico	L	(PNUJEL, 1977)
PDL	Propositional Dynamic Logic	Teórico	L	(FISCHER; LADNER, 1979)
Part. LTL	Temporal Horn Clauses	TEMPLOG	L	(ABADI; MANNA, 1989)
UB (Ext. LTL)	Unified system of Branching time	Teórico	R	(BEN-ARI; MANNA; PNUJEL, 1981)
Ext. LTL	Intenções de Cohen e Levesque	Teórico BDI	L	(COHEN; LEVESQUE, 1990)
CTL (Ext. UB)	Computational Tree Logic	Teórico	R	(CLARKE; EMERSON, 1982)
PML (Ext. LTL)	Propositional MetaM Logic (Passado/Futuro)	CONCURRENT METATEM	L	(BARRINGER et al., 1990)
CTL* (Ext. CTL)	BDI Logics	Teórico	R	(RAO, 1991)
KARO (Multimodal)	Knowledge, Abilities Results and Opportunities	Teórico	R	(MEYER; HOEK; LINDER, 1999)
<i>LOG_RA</i> (Ext. BDI _{CTL})	Logic Of Rational Agents	Teórico	R	(WOOLDRIDGE, 2000)

¹ Part. = parte da, Ext. = extensão da

² Linear ou Ramificado

3 TRABALHOS RELACIONADOS

O problema do sequenciamento de aprendizagem é muitas vezes modelado como um problema da satisfação de restrições (WAN; NIU, 2014) e algumas técnicas de IA usam heurísticas para otimização tais como: algoritmos de colônia de formigas (YANG; WU, 2009) e de enxame de partículas (DHEEBAN et al., 2010); programação genética (HONG et al., 2007; SHMELEV; KARPOVA; DUKHANOV, 2015); redes neurais (BUTZ; HUA; MAGUIRE, 2006); planejamento (GARRIDO; ONAINDIA, 2013); sequenciamento baseado em regras (BRA; SMITS; STASH, 2006); e grafos para escolha de caminho (KARAMPIPERIS; SAMPSON, 2005; SANGINETO et al., 2008).

A grande maioria destes trabalhos fazem uso de Objetos de Aprendizagem (OAs) e com o advento da *Web* semântica, ontologias foram criadas e estendidas para compartilhar conceitos em um formato que é compreensível para máquinas. Recentemente, surgiram padrões de especificação para o Sequenciamento e Navegação de “caminhos de aprendizagem”, do inglês *learning paths*, utilizando OAs para as atividades interativas e fazendo uso de padrões de sequenciamento, como descritos na Seção 2.3.

3.1 TRABALHOS RELACIONADOS COM OBJETOS DE APRENDIZAGEM

Algumas poucas iniciativas ainda criam ontologias de sequenciamento próprio diretamente do LOM, como no caso de (MARCOS et al., 2008) e trabalhos como o *framework* LECOMPS (STERBINI; TEMPERINI, 2009) que fazem o sequenciamento de recursos educacionais em repositórios específicos, nomeados de *Pools*, sendo estendido em outros trabalhos (LIMONGELLI et al., 2009; STERBINI; TEMPERINI, 2010). A grande desvantagem destes trabalhos é o uso de padrão de ontologia próprio para seus OAs, criando dependência para seus LMS e repositórios.

Por outro lado, o padrão de metadados SCORM é o mais frequentemente utilizado. Dentre os trabalhos utilizando SCORM existem aqueles que o estendem com a finalidade de sequenciamento (CHEN, 2009; DEMARCOS et al., 2009; SÁNCHEZ-BRITO; RUIZ-ASCENCIO; GARCÍA-HERNÁNDEZ, 2014; LENDYUK et al., 2015; MAHMOUD et al., 2015) e trabalhos que usam o sequenciamento fornecido apenas pelo SCORM SN (ZHU; YAO, 2009; ARCE-C’RDENAS; GARCIA-VALDEZ, 2012).

O grupo de pesquisa em Inteligência Artificial e Tecnologia Educaci-

onal (IATE), da Universidade Federal de Santa Catarina, vem contribuindo com soluções educacionais que visam adaptação do conteúdo ao Aprendiz. Palomino, Silveira e Nakayama (2015) propõem um modelo de STI que integra LMS para a criação de um ambiente inteligente de aprendizagem aproveitando características de ambos, STI e LMS. Já Amorim Jr. e Silveira (2015) utilizam de Objetos de Aprendizagem Inteligentes e o sequenciamento do SCORM para seleção dinâmica visando adaptação e reuso.

Outros trabalhos envolvendo sequenciamento utilizam tempo como parâmetro, mas considerando o controle da duração de atividades e agendamento de atividades (KAMSA et al., 2015; LENDYUK et al., 2015). Contudo, nenhum trabalho encontrado usa de aspectos temporais na área de educação conforme tratados nesta dissertação: utilizando de lógica modal e tempos verbais.

O trabalho mais similar ao desta dissertação com Objetos de Aprendizagem é o de Garrido, Morales e Serina (2016). Este trabalho utiliza de planejamento para o sequenciamento de OAs, porém não na forma de crenças em agentes, e preocupando-se com uma ontologia própria para o planejamento. Como nesta dissertação este trabalho necessita de uma conversão para uma forma fácil para lidar com proposições, considerando pré e pós-condições, e onde as anotações dos OAs são feitas por *designers* antes da execução do sistema. O diferencial da proposta desta dissertação em relação a de (GARRIDO; MORALES; SERINA, 2016), além de características específicas como a utilização de BDI ao invés de um planejamento clássico e de usar um modelo simples para anotação e não uma ontologia, são: uma estrutura mais consistente de domínio e o suporte a aspectos de temporalidade do Aprendiz.

O modelo conceitual MATHEMA, utilizado na proposta desta dissertação, é uma solução para a estrutura do domínio de forma consistente a partir das dimensões. Na busca por soluções foi encontrado o trabalho de Zhiping, Yu e Tianwei (2011), um modelo formal para sistemas de recomendação de conteúdo, que é compatível com a definição de níveis de domínio, mas menos clara que as relações do modelo conceitual MATHEMA.

Atualmente, o SCORM é o padrão de especificação utilizado para o sequenciamento de atividades com OAs. O sequenciamento ocorre por meio de árvores de atividades em níveis hierárquicos (granularidade) para descrever cursos, módulos e atividades (MELIA; BARRETT; PAHL, 2006).

Por decisão de projeto, escolheu-se especificar o sequenciamento na forma de predicados no modelo conceitual MATHEMA para não apenas descrever atividades com OAs, mas compor um plano de forma automática que pode posteriormente ser convertido para o formato desejado.

3.2 TRABALHOS DA ÁREA DE SISTEMAS TUTORES INTELIGENTES

O uso de Redes Petri Objeto (RPO) no Mathtutor faz com que a rede considere os estados das atividades pedagógicas de uma forma atômica, como a que ocorre atualmente com OAs. As ações de interação pedagógica são tratadas como estados/lugares, no entanto, RPOs não consideraram tanto a reutilização de recursos hipermídia. Os trabalhos envolvendo Redes de Petri, largamente utilizadas nos sistemas tutores usando MATHEMA, contudo, não consideraram a temporalidade nas crenças dos agentes do sistema.

O arcabouço de STI Mathtutor (YAMANE et al., 2006; FRIGO et al., 2007; POZZEBON et al., 2008), do projeto MATHNET, como apresentado na Seção 2.4.1.1, conseguiu diversas soluções de personalização por meio de ferramentas de autoria feitas para uso dos Especialistas do Domínio. A proposta do Mathtutor se assemelha a esta dissertação por ter um mecanismo (um arcabouço) para a criação de regras de curso (JESS) e o seu sistema de autoria facilita a montagem de cursos no modelo conceitual MATHEMA.

O trabalho desta dissertação é distinto dos trabalhos com MATHEMA na maneira de considerar a especificação, enquanto que eles fazem uso de Redes de Petri Objeto, esta proposta faz uso de temporalidade explícita ao Agente Tutor, de acordo com o contexto inerente aos artefatos.

Apesar de Redes de Petri comportarem a especificação de aspectos temporais em seus estados e transições concorrentes, a Lógica Temporal fornece o controle de especificação por meio de tempo verbal (passado, presente/futuro) representando mundos possíveis e acesso às fórmulas lógicas e inferência, não apenas por mudanças de estado com transição. O JESS faz uso de condições gatilho para a inferência, mas não considera a temporalidade. Portanto, a Lógica temporal é melhor para a definição de crenças de agente e em específico para a deliberação para aspectos de temporalidade.

3.3 LÓGICA MODAL

As maiores inspirações para esta proposta provém de trabalhos comparando BDI e Lógica Temporal na formalização de agentes: o modelo de BDI em (RAO, 1995) utiliza lógica modal para descrever dinamicidade, e por isso, lida em sua especificação com aspectos temporais, também conhecida através de uma família de lógicas BDI; em (WOOLDRIDGE, 1996) usa-se a PML para especificar semântica de passado, futuro e inferências semelhantes a regras de passado que infere futuro; e em (FISHER, 2005) também usa-se da PML para fins semânticos e incorporando o uso de recursos para realização de ações como parte do raciocínio do agente.

3.3.1 Trabalhos em Lógica BDI

LORA e *KARO* não possuem implementação, enquanto que a *PML* é implementada em agentes no *CONCURRENT METATEM*. Porém, como a Lógica Temporal lida com operadores de lógica clássica e um parâmetro a mais (modalidade), a semântica dos operadores da *PML* nesta dissertação é convertida para predicados em lógica, a serem utilizados no interpretador *Jason*, estendendo a linguagem *AgentSpeak(L)*.

A prática de fazer tal conversão, de Lógica Temporal para predicados em um ciclo de raciocínio BDI, não é a ideal para fins de verificação e especificação de sistemas. Porém, este abuso formal é empregado nesta dissertação com as finalidades de expressar temporalidade no Agente Tutor BDI, a partir da implementação dos operadores da *PML*, e de fazer uso do raciocínio dinâmico em seu ciclo deliberativo, a partir dos parâmetros nos contextos dos planos do mesmo. Desta forma, é possível escolher o melhor plano do agente no *Jason* por correspondência com os parâmetros de aspectos temporais e compor, no corpo do plano, uma sequência de ações.

Um outro exemplo de trabalho que faz uma conversão semântica de operadores modais é o de *Besold e Schiemann (2010)*, que converte modalidades para Lógica Descritiva em Sistemas Multicontexto (*SMC*), sem o uso dos aspectos temporais apresentados na *PML*.

Para verificação e especificação de sistemas utilizando Lógica Temporal, o ideal é o uso de ferramentas para a checagem de modelos formais em sistemas, tais como a linguagem para a modelagem de processos *Process Meta Language* (*Promela*) onde o sistema pode ser checado com o verificador de modelos *SPIN* (*NEUMANN, 2014*), mas estas ferramentas não são intuitivas para conceitos abstratos e ambientes complexos como, por exemplo, ambientes com agentes providos de crenças, desejos e intenções.

Os aspectos temporais para representação do conhecimento da proposta são tratados na Seção 4 para elucidar a estruturação do domínio de conhecimento e a abordagem temporal, dada pelo histórico no curso e planejamento adequado dentro desta visão.

4 SMART ITS

“Todo o conhecimento humano começou com intuições, passou daí aos conceitos e terminou com ideias.”

(Immanuel Kant)

Neste capítulo, o modelo proposto neste trabalho é detalhado e desenvolvido tendo como hipótese que a sequência de interações de aprendizagem pode ser planejada automaticamente considerando aspectos temporais. A proposta desta dissertação baseia-se no STI MATHEMA para a criação de uma nova arquitetura, chamada de Smart ITS, integrando um novo Agente Tutor BDI, cujas crenças são representadas em Lógica Temporal. O Agente Tutor faz uso desta lógica para raciocinar sobre a estrutura de domínio, descrita no modelo conceitual MATHEMA (COSTA, 1997). Sobre esta estrutura, ele é capaz de planejar *curricula* de forma personalizada e adaptativa para as interações pedagógicas do Agente Aprendiz com o sistema. Conforme descrito na Seção 2.4, o STI MATHEMA é constituído por três tipos de agentes artificiais: Agentes Tutores – principal foco deste trabalho –, Agentes de Interface e Agentes Aprendizes. A dissertação, foca em específico no Sistema Tutor mostrado na Figura 4, na forma de um Agente Tutor BDI.

O uso de Lógica Temporal possibilita expressar o atual estado no qual o Aprendiz se encontra em relação ao Domínio de Conhecimento para a seleção e sequenciamento de OAs, equivalentes as atividades de interação pedagógica, de forma flexível. Tendo crenças temporais, o Agente Tutor é capaz de elaborar um artefato *curriculum* para o Agente de Interface por descrever as atividades de interação pedagógica adequadas ao Aprendiz. O raciocínio do Agente Tutor considera que as crenças são agregadas a aspectos temporais provenientes das informações que coleta de artefatos e dos planos prontos em AgentSpeak, desde o início de sua execução. Um plano BDI é selecionado considerando premissas temporais em seu contexto. Dentro do plano, isto resulta na escolha de ações a serem sequenciadas a partir da verificação da satisfação das proposições de interações pedagógicas com OAs. Assim, cada composição de plano elabora uma sequência personalizada ao Aprendiz, considerando a disciplina que cursa, no momento em que o plano é requerido. A interação continuada do Agente Tutor no sistema provê um sequenciamento de OAs de forma adaptativa e personalizada, assim como monitora as atividades do Aprendiz, alterando a percepção que o Agente Tutor possui do Aprendiz no STI em tempo de execução.

Nesta proposta, existem três artefatos precedentes à execução do sis-

tema, os quais devem ser elaborados por agentes humanos com atribuições de papéis distintos, citados na Seção 2.3:

- **Artefato de Domínio:** contém os dados do Domínio de Conhecimento na estrutura do modelo conceitual: contextos, profundidades e lateralidades, seguindo a Semântica de Plano, e as regras temporais para o sequenciamento. A criação e alteração deste artefato fica ao encargo de Especialistas de Domínio e *Designers* Instrucionais;
- **Artefato de OAs:** contém para cada OA: os dados de identificação do repositório, triplas RDF (*Resource Description Framework*) para a definição de outros metadados e anotações temporais de Semântica de Ação. O *Designer* de Conteúdo é o responsável por este artefato;
- **Artefato de Aprendiz:** contém três tipos de dados, no que tange o Aprendiz: dados pessoais, fornecidos e mantidos pelo Aprendiz; dados sobre modelo cognitivo e preferências, fornecidos por meio de avaliação, criada por um *Designer* Instrucional; e dados de conhecimento prévio, fornecidos por experiências passadas nas atividades do sistema.

Na implementação desta dissertação, o foco está no papel de planejamento do Agente Tutor BDI. A proposta de arquitetura aqui apresentada leva o nome de Smart ITS e consiste de um sistema educacional para a personalização e adaptação de um curso para o Aprendiz utilizando de Lógica Temporal para o sequenciamento de OAs.

A Figura 5 mostra um diagrama de visão geral da arquitetura Smart ITS, na qual a diagramação do ambiente multiagente seguiu a metodologia Prometheus (PADGHAM; WINIKOFF, 2005), a partir do *Prometheus Design Tool* (PDT) (GROUP, 2014). Decorrente da ausência de suporte ao paradigma de Agentes & Artefatos, o símbolo para artefato é representado como dado, sendo o mais próximo do conceito devido ao fato de que artefatos permitem acesso a banco de dados.

Os detalhes dos elementos da proposta de arquitetura desde trabalho são descritos na Seção 4.1. Posteriormente, na Seção 4.2 detalha-se o Agente Tutor e na Seção 4.3 é vista a representação da estrutura de domínio do Agente Tutor de acordo com o MATHEMA. A Seção 4.4 detalha como esta estrutura é utilizada como plano para a elaboração do *curriculum*, para então demonstrar as crenças sobre o domínio de conhecimento no Agente Tutor, Seção 4.4.1, e os conjuntos fazendo uso das semânticas de Passado, Ação e Plano nas subseções 4.4.2, 4.4.3 e 4.4.4. A Seção 4.5 contempla o ciclo de deliberação do Agente Tutor e a Seção 4.5.1 mostra um exemplo de implementação com Jacamo (BOISSIER et al., 2013), uma integração de Jason, Cartago e Moise.

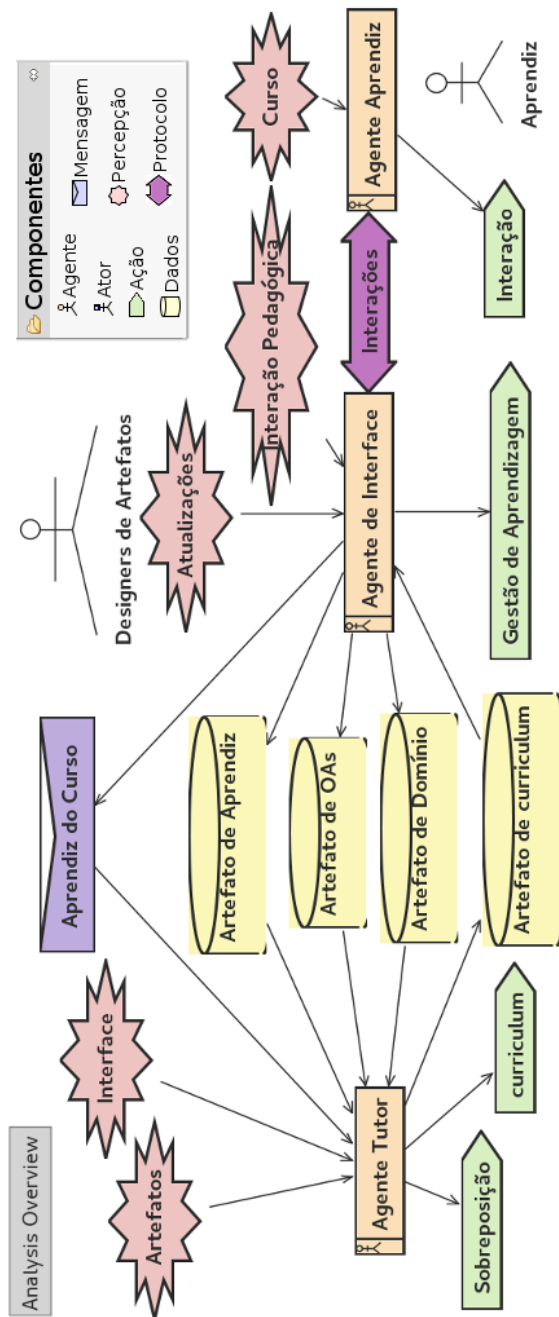


Figura 5 – Visão Geral da Arquitetura Smart ITS

4.1 ARQUITETURA DO SISTEMA MULTIAGENTE DE TUTORIA

Definição 2 *A arquitetura do Smart ITS é composta de uma sociedade de agentes A e ambientes virtuais de aprendizagem E : $SmartITS = \langle A, E \rangle$.*

Definição 3 *A sociedade de agentes A está projetada para possuir três tipos de agentes $A = \langle AA, AI, AT \rangle$, onde: $AA = \{aa_1, aa_2, \dots, aa_n\}$ é o conjunto de Agentes Aprendizes; $AI = \{ai_1, ai_2, \dots, ai_m\}$ é o conjunto de Agentes de Interface; e $AT = \{at_1, at_2, \dots, at_k\}$ é o conjunto de Agentes Tutores. Um agente no sistema pode ser apenas de um desses tipos.*

O Agente Aprendiz representa o Aprendiz físico que interage com os OAs, este podendo apenas ver e interagir com a Interface, na forma de um agente artificial. As atividades de interação pedagógica com o Aprendiz servem de entrada para o Smart ITS.

O Agente de Interface é responsável pelo controle das informações para o curso e serve de canal de comunicação entre o Agente Tutor e o Agente Aprendiz. Ele usa os OAs conforme a receita do plano contida no *curriculum* para efetuar as atividades de interação com o Aprendiz e armazena o resultado das atividades no Artefato de Aprendiz. O Agente de Interface também serve para intermediar o acesso de pessoas que possuem papéis de manutenção e atualização do Smart ITS, possibilitando a criação, alteração ou remoção dos artefatos para os quais possuem permissão.

O Agente Tutor é o especialista de um determinado Domínio de Conhecimento. É responsável pela tutoria dos Aprendizes inscritos no curso de mesmo domínio e, para tanto, tem acesso aos artefatos para a elaboração de *curricula* aos seus Aprendizes. Portanto, a principal função do Agente Tutor é o planejamento do *curriculum* a ser entregue para o uso do Agente de Interface, através dos dados coletados nos artefatos. O Agente Tutor BDI do Smart ITS tem como diferencial a percepção de contextos (não confundir com o termo do modelo conceitual MATHEMA) temporais na deliberação de seus planos para a interpretação do Artefato de Domínio, Artefato de Aprendiz e Artefato de OAs. Esta percepção ocorre conforme as semânticas dos operadores, da Seção 4.2, usados no ciclo de deliberação, Seção 4.5, para produzir o Artefato de *curriculum*. Por exemplo, o Agente Tutor pode acessar os registros de atividades do Agente Aprendiz e diagnosticar a situação dele no curso, planejar sobre as futuras ações sobre o domínio estruturado no modelo conceitual MATHEMA, e elaborar um curso de ações para seu curso através de uma sequência de OAs que melhor serve ao Aprendiz no momento.

Definição 4 *O ambiente de aprendizagem E , do Smart ITS, suporta dois tipos de Espaços de Trabalho: $E = \langle ETIP, ETEC \rangle$, onde o Espaço de Tra-*

balho de Interação Pedagógica (ETIP), $ETIP = \{etip_1, etip_2, \dots, etip_n\}$, é o conjunto de classes de aula que apenas permitem o acesso de agentes aprendizes e agentes de interface, $\{AA, AI\} \in ETIP \subseteq E$; e o Espaço de Trabalho de Elaboração de curriculum (ETEC), $ETEC = \{etec_1, etec_2, \dots, etec_m\}$, é o conjunto de salas de professores e que apenas permite acesso dos agentes de interface e agentes tutores, $\{AI, AT\} \in ETEC \subseteq E$.

Definição 5 *Um curso único no Smart ITS consiste de um Agente Tutor para a tutoria individual e personalizada a um Aprendiz, organizada no sistema multiagente em uma tripla de agentes, um de cada tipo: $\langle aa_n, ai_m, at_k \rangle$ localizados em um ambiente com instâncias únicas dos dois espaços de trabalho, $e = \langle etip_i, etec_j \rangle$, e onde o Agente de Interface, ai_m , é o mesmo em ambos espaços de trabalho, $etip_i$ e $etec_j$.*

Um Agente Tutor pode tutorear vários Agentes Aprendizes que se inscrevam em um mesmo curso, onde esse atua como especialista do domínio do curso. A proposta do Agente Tutor é genérica o suficiente para receber as informações necessárias do domínio a partir de um artefato descrevendo o curso em Artefato de Domínio. Poderia-se então criar apenas um Agente Tutor para todos os cursos, mas tanto por questões de identificação do agente responsável pelo Agente Aprendiz se tornar mais fácil, quanto por questões de tempo de resposta e sobrecarga, e devido a analogia de uma classe de aula, tomou-se a decisão de projeto de haver um Agente Tutor para cada curso. Sem deixar de elaborar o *curriculum* sempre de forma individual e personalizada à cada Aprendiz. Por outro lado, um Agente Aprendiz pode participar de vários cursos e, portanto, ter vários Agentes Tutores, um para cada curso.

Um único Agente de Interface também pode ser utilizado em vários cursos. Como ele é o agente responsável pelo controle dos cursos em um sistema de gestão de aprendizagem, LMS, faz sentido ter um Agente de Interface para cada instância de LMS para evitar redundâncias na identificação. Isto se deve ao fato de que o Agente de Interface toma conhecimento sobre quais são os Aprendiz e Agente Tutor em uma instância de curso único.

Portanto, um Agente Tutor pode tutorear vários Agentes Aprendizes em relação a um curso (aridade um para muitos); um Agente Aprendiz pode estar inscrito em vários cursos e, portanto, ter vários Agentes Tutores em relação aos diversos cursos em que está inscrito (aridade um para muitos); o Agente de Interface tem conhecimento e controle sobre todos os Agentes Tutores (um para muitos) e Agentes Aprendizes (um para muitos) na instância de um LMS; e por fim, um curso tem aridade de um Agente Tutor (um para um) para um Aprendiz (um para um) e com o intermédio de um Agente de Interface (um para um).

4.2 PROPOSTA DE AGENTE TUTOR

Nesta seção a proposta do Agente Tutor BDI é detalhada. Nela, aproveita-se as estruturas robustas e comprometimento com o domínio em conjunto com a flexibilidade e adaptação dos recursos educacionais na forma de OAs. O Agente Tutor BDI é peça fundamental na arquitetura Smart ITS por sua capacidade de representar crenças em *Propositional MetateM Logic* (PML). Este mesmo agente possui diferentes crenças temporais conforme a semântica do contexto das fontes por ele percebidas, ou seja, das informações provindas dos artefatos. Ele faz uso de PML para o raciocínio sobre planos dinâmicos e tem como finalidade a criação de cursos de ação.

Em BDI, os conjuntos de crenças, desejos e intenções lidam com a base de crenças e manipulações destas crenças por meio de funções. Crenças que se encontram na forma de fatos podem ser manipuladas para criar, através de uma função geradora de opções, um conjunto de mundos-possíveis com o nome de *desejos*. No AgentSpeak, visto na Seção 3.3.1, sempre que um agente se compromete com uma intenção e esta é consistente no interpretador, um plano proveniente da biblioteca de planos é selecionado para que o agente aja conforme a sequência de instruções em relação àquela intenção.

Os operadores da Lógica Temporal utilizados aqui são mapeados para representação em predicados e são apresentados com uma breve descrição de suas respectivas semânticas genéricas no Quadro 4. Estes operadores são baseados na PML e introduzidos no Quadro 2 da Seção 2.5.1.2. As semânticas dos operadores usadas nos contextos temporais desta dissertação serão melhor descritos nas subseções 4.4.2, 4.4.3 e 4.4.4. Elas permitem que o Agente Tutor BDI planeje, mantendo o ciclo de raciocínio BDI, como descrito na Seção 4.5, a partir de suas crenças.

Quadro 4 – Operadores Temporais dos Tempos Verbais Passado e Futuro

Predicado	Símbolo	Semântica genérica ¹
<i>Passado</i>		
$\text{past}(\varphi)$	$\blacksquare\varphi$	φ é sempre verdade no passado
$\text{precon}(\varphi)$	$\blacklozenge\varphi$	φ é verdade em algum momento do passado
$\text{back}(\varphi)$	$\bullet\varphi$	φ é verdade no momento imediatamente anterior
<i>Operador Binário de Passado</i>		
$\text{since}(\varphi, \psi)$	$\varphi \mathcal{S} \psi$	φ é verdade se ψ aconteceu ($\blacklozenge\psi$), então $\blacklozenge\varphi$
<i>Futuro</i>		
$\text{next}(\varphi)$	$\circ\varphi$	φ é verdade no próximo momento no tempo
$\text{postcon}(\varphi)$	$\blacklozenge\varphi$	φ é verdade agora ou em um futuro momento
$\text{future}(\varphi)$	$\square\varphi$	φ é verdade agora e para sempre no futuro
<i>Operador Binário de Futuro</i>		
$\text{until}(\varphi, \psi)$	$\varphi \mathcal{U} \psi$	φ é verdade até ψ ser verdade ($\blacklozenge\psi$)

¹ Por simplificação, uma proposição é dita interpretada como “verdadeira” quando é atualmente satisfeita ao referenciar outro tempo.

O uso destes operadores ocorre em semânticas dadas pelos seguintes conjuntos temporais: o conjunto Passado contém crenças sobre o estado do mundo por meio de um histórico; o conjunto Ação é constituído de proposições representando os recursos que afetam o mundo; e o conjunto Plano estrutura os conceitos na forma de sequência (como uma receita) de ações. Não obstante, estes conjuntos são alterações das definições dos conjuntos de crenças, desejos e intenções que carregam o nome de BDI, porém representadas nas crenças do agente e levando em consideração aspectos temporais. Estes aspectos temporais servem de parâmetro para a seleção de planos, customizáveis de acordo com as intenções do programador e que, já no início da execução, encontram-se na biblioteca de planos do AgentSpeak.

O plano BDI do Agente Tutor para a elaboração do *currículo* passa por um processo de simulação, no qual ocorre a comparação por meio do Modelo de Sobreposição. A simulação considera os predicados do Domínio de Conhecimento e compara eventuais estados futuros de interação do Aprendiz com OAs. Quando ela é finalizada, as instruções para a sequência de OAs está pronta para retornar o produto final: o Artefato de *currículo*. Na simulação, o Agente Tutor necessita manipular os seguintes conjuntos de crenças: Conjunto Sobreposição (CS); Conjunto Diferença (CD); e o *currículo*.

O Conjunto Sobreposição (CS) é a intersecção do Domínio de Aprendiz (DA), aquilo que retorna do registro de atividades do Aprendiz (Artefato de Aprendiz), com o Domínio de Conhecimento (DC) do curso (Artefato de

Domínio) no qual o Agente Tutor é especialista ($DA \cap DC$), ou seja, aquelas atividades que satisfazem proposições dentro do curso e que o Aprendiz já tem conhecimento. Após a criação do CS, o Agente Tutor compara as proposições do Conjunto Sobreposição para o planejamento iniciar com as crenças do DC, formando o Conjunto Diferença (CD), ou seja, aquilo que o Aprendiz ainda não viu no curso ($CD = DC - CS$). A Figura 6 demonstra visualmente estas relações. A partir disto, inicia-se a simulação considerando as proposições do CD para a elaboração do *curriculum* em uma hierarquia de árvore, considerando a temporalidade de ações.

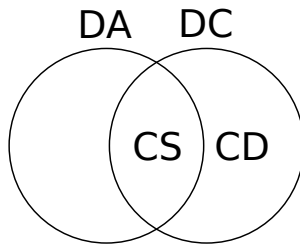


Figura 6 – Conjuntos de Domínio de Aprendiz e Conhecimento

O Conjunto Sobreposição é a fonte para saber quais proposições realmente já foram satisfeitas e o *curriculum* – que inicia a simulação vazio e vai sendo preenchido de ações futuras do Conjunto Diferença, onde este vai sendo esvaziado –, se torna a fonte para saber quais proposições se espera futuramente estarem satisfeitas em um dado momento/estado de acordo com a simulação. Portanto, o Conjunto Sobreposição e o *curriculum* servem de parâmetro para pré-condições, que são os gatilhos para saber quais atividades futuras devem ser satisfeitas.

4.3 ESTRUTURA DO DOMÍNIO DE CONHECIMENTO

Esta seção descreve como o Agente Tutor utiliza uma estrutura bem definida de curso em consonância com os operadores de Lógica Temporal por meio de BDI. A semântica para os operadores temporais depende do contexto do conjunto no qual se encontram. Como os projetistas de recursos atualmente seguem padrões de metadados, para a criação de recursos esta proposta exige poucas alterações nas anotações, pois estas podem ser mapeadas para o uso do Agente Tutor a partir do contexto temporal em que se encontram. Isto melhora a adaptação individual ao Aprendiz sem perder o controle sobre o Domínio de Conhecimento.

Por exemplo, os OAs, são anotados e representados no Agente Tutor como proposições em um formato que viabiliza o sequenciamento e navegação com o uso de pré e pós-condições. Assim as proposições sobre OAs podem ser usados de parâmetro para decisão das atividades a serem tomadas para que o Domínio de Aprendiz se aproxime do Domínio de Conhecimento.

O exemplo utilizado nesta dissertação corresponde a um curso com conceitos e exercícios sobre “Métodos Algorítmicos” e segue o modelo conceitual do Domínio de Conhecimento MATHEMA, em sua *visão externa*, contendo contextos, profundidades e lateralidades, conforme Quadro 5.

Quadro 5 – Modelo Conceitual do Curso “Métodos Algorítmicos”

Índices	Rótulos
<i>Contextos</i>	
c_1	Iteração
c_2	Recursão
<i>Profundidades</i>	
c_1p_1	Laços de repetição
c_2p_1	Funções recursivas
c_1p_2	Algoritmos iterativos
c_2p_2	Algoritmos recursivos
c_1p_3	Armazenamento iterativo
c_2p_3	Armazenamento recursivo
<i>Lateralidades</i>	
$c_1p_1l_1$	Aula - Parâmetros condicionais
$c_2p_1l_1$	Aula - Chamadas de função
$c_1p_2l_1$	Implementação - Fatorial iterativo
$c_2p_2l_1$	Implementação - Fatorial recursivo
$c_1p_2l_2$	Trabalho - Fibonacci iterativo
$c_2p_2l_2$	Trabalho - Fibonacci recursivo
$c_1p_3l_1$	Teste - Lista iterativa
$c_2p_3l_1$	Teste - Pilha recursiva

Como já mencionado, o Agente Tutor representa suas crenças na forma de proposições e agrega operadores lógicos temporais, que servem de predicado, dependendo de qual tipo de artefato acessa para retirar informações. As informações sobre a estrutura de domínio são retiradas do Artefato de Domínio e são necessárias para o plano de elaboração do *currículo*.

4.4 ESTRUTURA DE PLANO

O modelo temporal de planos converte a estrutura do domínio para planos BDI e, para isso, contém uma semântica que mapeia proposições em *fbf* sobre números naturais para estabelecer as relações de acessibilidade temporal à outras proposições, conforme visto na Seção 2.5.1.3. A execução em Lógica Temporal segue um conjunto de fórmulas em um modelo de temporalidade linear e discreta no sistema. Esta sequência com índices naturais retorna valores booleanos quando as proposições são acessadas de acordo com a Equação 2.8, ou seja, $\langle \mathcal{M}, i \rangle \models \varphi$, onde \mathcal{M} é um modelo, com um índice temporal i , e uma dada proposição φ sendo interpretada como verdade.

A partir disto é possível criar níveis hierárquicos, que podem ser chamadas de granularidade das proposições ou dimensões no MATHEMA, sobre a mesma estrutura, novamente representada atômicamente por uma outra proposição, ψ . A organização destes níveis hierárquicos na estrutura do modelo conceitual MATHEMA é mostrada na Figura 7. Sendo um OA o nível mais baixo de granularidade. A cada lateralidade, que é a proposição da atividade com OA, na estrutura \mathcal{M} são associadas as dimensões da *visão externa* do MATHEMA, ou seja, Contextos (*C*), Profundidades (*P*) e Lateralidades (*L*). Portanto, para um OA existe um contexto, uma profundidade e uma lateralidade aninhados no Domínio de Conhecimento do sistema tutor.

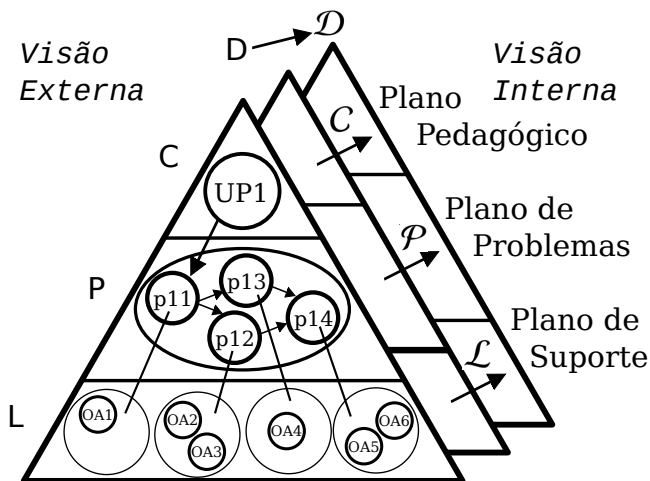


Figura 7 – Modelo Conceitual MATHEMA Usando OAs
Adaptado de (FRIGO et al., 2007).

Para o Agente Tutor é necessário constatar a satisfação das proposições atômicas para composição de plano. A partir disto, três dimensões da *visão externa* do MATHEMA (contexto, profundidade e lateralidade) são criadas para o planejamento e provêm ramificação numa estrutura CTL:

$$\begin{aligned} \langle D, n \rangle &\models c_i, \\ \langle c_i, j \rangle &\models p_j, \\ \langle p_j, t \rangle &\models l_t, \end{aligned} \tag{4.1}$$

onde o Domínio de Conhecimento (D) do curso tem acessibilidade aos seus contextos, cada contexto às suas profundidades, e cada profundidade às suas lateralidades. Portanto, uma proposição de lateralidade é interpretada válida em $D_n c_i p_j l_t$ pertencendo a um determinado Domínio de Conhecimento e às dimensões específicas na estrutura do modelo conceitual MATHEMA. Esta lateralidade é representada na sua forma geral e por um exemplo de fato conforme o Quadro 5 na base de crenças como:

lateralidade($n, i, j, t, \text{"nome da lateralidade"}$).
lateralidade(1,1,1,1,"Aula - Parâmetros condicionais").

onde n é o índice do Domínio de Conhecimento, i é o índice do contexto, j é o índice da profundidade, e t é o índice da lateralidade, sendo este seguido de uma cadeia de caracteres para nomear a lateralidade.

4.4.1 Crenças do Agente Tutor Sobre o Domínio de Conhecimento

O Agente Tutor BDI criado no Jason (BORDINI; HÜBNER; WOOLDRIDGE, 2007), descobre os tópicos do Domínio de Conhecimento na estrutura MATHEMA a partir dos dados acessados chamando por uma função do Artefato de Domínio, uma classe Java estendendo a classe *Artifact* do CArTAgO (RICCI; VIROLI; OMICINI, 2006), para então convertê-los para crenças. Esta leitura e conversão segue os modelos de plano vistos na Equação 4.1. Deixando \mathcal{D} ser o Domínio de Conhecimento, no qual o STI tem um curso e é descrito no Artefato de Domínio; considere que \mathcal{C} é o conjunto total de contextos dentro deste domínio \mathcal{D} ; considere \mathcal{P} como o conjunto total de profundidades do conjunto total de contextos \mathcal{C} ; e considere que \mathcal{L} é o conjunto total de lateralidades dos pares contexto/profundidade.

Separando \mathcal{D} em um número n de pontos de vista, ou seja, diferentes contextos, temos que n é o número total de elementos encontrados no conjunto de contextos \mathcal{C} . Porque \mathcal{D} é a soma de todos os contextos dentro do domínio, como já visto, e tendo em vista que consideramos apenas um do-

mínio, este tem o mesmo tamanho de C . Assim sendo, o tamanho de \mathcal{D} , $|\mathcal{D}|$, possui a mesma cardinalidade que o número total de contextos $|C|$. Um elemento em \mathcal{D} tem a seguinte configuração:

$$D = \langle C_i, P_i, L_i \rangle \quad (4.2)$$

onde D é uma *visão externa*, conforme Figura 7, e

$$\mathcal{D} = \{D_1, D_2, \dots, D_n\} \quad (4.3)$$

onde \mathcal{D} é o conjunto de visões externas D . Tomando isto em conta e associando ao domínio cada contexto c do conjunto de contextos com um índice i , iniciando em 1, tem-se:

$$C = \bigcup_{i=1}^{|\mathcal{D}|} C_i \quad (4.4)$$

onde estes são todos os elementos do conjunto de contextos ($\forall c \in C$) e contidos no domínio D . Agora, fazendo a associação de cada elemento profundidade p do conjunto de profundidades com um índice i , iniciando em 1, aos contextos, tem-se:

$$\mathcal{P} = \bigcup_{i=1}^{|\mathcal{D}|} P_i \quad (4.5)$$

onde $P_i = \{pi_1, pi_2, \dots, pik_i\}$ e no qual k_i varia para cada *visão externa* i .

Desta vez todas as profundidades p do conjunto de profundidades P não necessariamente estão contidas em um contexto c_i . Isto apenas aconteceria se o tamanho do conjunto de contextos fosse de apenas um contexto $|C| = 1$ e o contexto c_i conteria o conjunto de profundidades ($\forall p \in P = c_i$). Havendo mais de um contexto, apenas assegura-se que cada profundidade em um contexto $c_i p_j$ é uma profundidade de índice único p_j , contida em um determinado contexto c_i ($\forall p \in P \cap c_i$).

Finalmente, na associação das lateralidades l com índice t , iniciando em 1, para cada par contexto/profundidade, temos:

$$\mathcal{L} = \bigcup_{i=1}^{|\mathcal{D}|} \bigcup_{j=1}^{|\mathcal{P}_i|} L_{ij} \quad (4.6)$$

onde cada $L_{ij} = \{lij_1, lij_2, \dots, lij_{m_j}\}$ e cada m_j varia para cada problema pi_j .

Assim, D encontra-se em três dimensões: C , P e L ($\mathcal{D} = C \supset P \supset \mathcal{L}$). É por meio deste tipo de consulta ao Artefato de Domínio que são criadas as

crenças sobre o Domínio de Conhecimento.

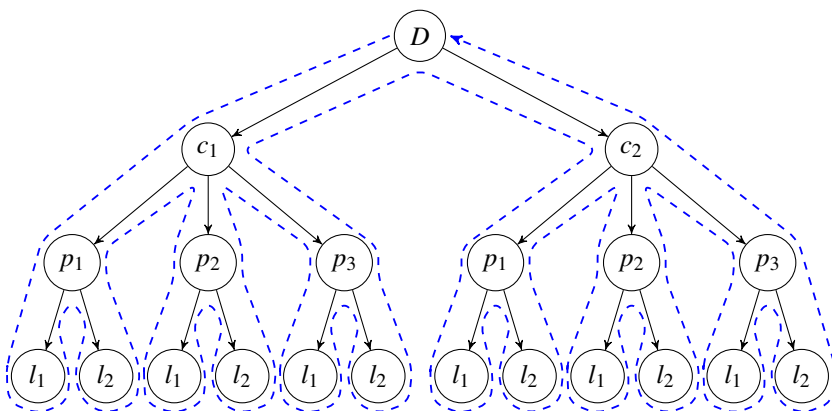
A granularidade das proposições é muito importante no Conjunto Diferença, conforme citado na 4.2, porque a tomada de decisão sobre um plano ocorre em relação ao estado corrente do mundo ao nível em que Aprendiz se encontra no curso, pela sua comparação com as dimensões do modelo conceitual MATHEMA.

Como já visto, a Figura 7 exibe o modelo temporal e semânticas propostas para o sequenciamento de OAs nas visões do MATHEMA. Esta estrutura para o modelo de planejamento temporal fornece uma base modular sobre a criação do *curriculum*. Ela descreve a sequência do plano em uma estrutura temporal linear voltada para o futuro, PTL, que permite a criação de um Conjunto Diferença e seleção de um curso de ações por meio de simulação dos próximos passos. Na implementação é empregada uma estrutura temporal que segue o padrão de especificação de sequenciamento simples das folhas da árvore, equivalentes a lateralidades/OAs. Como mostrado na Figura 8, usa-se um algoritmo de busca em profundidade. Para percorrer a árvore contendo as dimensões do modelo conceitual MATHEMA na simulação. Com o Conjunto Diferença já criado, o planejamento no Agente Tutor sobre o *curriculum* inicia no nó raiz:

1. visita o primeiro nó (o de menor índice) do Conjunto Diferença na dimensão em que se encontra (que por estar no Conjunto Diferença ainda não está validado/satisfeito) e se não for um nó folha (uma lateralidade), desce para a dimensão abaixo, repetindo (1);
2. sendo o primeiro nó folha (a lateralidade de menor índice e que por estar no Conjunto Diferença ainda não está satisfeita), adiciona uma ação ao plano (um OA para a sequência do *curriculum*) que satisfaça a lateralidade, e a marca como satisfeita;
3. volta uma dimensão acima no nó percorrido da árvore, verificando a validade do nó perante os subníveis e:
 - se o nó estiver satisfeito/válido na simulação, remove do Conjunto Diferença e:
 - se estiver no nó raiz, para;
 - senão repete (3).
 - senão repete a busca com (1), do nó em que parou.

Portanto, percorre contextos, profundidades e lateralidades para validar contextos e profundidades, e satisfazer todas as lateralidades com OAs que integram a sequência PTL do *curriculum*.

Figura 8 – Algoritmo de Profundidade para o Sequenciamento Simples



Semelhante à árvore de atividades do IMS e do SCORM que seguem sequenciamento simples em seu arquivo “manifest.xml” e a declaração é feita por uma árvore de atividades hierárquica com os níveis de curso, módulo e lição. Uma atividade neste caso, é descrita como um OA, que pode estar em padrão LOM, SCO, ou atividade como no Tin Can.

É importante ressaltar que apenas as folhas são sequenciadas no *curriculum* e removidas do Conjunto Diferença, pois compreendem lateralidades a serem satisfeitas, que, por sua vez, dizem respeito às atividades utilizando OAs. Os outros nós dizem respeito às granularidades dadas pelas dimensões de contexto e profundidade, e podem ser validadas para que seus nós não sejam mais percorridos. Neste caso, o nível é removido do Conjunto Diferença após a inserção do OA no *curriculum*. A verificação de pré e pós-condições também compara atividades simuladas no *curriculum*, e não apenas as atividades que o Aprendiz realmente realizou. Estas constam nos registros do Artefato de Aprendiz e são as proposições do Conjunto Diferença real, sem a simulação do plano de *curriculum*. No próximo ciclo de inferências do Agente Tutor a simulação inicia novamente com o Conjunto Diferença no estado atual do Aprendiz, onde a dinamicidade temporal faça o Agente Tutor produzir um novo *curriculum* de acordo com as necessidades dele.

Os OAs podem satisfazer mais de uma proposição, não precisando ser exclusivamente a próxima lateralidade, pois pode satisfazer mais lateralidades ou até validar níveis (profundidades e contextos). Em um sistema com mais cursos e estruturas parecidas, esta validação retorna registros do que o Artefato de Aprendiz já aprendeu para que algo novo seja apresentado a ele.

4.4.2 Semântica de Passado

Inferida a partir das sentenças do Artefato do Aprendiz, o Agente Tutor considera as informações acessadas como registros que condizem com as atividades passadas do Aprendiz e, portanto, são convertidas para crenças lógicas na Semântica de Passado. O acesso ao Artefato do Aprendiz acontece ao chamar a função `getAtividade`:

```
getAtividade(Indice,X,Y,C,P,L)[artifact_id(ArtAprendiz)];
```

As crenças que retornam através do valor das variáveis desta função são utilizadas no ciclo de deliberação do Agente Tutor, servindo de pré-condições para ações nas quais é feita a distinção entre o que fora verdade em algum momento no curso ($\diamond\varphi$) com o que sempre fora verdade no passado, mas é uma crença fraca de satisfação, pelo fato da proposição não ter estar dentre as atividades diretamente relacionadas com o curso atual ($\blacksquare\varphi$). Uma crença de passado é adicionada seguindo a forma geral e comentada:

```
/* Aprendiz onde aprendiz(id,predicado,domínio,
 * dimensão,contexto,profundidade,lateralidade) */
+aprendiz(X,"precon",Y,"lateralidade",C,P,L).
/* Conjunto Diferença onde cd(aprendiz,domínio,
 * dimensão,contexto,profundidade,lateralidade) */
+cd(X,Dominio,"lateralidade",C,P,L).
```

e seguindo o exemplo do Quadro 5, as crenças podem ser populadas conforme as informações do Artefato de Aprendiz e adicionadas à base de crença do Agente Tutor:

```
+aprendiz("ana","precon",1,"lateralidade",1,1,1).
+aprendiz("ana","past",4,"lateralidade",1,1,2);
+cd(ana,1,"lateralidade",1,1,1).
```

onde o Aprendiz realizou uma lateralidade no curso atual, de domínio 1, e uma outra lateralidade em um curso diferente, de domínio 4.

4.4.3 Semântica de Ação

A semântica de uma ação é inferida a partir do Artefato de OAs. Uma ação representa um desejo a ser realizado por uma atividade com um OA. Desejos são crenças do que o agente quer com que o mundo se pareça e o melhor deles (realizável e deixando mais próximo do ideal, neste caso, o

Domínio de Conhecimento) é selecionado como intenção. Ações tem uma semântica de passado e futuro porque uma ação deve ser comparada com o domínio do aprendiz (passado), em pré-condição ($\diamond\varphi$), e o Domínio de Conhecimento (futuro) para tomar uma atitude no mundo, em pós-condição ($\heartsuit\varphi$). A adição de uma crença de ação segue a forma geral:

```
/* OA onde oa(id_oa,predicado,domínio,
 * dimensão,contexto,profundidade,lateralidade)*/
+oa(OA,"precon",X,"lateralidade",C,P,L).
```

e é populada conforme:

```
+oa(1,"postcon",1,"lateralidade",1,1,1).
+oa(2,"precon",1,"lateralidade",1,1,1).
+oa(2,"postcon",1,"lateralidade",1,2,2).
```

As ações também podem demonstrar sequências simples nas quais outras ações são requeridas (dependências). A anotação de dependência em OAs é semelhante à sequência elaborada do *curriculum*, onde existem momentos e o acesso deles ocorre com próxima ação ($\circ\varphi$) e/ou ação anterior ($\bullet\varphi$).

```
/* Dependência de OA onde
 * oa_depende(id_oa,predicado,id_oa)*/
+oa_depende(X,"prev",Y).
```

e a dependência de OA é populada conforme:

```
+oa_depende(3,"prev",1).
+oa_depende(3,"next",4).
+oa(3,"postcon",1,"profundidade",1,2).
```

4.4.4 Semântica de Plano

A semântica do plano é inferida a partir do Artefato de Domínio. Um plano neste caso relaciona o aprendiz com a estrutura do modelo conceitual de domínio. Esta estrutura possui ações futuro-dirigidas, mas considera-se o passado para a leitura do estado corrente de mundo. O nível de granularidade empregado em planos é maior do que nas ações — que são intencionalmente representadas como unidades de recurso atômicas na forma de OAs — e até por isso, consiste em uma sequência delas. Um plano elaborado para o aprendiz descreve atividades de estados futuros de acordo com o conjunto diferença: a parte do domínio que o aprendiz não tem atividades satisfeitas, mas que faz parte do domínio.

O plano faz uso de ações que configuram pré-condições ($\diamond\varphi$) e pós-condições ($\heartsuit\varphi$) no contexto de plano BDI. Os melhores OAs garantem que a proposição de pós-condição seja a próxima ação ($\circ\varphi$). Em cada nova atividade para os estados com OAs, estes são sequenciados como ações no plano e gravados no curriculum. Na implementação, a estrutura temporal segue um raciocínio para a elaboração de sequenciamento simples, no qual percorre-se a estrutura como em um algoritmo de busca em profundidade para o planejamento ($\square\varphi$). Para dar duração ao sequenciamento simples, regras temporais devem ser empregadas seguindo crenças utilizando o operador *until* ($\square\varphi\mathcal{U}\psi$) ou o operador *since* ($\square\varphi\mathcal{S}\psi$).

4.5 AGENTE TUTOR: SEQUENCIANDO OBJETOS DE APRENDIZAGEM UTILIZANDO OPERADORES TEMPORAIS

O ciclo de tutoria para o Agente Tutor segue cinco passos: (i), o Agente Tutor espera por informações que provenham do Agente de Interface referentes a sua responsabilidade para tutoria de novos aprendizes, para prover um curso com sequencia de OAs, e dos identificadores respectivos aos Artefato de Aprendiz, Artefato de Domínio e Artefato de OAs; (ii), o Agente Tutor consulta os artefatos e: se não há atualizações neles, repete (i) e (ii), de outra forma, atualiza suas crenças e converte os dados em fatos ao incluir os predicados temporais dependentes do contexto; (iii), o Agente Tutor manipula as suas crenças do o que sabe sobre Aprendiz, e compara crenças que mantém sobre ele em relação com o Domínio de Conhecimento para então criar o Conjunto Diferença do plano; (iv), o Agente Tutor compõe um plano para o *curriculum* de cada um dos Aprendizes ao escolher uma sequência de OAs, respeitando os contextos e as regras temporais estabelecidas do plano; (v), o Agente Tutor converte o plano que se encontra descrito em PTL, em proposições lógicas correspondentes para a criação do Artefato do *curriculum* para o uso do Agente de Interface.

Resumindo, conforme a Figura 5 da Seção 4.1, o ciclo completo de deliberação do Agente Tutor completa os passos (i) e (ii) em resposta às mudanças que ocorrem no ambiente, representados pelo ícone de “Percepção”; realiza os passos (iii) e (iv) para o planejamento da sequência do curso representados pelo ícone “Ação”; e realiza o passo (v) para a criação do Artefato do *curriculum* a ser manipulado pelo Agente de Interface, representado pelo ícone “Dados”.

4.5.1 Exemplo

Nesta seção, ilustramos o processo de deliberação de um Agente Tutor, nomeado de Bob, em instâncias de curso. O principal objetivo do Agente Tutor, como já evidenciado anteriormente, é a criação de uma sequência de atividades representadas na forma de predicados e que tomam forma no mundo real a partir de OAs. Ele atinge seus objetivos no ciclo deliberativo por ter planos prontos para exercer sua atividade em função do que percebe em aspectos temporais: a Semântica de Passado, representada pelo Artefato de Aprendiz; a Semântica de Plano, representada pelo Artefato de Domínio; e a Semântica de Ação, representada pelo Artefato de OAs.

A implementação fez uso do Jacamo e segue a perspectiva das crenças do Agente Tutor em relação aos artefatos, suas transformações por meio de planos e a criação de curricula a partir do modelo conceitual MATHEMA do Quadro 5.

4.5.1.1 Um Curso Simples

Neste exemplo, o Agente Tutor Bob é responsável pela tutoria da Agente Aprendiz aqui chamada de Ana em um curso de “Métodos Algorítmicos” com temporalidade de sequenciamento simples, para o Artefato de Domínio. Na primeira interação com o sistema, Ana se registra no curso e Agente de Interface armazena os dados fornecidos no Artefato de Aprendiz. Agente de Interface informa (ou cria um novo agente caso ainda não haja) ao Agente Tutor Bob que ele se tornou responsável pela tutoria de Ana. Bob converte as informações recebidas do Agente de Interface, a fonte de informações, em crenças próprias:

```
/* Artefato de Domínio onde ad(curso,artefato)*/
+ad("Métodos Algorítmicos",cobj_3).
/* Artefato do Aprendiz onde aa(aprendiz,artefato)*/
+aa(ana,cobj_6).
/* Responsabilidade sobre o Aprendiz
* onde ra(aprendiz,curso)*/
+ra(ana,"Métodos Algorítmicos").
/* Artefato de Objetos de Aprendizagem
* onde aoa(aprendiz,curso,artefato)*/
+aoa(ana,"Métodos Algorítmicos",cobj_7).
```

Agora, Bob tem referência o suficiente para consultar informações sobre domínio, aluno e OAs.

Inicialmente, Bob deve constatar se houveram atualizações nas suas crenças sobre o Domínio de Conhecimento em que é especialista. Retornando ao modelo conceitual do curso de “Métodos Algorítmicos”, apresentado na Seção 4.3, para ter acesso às informações do curso nesta estrutura, Bob consulta o Artefato de Domínio para adicionar novas crenças. Este Artefato de Domínio, ao qual o Agente Tutor faz acesso no ETEC, é um objeto Java da classe DomainArtifact, estendendo a classe Artefato da biblioteca do CArtAgO e permite o retorno de dados por meio de operações.

```

/* Cursos onde curso(id,nome)*/
+dominio(1,"Métodos Algorítmicos").
/* Contextos onde contexto(domínio,id,nome)*/
+contexto(1,1,"Iteração").
+contexto(1,2,"Recursão").
/* Profundidades onde
* profundidade(domínio,contexto,id,nome)*/
+profundidade(1,1,1,"Laços de repetição").
+profundidade(1,2,1,"Recursão em funções").
+profundidade(1,1,2,"Algoritmos iterativos").
+profundidade(1,2,2,"Algoritmos recursivos").
+profundidade(1,1,3,"Armazenamento iterativo").
+profundidade(1,2,3,"Armazenamento recursivo").
/* Lateralidades onde
* lateralidade(domínio,contexto,profundidade,id,nome)*/
+lateralidade(1,1,1,1,"Parâmetros condicionais").
+lateralidade(1,2,1,1,"Chamadas de funcao").
+lateralidade(1,1,2,1,"Fatorial iterativo").
+lateralidade(1,2,2,1,"Fatorial recursivo").
+lateralidade(1,1,2,2,"Fibonacci iterativo").
+lateralidade(1,2,2,2,"Fibonacci recursivo").
+lateralidade(1,1,3,1,"Lista iterativa").
+lateralidade(1,2,3,2,"Pilha recursiva").

```

As crenças do domínio devem ser comparadas com as lateralidades já satisfeitas pelo Aprendiz, que equivalem a OAs, com predicados de Lógica Temporal.

Bob sequencia uma navegação simples pelo contexto, profundidade e lateralidade de seus índices correspondentes.

Para evitar um caso inicial sem atividades sobre Ana no curso, considera-se que ela já tenha realizado atividades no sistema que satisfazem algumas atividades deste curso e portanto o Agente Tutor deve tomar conhecimento. Desta maneira, Ana já viu uma aula sobre o conteúdo de laços de repetição

($c_1p_1l_1$) e implementou um algoritmo iterativo de fatorial ($c_1p_2l_1$).

Bob observa cada um dos Artefato de Aprendiz de seus aprendizes procurando atualizações. Se não há alguma atualização, ele acessa os dados, que estão na forma de resultados de atividades realizadas pelo Aprendiz e os converte em fatos de lógica. Devido ao fato do Artefato de Aprendiz de Ana conter atualizações para com o Bob, este muda suas crenças sobre o estado atual de Ana ($\diamond c_1p_1l_1$ e $\diamond c_1p_2l_1$):

```
/* Aprendiz onde aprendiz(id,predicado,domínio,
 * dimensão,contexto,profundidade,lateralidade) */
+aprendiz("ana", "precon", 1, "lateralidade", 1, 1, 1).
+aprendiz("ana", "precon", 1, "lateralidade", 1, 2, 1).
```

Após isto, Bob pode começar o Modelo de Sobreposição para configurar os conjuntos Conjunto Sobreposição, Conjunto Diferença e o *curriculum*. O Conjunto Diferença (CD) é iniciado recebendo todos contextos, profundidades e lateralidades do domínio, exceto tudo o que existe como crença de que Ana já realizou, o Conjunto Sobreposição dado por todas as crenças de pré-condição (precon):

```
/* Conjunto Diferenca onde cd(aprendiz,domínio,
 * dimensao,contexto,profundidade,lateralidade) */
+cd(ana, 1, "lateralidade", C,P,L).
```

onde o Conjunto Sobreposição é um conjunto de crenças do Agente Tutor que corresponde às lateralidades que são atividades no Domínio de Aprendiz (DA) e já foram realizadas no Domínio de Conhecimento (DC), e o Conjunto Diferença é o que Ana não viu dentro do Domínio de Conhecimento ($CD = DC - CS$). Como já visto na Figura 6 da Seção 4.2. Portanto, nenhuma das lateralidades do curso que Ana já realizou com êxito se encontrarão no Conjunto Diferença inferido por Bob.

Por causa das granularidades, o plano verifica recursivamente a satisfazibilidade dos ramos da árvore de estados nos níveis superiores, conforme a árvore da Figura 8 da Seção 4.4.1. Isto fará com que o Bob adicione ao Conjunto Diferença todas as profundidades, exceto àquelas com lateralidades validando a profundidade, e o mesmo acontece com todos os contextos em relação às profundidades:

```
+cd(ana, 1, "profundidade", C,P).
+cd(ana, 1, "contexto", C).
```

Assim, Bob tem o seu Conjunto Diferença atualizado e no caso deste exemplo, Ana está mais próxima do Domínio de Conhecimento do que um

Aprendiz sem registros de atividades no curso, porque ela já satisfaz algumas proposições do domínio.

Agora, Bob ainda não pode começar a sequenciar OAs e elaborar o *curriculum* descrevendo atividades específicas para a Ana. Para seguir o plano de curso na estrutura do modelo conceitual MATHEMA ele deve primeiro ter informações sobre os OAs que satisfazem as proposições. Ele acessa o Artefato de OAs e adiciona as seguintes crenças:

```
/* Repositório onde
* repositório(id,domínio,nome,endereço) */
+repositorio(1,1,"localhost","http://127.0.0.1").
/* Identificador do OA onde id_oa(id,repositorio)*/
+id_oa(1,1).
/* OA onde oa(id_oa,predicado,domínio,
* dimensão,contexto,profundidade,lateralidade)*/
+oa(1,"precon",1,"profundidade",1,1).
+oa(1,"precon",1,"lateralidade",1,2,1).
+oa(1,"postcon",1,"lateralidade",1,2,2).
+id_oa(2,1).
+oa(2,"precon",1,"profundidade",1,2).
+oa(2,"postcon",1,"lateralidade",1,3,1).
...
```

A próxima lateralidade, definida pelo contexto, no Conjunto Diferença é a crença futura de implementar um algoritmo Fibonacci é descrita como uma pós-condição ($\diamond_{c_1 p_2 l_2}$). Existe um OA corresponde com a pós-condição, mas precisa que pré-condições de Ana sejam satisfeitas ($\diamond_{c_1 p_1}$ and $\diamond_{c_1 p_2 l_1}$). Bob consulta as suas crenças e Ana satisfaz essas pré-condições, fazendo com que o OA esteja apto para Ana no próximo momento no futuro.

Caso houverem dois ou mais OAs aptos para a realização da atividade, o contexto do plano de Bob seleciona o OA com melhores atributos que casem com o Modelo de Aprendiz. Caso não hajam diferenciais significativos, o primeiro OA apto é selecionado.

Bob então seleciona o OA de id 1 para a próxima ação e, portanto, adiciona este para que seja usado na próxima atividade do *curriculum* ($\circ_{c_1 p_2 l_2}$). Consequentemente, também retira do Conjunto Diferença esta lateralidade que é dada como satisfeita, além da profundidade, que é dada como válida:

```
/* curriculum onde curriculum(id,aprendiz,domínio,
* id_oa,dimensão,contexto,profundidade,lateralidade) */
+curriculum(1,ana,1,1,"lateralidade",1,2,2).
-cd(ana,1,"lateralidade",1,2,2).
-cd(ana,1,"profundidade",1,2).
```

Bob supõe em sua simulação, considerando a atividade do curriculum já adicionada, que Ana futuramente complete a atividade com a implementação do algoritmo de Fibonacci ($\diamond c_1 p_2 l_2$) e supõe que a próxima atividade dela pode ser prestar um teste sobre listas.

Bob supõe que Ana irá passar o teste ($\diamond c_1 p_3 l_1$) e assim o fazendo completa as três profundidades no contexto de iteração. Isto é, o primeiro contexto é um objetivo alcançado ($\diamond c_1$) na simulação e por isso é incluído como um fato no plano do *curriculum*. Consequentemente, o mesmo contexto é retirado das crenças do Conjunto Diferença na simulação de atividades com OAs que suprem as proposições do plano, de acordo com as crenças que Bob possui em relação à Ana.

```
+curriculum(2, ana, 1, 2, "lateralidade", 1, 3, 1).
-cd(ana, 1, "lateralidade", 1, 3, 1).
-cd(ana, 1, "contexto", 1).
```

Para que Ana possa então entrar no segundo contexto (c_2), Bob busca pela primeira profundidade dela que ainda não está validada ($c_2 p_1$) e a primeira lateralidade desta não satisfeita ($c_2 p_1 l_1$), retornando a próxima ação.

Assim, a intenção de Bob para a próxima lição será uma atividade envolvendo chamadas de função ($\circ c_2 p_1 l_1$). Nesta atividade Bob encontra um OA que satisfaz a proposição, insere a crença no *curriculum*, remove a crença do Conjunto Diferença, e assim por diante. Este planejamento continua até a conclusão do *curriculum* produzido pelo Agente Tutor Bob, no domínio do curso de Métodos Algorítmicos, especialmente para a Aprendiz Ana.

5 CONCLUSÃO

“Nasci para satisfazer a grande necessidade que eu tinha de mim mesmo.”

(Jean-Paul Sartre)

Esta dissertação analisou trabalhos correlatos nas áreas de Sistemas Tutores Inteligentes (STI) e Sistemas Hiperfídia Adaptativos Educacionais (SHAE), e mais especificamente em sequenciamento de OAs e lógicas modais considerando a modalidade temporal em agentes. A dissertação integrou um modelo de domínio consistente com o uso Objetos de Aprendizagem (OAs) por meio de Lógica Temporal para a produção de *curricula* consistindo na sequência de OAs e provendo personalização e adaptação ao Aprendiz.

Para isto, a proposta apresenta uma arquitetura de STI multiagente, denominada Smart ITS, na qual artefatos e agentes de *software* viabilizam um ambiente de tutoria que considera aspectos temporais como um fator adicional para a adaptação e personalização para o Aprendiz no curso. Estes aspectos são relevantes no diagnóstico do Aprendiz, por Modelo de Sobreposição, em relação ao Domínio de Conhecimento e posterior sequenciamento das atividades com OAs.

O principal componente da arquitetura Smart ITS é o Agente Tutor. Devido a sua capacidade de inferir o contexto temporal de dados dos artefatos que possui acesso, ele efetua o sequenciamento de OAs para um dado Agente Aprendiz gerando um *curriculum*. Isto se deve à representação de crenças no Agente Tutor por meio de Lógica Temporal e suas semânticas de passado e futuro, adaptadas da linguagem temporal PML para uma representação em AgentSpeak. A representação do domínio do curso levou em consideração a estrutura multidimensional do modelo conceitual MATHEMA para a representação do curso. O Agente Tutor integra o modelo conceitual MATHEMA e faz uso de Lógica Temporal por ciclo de raciocínio BDI. Os planos do Agente Tutor focam no sequenciamento de OAs, onde a Lógica Temporal torna possível a elaboração de *curricula* de forma dinâmica, com o objetivo de atender às necessidades educacionais do Aprendiz.

A implementação do sistema multiagente Smart ITS foi desenvolvida em JaCaMo e conteve os elementos da arquitetura proposta com a finalidade de elaboração de *curricula*. O exemplo apresentado demonstrou o ciclo de raciocínio BDI no Agente Tutor: como o Agente Tutor representa as suas crenças sobre os artefatos de Domínio de Conhecimento, de Aprendiz e de OA em Lógica Temporal; e como ele elabora os *curricula* em uma sequência linear para interações pedagógicas futuras com o uso de OAs.

Assim sendo, nesta dissertação, a implementação do Smart ITS possibilitou demonstrar uma integração de características das áreas de SHAE e STI em uma arquitetura multiagente para a elaboração de um curso personalizado ao Aprendiz, fazendo uso da estrutura de domínio bem definida do modelo conceitual MATHEMA e da flexibilidade de sequenciamento de atividades, providas por recursos na forma de OAs sequenciados por meio de operadores de Lógica Temporal adaptados da PML.

5.1 TRABALHOS FUTUROS

Este trabalho focou em grande parte na inteligência artificial que envolve o raciocínio de um agente considerando predicados de Lógica Temporal. O modelo é genérico para declarações proposicionais que fazem uso de atividades com OAs, e que equivalem a ações no plano. Para trabalho futuro, almeja-se a integração desta arquitetura em um curso Moodle, fazendo uso de registros de atividade no padrão de especificação Tin Can.

Outra extensão à implementação contempla, como mostramos na Seção 4.4.4 com plano, a aplicação de um novo predicado de regra. Estas regras temporais serviriam para a satisfazibilidade de diferentes usos dos operadores da lógica para casos específicos de acordo com gradações dos resultados de atividades. Provendo reforço de atividade ($\diamond\varphi\mathcal{S}\psi$), validação de proposições para avançar o caminho de atividades ($\diamond\varphi\mathcal{U}\psi$), ou provimento de conteúdos específicos condicionados ao diagnóstico do Aprendiz.

REFERÊNCIAS

ABADI, M.; MANNA, Z. Temporal logic programming. **Journal of Symbolic Computation**, v. 8, n. 3, p. 277 – 295, 1989. ISSN 0747-7171.

ADL. **SCORM 2004 4th Edition Sequencing and navigation**. [S.l.]: Advanced Distributed Learning, 2009.

ADL. **Experience API**. [S.l.]: Advanced Distributed Learning, 2013.

ALMEIDA, H. O. de. **Compor-desenvolvimento de software para sistemas multiagentes**. Tese (Doutorado) — Universidade Federal de Campina Grande, 2004.

AMORIM JR., J.; SILVEIRA, R. Seleção dinâmica de objetos de aprendizagem baseados no padrão scorm no ambiente moodle. In: **Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação-SBIE)**. [S.l.: s.n.], 2015. v. 26, n. 1, p. 967.

ARCE-C’RDENAS, F.; GARCIA-VALDEZ, M. Learning objects for intelligent environments. In: **2012 Eighth International Conference on Intell. Env**. [S.l.: s.n.], 2012.

ASSUNÇÃO, F. M. **SHART-Web: Um Sistema Tutor em Harmonia Tradicional na Web**. Tese (Doutorado) — Universidade Federal da Paraíba, 2001.

BARRINGER, H. et al. Metatem: A framework for programming in temporal logic. In: SPRINGER. **Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness**. [S.l.], 1990. p. 94–129.

BARRINGER, H. et al. Metatem: An introduction. **Formal Aspects of Comp.**, Springer, v. 7, n. 5, p. 533–549, 1995.

BEN-ARI, M.; MANNA, Z.; PNUELI, A. The temporal logic of branching time. In: **Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages**. New York, NY, USA: ACM, 1981. (POPL ’81), p. 164–176. ISBN 0-89791-029-X.

BENJAMIN, L. T. A History of Teaching Machine. **American Psychological**, v. 43, n. 9, 1988.

BESOLD, T. R.; SCHIEMANN, B. A multi-context system computing modalities. In: **23rd International Workshop on Description Logics DL2010**. [S.l.: s.n.], 2010. p. 431.

BOISSIER, O. et al. Multi-agent oriented programming with jacamo. **Science of Computer Programming**, Elsevier, v. 78, n. 6, p. 747–761, 2013.

BOND, A. H.; GASSER, L. **A Survey of Distributed Artificial Intelligence**. Pasadena, CA, Agosto 1988.

BORDINI, R. H.; HÜBNER, J. F.; WOOLDRIDGE, M. **Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)**. [S.l.]: John Wiley & Sons, 2007. ISBN 0470029005.

BORGES, H. P. **Assistente de resolução de problemas para o sistema mathnet**. Dissertação (Mestrado) — PPGEE/UFMA, Maio 2002.

BRA, P. D.; HOUBEN, G.-J.; WU, H. Aham: a dexter-based reference model for adaptive hypermedia. In: ACM. **Proceedings of the tenth ACM Conference on Hypertext and hypermedia: returning to our diverse roots: returning to our diverse roots**. [S.l.], 1999. p. 147–156.

BRA, P. D.; SMITS, D.; STASH, N. Creating and delivering adaptive courses with aha! In: SPRINGER. **European Conference on Technology Enhanced Learning**. [S.l.], 2006. p. 21–33.

BRATMAN, M. **Intention, plans, and practical reason**. Cambridge, MA: Harvard University Press, 1987.

BROOKS, R. A. **Achieving artificial intelligence through building robots**. [S.l.], 1986.

BROWN, J. S.; BURTON, R. R. Sophie: A pragmatic use of artificial intelligence in cai. In: **Proceedings of the 1974 Annual ACM Conference - Volume 2**. New York, NY, USA: ACM, 1974. (ACM '74), p. 571–579.

BUTZ, C. J.; HUA, S.; MAGUIRE, R. B. A web-based bayesian intelligent tutoring system for computer programming. **Web Intelligence and Agent Systems: An International Journal**, IOS Press, v. 4, n. 1, p. 77–97, 2006.

CARBONELL, J. **AI in CAI: An Artificial-intelligence Approach to Computer-assisted Instruction**. [S.l.]: IEEE, 1970.

CARDOSO, J. et al. Mathtutor: a multi-agent intelligent tutoring system. In: **Artificial Intelligence Applications and Innovations**. [S.l.]: Springer, 2004. p. 231–242.

CHEN, C.-M. Ontology-based concept map for planning a personalised learning path. **British Journal of Educational Technology**, Wiley Online Library, v. 40, n. 6, p. 1028–1058, 2009.

CHOMSKY, N. Three models for the description of language. **IRE Transactions on Information Theory**, v. 2, n. 3, p. 113–124, 1956.

CLANCEY, W. J. The epistemology of a rule-based expert system - A framework for explanation. **Artif. Intell.**, v. 20, n. 3, p. 215–251, 1983.

CLARKE, E. M.; EMERSON, E. A. **Design and synthesis of synchronization skeletons using branching time temporal logic**. [S.l.]: Springer, 1982.

COHEN, P. R.; LEVESQUE, H. J. Intention is choice with commitment. **Artif. Intell.**, Elsevier Science Publishers Ltd., Essex, UK, v. 42, n. 2-3, p. 213–261, mar. 1990.

CORREIA, A. Distributed artificial intelligence. **SIBAG**, Porto, Portugal, p. 9, 2011.

COSTA, E. d. B.; LOPES, M. A.; FERNEDA, E. Mathema: A learning environment based on a multi-agent architecture. In: **Proceedings of the 12th Brazilian Symposium on Artificial Intelligence: Advances in Artificial Intelligence**. London, UK, UK: Springer-Verlag, 1995. (SBIA '95), p. 141–150. ISBN 3-540-60436-7.

COSTA, E. de B. **Um modelo de ambiente interativo de aprendizagem baseado numa arquitetura multi-agentes**. Tese (Doutorado) — UFPA, 1997.

COSTA, N. S. **Modelagem e Construção de uma Ferramenta de Autoria para um Sistema Tutorial Inteligente**. Dissertação (Mestrado) — PPGE/UFMA, Maio 2002.

DE-MARCOS, L. et al. A new sequencing method in web-based education. In: IEEE. **2009 IEEE Congress on Evolutionary Computation**. [S.l.], 2009. p. 3219–3225.

DENNETT, D. C. Intentional systems. **The Journal of Philosophy**, Philosophy Documentation Center, v. 68, n. 4, p. 87, fev 1971.

DHEEBAN, S. et al. Personalized e-course composition approach using digital pheromones in improved particle swarm optimization. In: IEEE. **2010 Sixth International Conference on Natural Computation**. [S.l.], 2010. v. 5, p. 2677–2681.

D'INVERNO, M. et al. A formal specification of dmars. In: _____. **INTELLIGENT AGENTS IV**. [S.l.]: Springer, 1998. p. 155–176. ISBN 3-540-64162-9.

ELECTRICAL, I. of; COMMITTEE, E. E. L. T. S. **IEEE Standard for Learning Object Metadata (Draft). IEEE Standard 1484.12.1**. New York: [s.n.], 2002.

EMERSON, E. A. Handbook of theoretical computer science (vol. b). In: LEEUWEN, J. van (Ed.). Cambridge, MA, USA: MIT Press, 1990. cap. Temporal and Modal Logic, p. 995–1072.

FARIA, T. d. F. et al. **Um ambiente interativo multiagentes para o ensino de estrutura da informação**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, Abril 2001.

FILHO, S. S. A. **Modelagem do Agente Estratégico para ambientes multiestratégicos de aprendizagem cooperativa computadorizada**. Dissertação (Mestrado) — PPGEE/UFMA, Abril 2002.

FISCHER, M. J.; LADNER, R. E. Propositional dynamic logic of regular programs. **Journal of computer and system sciences**, Elsevier, v. 18, n. 2, p. 194–211, 1979.

FISHER, M. A survey of concurrent metatem - the language and its applications. In: **Proceedings of the First International Conference on Temporal Logic**. London, UK, UK: Springer-Verlag, 1994. (ICTL '94), p. 480–505. ISBN 3-540-58241-X.

FISHER, M. Metatem: The story so far. In: SPRINGER. **International Workshop on Programming Multi-Agent Systems**. [S.l.], 2005. p. 3–22.

FISHER, M. **An Introduction to Practical Formal Methods Using Temporal Logic**. [S.l.]: Wiley, 2011. ISBN 978-0-470-02788-2.

FLEET, K. White Paper, **Learning Design and e-learning**. [S.l.]: Epic Learning Group, 2012.

FRIGO, L. B. et al. **Um ambiente interativo multiagente para o ensino de estrutura da informação**. Março 2002.

FRIGO, L. B. et al. **Um modelo de autoria para sistemas tutores adaptativos**. Tese (Doutorado) — UFSC, 2007.

GARRIDO, A.; MORALES, L.; SERINA, I. On the use of case-based planning for e-learning personalization. **Exp. Syst. with Appl.**, v. 60, p. 1 – 15, 2016. ISSN 0957-4174.

GARRIDO, A.; ONAINDIA, E. Assembling learning objects for personalized learning: an ai planning perspective. **IEEE Intelligent Systems**, IEEE, v. 28, n. 2, p. 64–73, 2013.

GAVIDIA, L. C. V. d. A. J. J. Z. Sistemas tutores inteligentes. **Trabalho de Conclusão da Disciplina de IA, Universidade Federal do Rio de Janeiro**, COPPE-UFRJ, p. 24, 2003.

GENESERETH, M. R.; NILSSON, N. J. **Logical foundations of artificial intelligence**. [S.l.]: Morgan Kaufmann, 1988. ISBN 978-0-934613-31-6.

GÓIS, G. M. **Um Sistema Tutor Multi-Agentes no Domínio de Redes de Petri**. Dissertação (Mestrado) — Universidade Federal da Paraíba, 2000.

GROUP, R. I. A. **Prometheus Design Tool (PDT)**. 2014. Disponível em: <<https://sites.google.com/site/rmitagents/software/prometheusPDT>>.

GRUBIŠIĆ, A.; STANKOV, S.; ŽITKO, B. Adaptive courseware: A literature review. **j-jucs**, v. 21, n. 9, p. 1168–1209, set 2015.

HENZE, N.; NEJDL, W. A logical characterization of adaptive educational hypermedia. **New review of hypermedia and multimedia**, Taylor & Francis, v. 10, n. 1, p. 77–113, 2004.

HEWITT, C.; JR., H. B. **Actors and Continuous Functionals**. [S.l.], 1977. MIT-LCS-TR-194.pdf.

HONG, C.-M. et al. Intelligent web-based tutoring system with personalized learning path guidance. In: IEEE. **Seventh IEEE International Conference on Advanced Learning Technologies (ICALT 2007)**. [S.l.], 2007. p. 512–516.

IMS. **IMS Simple Sequencing Best Practice and Implementation Guide**. [S.l.]: IMS Global Learning Consortium, 2003.

IMS. **IMS Simple Sequencing Information and Behavior Model**. [S.l.]: IMS Global Learning Consortium, 2003.

IMS. **IMS Simple Sequencing XML Binding**. [S.l.]: IMS Global Learning Consortium, 2003.

IMS Global Learning Consortium. **IMS Learning Design Specification**. 2003.

INGRAND, F. F.; GEORGEFF, M. P.; RAO, A. S. An architecture for Real-Time Reasoning and System Control. **IEEE Expert: Intelligent Systems and Their Applications**, IEEE Educational Activities Department, Piscataway, NJ, USA, v. 7, n. 6, p. 34–44, 1992.

ISHIDA, T. Parallel, distributed and multi-agent production systems - A research foundation for distributed artificial intelligence. In: LESSER, V. R.; GASSER, L. (Ed.). **Proceedings of the First International Conference on Multiagent Systems, June 12-14, 1995, San Francisco, California, USA**. [S.l.]: The MIT Press, 1995. p. 416–422. ISBN 0-262-62102-9.

KAMSA, I. et al. Learning time planning in a distance learning system using intelligent agents. In: **Information Technology Based Higher Education and Training (ITHET), 2015 International Conference on**. [S.l.: s.n.], 2015. p. 1–4.

KARAMPIPERIS, P.; SAMPSON, D. Adaptive learning resources sequencing in educational hypermedia systems. **Educational Technology & Society**, JSTOR, v. 8, n. 4, p. 128–147, 2005.

KRIPKE, S. A. Semantical considerations on modal logic. 1963.

MathNet: An Agent-Based Tutoring System for Supporting Cooperative and Distant Learning, v. 1. 195–200 p.

LENDYUK, T. et al. Individual learning path building on knowledge-based approach. In: IEEE. **Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), 2015 IEEE 8th International Conference on**. [S.l.], 2015. v. 2, p. 949–954.

LIMONGELLI, C. et al. Adaptive learning with the ls-plan system: a field evaluation. **IEEE Transactions on Learning Technologies**, IEEE, v. 2, n. 3, p. 203–215, 2009.

MAHMOUD, C. B. et al. A learning semantic web service for generating learning paths. In: IEEE. **Computer and Information Science (ICIS), 2015 IEEE/ACIS 14th International Conference on**. [S.l.], 2015. p. 627–631.

- MARCOS, L. de et al. Competency-based intelligent curriculum sequencing: comparing two evolutionary approaches. In: IEEE COMPUTER SOCIETY. **Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 03**. [S.l.], 2008. p. 339–342.
- MELIA, M.; BARRETT, R.; PAHL, C. A model-based approach to scorm sequencing. In: **In Proceeding of the Sixth Annual Irish Educational Technology User's Conference (EdTech06) - Research Track**. ILTA. [S.l.: s.n.], 2006.
- MEYER, J.-J. C.; HOEK, W. van der; LINDER, B. van. A logical approach to the dynamics of commitments. **Artificial Intelligence**, Elsevier, v. 113, n. 1, p. 1–40, 1999.
- MONTEIRO, J. P. G. O. **Inteligência Artificial na Educação**. Tese (Doutorado) — Instituto Superior de Engenharia do Porto, 2006.
- MOORE, J. L.; DICKSON-DEANE, C.; GALYEN, K. e-learning, online learning, and distance learning environments: Are they the same? **The Internet and Higher Education**, Elsevier, v. 14, n. 2, p. 129–135, 2011.
- MURRAY, T. Authoring intelligent tutoring systems: An analysis of the state of the art. **International Journal of Artificial Intelligence in Education (IJAIED)**, v. 10, p. 98–129, 1999.
- MURRAY, T. An overview of intelligent tutoring system authoring tools: Updated analysis of the state of the art. In: **Authoring tools for advanced technology learning environments**. [S.l.]: Springer, 2003. p. 491–544.
- NEUMANN, R. Using promela in a fully verified executable ltl model checker. In: SPRINGER. **Working Conference on Verified Software: Theories, Tools, and Experiments**. [S.l.], 2014. p. 105–114.
- NWANA, H. S. Intelligent tutoring systems: an overview. **Artificial Intelligence Review**, v. 4, n. 4, p. 251–277, 1990.
- NWANA, H. S. Software agents: An overview. **Knowledge Engineering Review**, v. 11, p. 205–244, 1996.
- OLIVEIRA, P. P. de et al. Um modelo de integração dos princípios de sistemas tutores inteligentes e e-learning a jogos do tipo mmorpg. 2012.
- PADGHAM, L.; WINIKOFF, M. **Developing intelligent agent systems: A practical guide**. [S.l.]: John Wiley & Sons, 2005.

PALOMINO, C. E. G.; SILVEIRA, R. A.; NAKAYAMA, M. K. Using agent-based adaptive learning environments for knowledge sharing management. **IJKL**, v. 10, n. 3, p. 278–295, 2015.

PINTO, I. I. B. S. **Plataforma para Construção de Ambientes Interativos de Aprendizagem baseados em Agentes**. Tese (Doutorado) — Universidade Federal de Alagoas, 2006.

PNUELI, A. The temporal logic of programs. In: **Proceedings of the 18th Annual Symposium on Foundations of Computer Science**. Washington, DC, USA: IEEE Computer Society, 1977. (SFCS '77), p. 46–57.

POZZEBON, E. et al. **Um modelo para suporte ao aprendizado em grupo em sistemas tutores inteligentes**. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2008.

RAO, A. S. Modeling rational agents within a bdi-architecture. 1991.

RAO, A. S. Decision procedures for propositional linear-time belief-desire-intention logics. In: SPRINGER. **International Workshop on Agent Theories, Architectures, and Languages**. [S.l.], 1995. p. 33–48.

RAO, A. S. Agentspeak (I): Bdi agents speak out in a logical computable language. In: **Agents Breaking Away**. [S.l.]: Springer, 1996. p. 42–55.

RAO, A. S. Decision procedures for propositional linear-time belief-desire-intention logics. In: **Intelligent Agents II Agent Theories, Architectures, and Languages**. [S.l.]: Springer, 1996. p. 33–48.

RAO, A. S.; GEORGEFF, M. P. **Asymmetry Thesis and Side-Effect Problems in Linear-Time and Branching-Time Intention Logics**. 1991.

RICCI, A.; VIROLI, M.; OMICINI, A. Cartago: A framework for prototyping artifact-based environments in mas. In: SPRINGER. **International Workshop on Environments for Multi-Agent Systems**. [S.l.], 2006. p. 67–86.

RICCI, A.; VIROLI, M.; OMICINI, A. A general purpose programming model & technology for developing working environments in mas. In: **5th Inter. Workshop “Programming Multi-Agent Systems”(PROMAS 2007)**. [S.l.: s.n.], 2007. p. 54–69.

RUSSELL, S. J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. 2. ed. [S.l.]: Pearson Education, 2003. ISBN 0137903952.

RUSTICI. Anatomy of a tin can api statement. Rustici Software, 2013.

SÁNCHEZ-BRITO, M.; RUIZ-ASCENCIO, J.; GARCÍA-HERNÁNDEZ, C. F. Scorm cloud for an advanced sequencing of learning objects on lms moodle platform. **Research in Computer Science**, v. 87, p. 19–26, 2014.

SANGINETO, E. et al. Adaptive course generation through learning styles representation. **Universal Access in the Information Society**, Springer, v. 7, n. 1-2, p. 1–23, 2008.

SEARLE, J. R. **Speech Acts: An Essay in the Philosophy of Language**. [S.l.]: Cambridge University Press, 1969.

SECCO, R. L. **Um Ambiente Interativo para Aprendizagem em Fração**. Tese (Doutorado) — Universidade Federal de Alagoas, 2007.

SHMELEV, V.; KARPOVA, M.; DUKHANOV, A. An approach of learning path sequencing based on revised bloom's taxonomy and domain ontologies with the use of genetic algorithms. **Procedia Computer Science**, Elsevier, v. 66, p. 711–719, 2015.

SHOHAM, Y. Agent-oriented programming. **Artificial intelligence**, Elsevier, v. 60, n. 1, p. 51–92, 1993.

SIBALDO, M. A. A. **Arquitetura e Modelos de Interações Cooperativas e Adaptativas entre Agentes Humanos e Artificiais no Domínio de Fração**. Dissertação (Mestrado) — Universidade Federal de Alagoas, 2010.

SKINNER, B. F. **Science and human behavior**. first paperback edition. New York: Free Pr. [u.a.], 1953.

Skinner, B. F. Teaching Machines. **Science**, v. 128, p. 969–977, 1958.

SLEEMAN, D.; BROWN, J. S. (Ed.). **Intelligent tutoring systems**. London: Academic Press, 1982. (Computers and people series). Le verso de la page titre porte la mention : cop. 1982 by Academic Press Ltd - First paperback printing 1985 - 2nd printing (pbk) 1988. ISBN 0-12-648681-6.

STERBINI, A.; TEMPERINI, M. Adaptive construction and delivery of web-based learning paths. In: IEEE. **2009 39th IEEE Frontiers in Education Conference**. [S.l.], 2009. p. 1–6.

STERBINI, A.; TEMPERINI, M. Selection and sequencing constraints for personalized courses. In: IEEE. **2010 IEEE Frontiers in Education Conference (FIE)**. [S.l.], 2010. p. T2C–1.

TEIXEIRA, C.; LABID, S.; NASCIMENTO, E. Modeling the cooperative learner based on it's actions and interactions within a teaching-learning session. In: IEEE. **Frontiers in Education, 2002. FIE 2002. 32nd Annual**. [S.l.], 2002. v. 1, p. T1A–19.

TEIXEIRA, C. M. de S. **Sistema MACSA de Modelagem do Aprendiz em Ambientes de Ensino Aprendizagem Cooperativos Computadorizados**. Tese (Doutorado) — PPGEE/UFMA, Abril 2013.

VICTORIO-MEZA, H.; MEJÍA-LAVALLE, M.; RODRÍGUEZ, G. Advances on knowledge representation of intelligent tutoring systems. In: . [S.l.: s.n.], 2014. p. 212–216.

WAN, S.; NIU, Z. Adaptive learning objects assembly with compound constraints. In: **Computing, Communications and IT Applications Conference (ComComAp), 2014 IEEE**. [S.l.: s.n.], 2014. p. 34–39.

WENGER, E. **Artificial Intelligence and Tutoring Systems: Computational and Cognitive Approaches to the Communication of Knowledge**. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. ISBN 0-934613-26-5.

WOOLDRIDGE, M. A knowledge-theoretic semantics for concurrent metattem. In: SPRINGER. **International Workshop on Agent Theories, Architectures, and Languages**. [S.l.], 1996. p. 357–374.

WOOLDRIDGE, M.; JENNINGS, N. R. Intelligent agents: Theory and practice. **Knowledge Engineering Review**, v. 10, p. 115–152, 1995.

WOOLDRIDGE, M. J. **Reasoning about rational agents**. Cambridge (Mass.), London: The MIT Press, 2000. (Intelligent robots and autonomous agents). ISBN 0-262-23213-8.

WOOLDRIDGE, M. J. **Introduction to Multiagent Systems**. New York, NY, USA: John Wiley & Sons, Inc., 2001. ISBN 047149691X.

YAMANE, E. E. et al. **Modelagem e implementação de uma ferramenta de autoria para construção de tutores inteligentes**. Dissertação (Mestrado) — UFSC, 2006.

YANG, Y. J.; WU, C. An attribute-based ant colony system for adaptive learning object recommendation. **Expert Systems with Applications**, v. 36, n. 2, Part 2, p. 3034 – 3047, 2009. ISSN 0957-4174.

ZHIPING, L.; YU, S.; TIANWEI, X. A formal model of personalized recommendation systems in intelligent tutoring systems. In: **Computer Science Education (ICCSE), 2011 6th International Conference on**. [S.l.: s.n.], 2011. p. 1006–1009.

ZHU, F.; YAO, N. Ontology-based learning activity sequencing in personalized education system. In: IEEE. **Information Technology and Computer Science, 2009. ITCS 2009. International Conference on**. [S.l.], 2009. v. 1, p. 285–288.