

UNIVERSIDADE FEDERAL DO PAMPA

IGOR OLIVEIRA DA FONSECA

**SISTEMA INTELIGENTE DE APOIO AO
ENSINO DE DEDUÇÃO NATURAL**

**Bagé
2019**

IGOR OLIVEIRA DA FONSECA

**SISTEMA INTELIGENTE DE APOIO AO
ENSINO DE DEDUÇÃO NATURAL**

Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Orientadora: Ana Paula Lüdtke Ferreira
Coorientadora: Sandra Dutra Piovesan

**Bagé
2019**

Ficha catalográfica elaborada automaticamente com os dados fornecidos pelo(a) autor(a) através do Módulo de Biblioteca do Sistema GURI (Gestão Unificada de Recursos Institucionais).

F676 Fonseca, Igor Oliveira da

Sistema inteligente de apoio ao ensino de Dedução Natural / Igor Oliveira da Fonseca.

83 p.

Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Pampa, ENGENHARIA DE COMPUTAÇÃO, 2019.

"Orientação: Ana Paula Lüdtke Ferreira; Coorientação: Sandra Dutra Piovesan".

1. Ensino de lógica. 2. Dedução Natural. 3. Sistemas tutores inteligentes. 4. Informática na educação. I. Título.

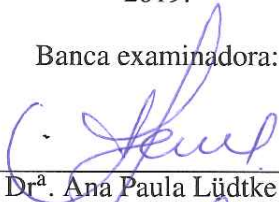
IGOR OLIVEIRA DA FONSECA

**SISTEMA INTELIGENTE DE APOIO AO
ENSINO DE DEDUÇÃO NATURAL**

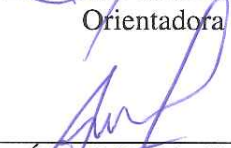
Trabalho de Conclusão de Curso apresentado ao curso de Bacharelado em Engenharia de Computação como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação.

Trabalho de Conclusão de Curso defendido e aprovado em: 26 de novembro de 2019.

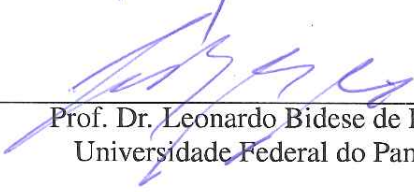
Banca examinadora:



Prof.ª. Dr.ª. Ana Paula Lüdtke Ferreira
Orientadora



Prof. Dr. Érico Marcelo Hoff do Amaral
Universidade Federal do Pampa



Prof. Dr. Leonardo Bidese de Pinho
Universidade Federal do Pampa

AGRADECIMENTO

À orientadora, a quem admiro como profissional, por ter proposto e guiado este trabalho. Aos demais professores do curso, pelos conhecimentos transmitidos.

À minha família e amigos de fora da universidade, por me darem suporte quando necessário.

À minha namorada, pelo incentivo na execução deste trabalho e nas demais empreitadas da vida.

Aos colegas e amigos de curso, que foram a grande alegria desse período universitário.

RESUMO

Este trabalho propõe o desenvolvimento de uma ferramenta de apoio ao processo de ensino-aprendizagem de conteúdos das disciplinas relacionadas à Lógica Matemática, ministradas em cursos da área de Computação. Sistemas de provas formais são trabalhados ao longo dessas disciplinas, com ênfase na Dedução Natural. A formalização de sentenças e a construção de modelos nas lógicas estudadas não são consideradas difíceis pelos alunos, mas a introdução do conteúdo de provas formais faz com que o nível de dificuldade aumente consideravelmente. Identificou-se, assim, a oportunidade de desenvolvimento de uma ferramenta multiplataforma que escolha um trajeto de aprendizagem para cada aluno, dependendo do seu nível de conhecimento, e auxilie na resolução dos exercícios propostos. A ferramenta foi aplicada ao contexto de dedução natural na lógica proposicional. A metodologia do trabalho consiste em uma revisão sistemática da literatura, abordando principalmente os conceitos relacionados a sistemas tutores inteligentes, seguida da proposta e desenvolvimento da ferramenta, que compreende o levantamento de requisitos do sistema, definição dos casos de uso, modelagem da arquitetura, implementação, teste e validação do sistema. O sistema implementado alcançou resultado satisfatório, com seus requisitos essenciais totalmente atendidos, integrando os conceitos pedagógicos para criação de um sistema tutor inteligente com as características previstas a esse tipo de ferramenta. Através de testes, observa-se que a sequência de aprendizagem é personalizada de acordo com o desempenho do estudante. Através da avaliação realizada por profissional da área abordada pelo sistema, foi considerado apto para ser utilizado como apoio ao ensino nas disciplinas em que se propõe, de modo que atende a seu objetivo principal.

Palavras-chave: Ensino de lógica. Dedução Natural. Sistemas tutores inteligentes. Informática na educação.

ABSTRACT

This paper proposes the development of a tool to support the teaching-learning process of Mathematical Logic contents, taught in Computer courses. Formal proof systems are worked throughout these disciplines, with an emphasis on Natural Deduction. The formalization of sentences and the construction of models in the studied logics are not considered difficult by the students, but the introduction of formal proof content causes the level of difficulty to increase considerably. Thus, it was identified the opportunity to develop a multiplatform tool that chooses a learning path for each student, depending on their level of knowledge, and assists in solving the proposed exercises. The tool was applied to the context of natural deduction in propositional logic. The methodology of the work consists of a systematic literature review, mainly addressing the concepts related to intelligent tutoring systems, followed by the proposal and development of the tool, which includes the survey of system requirements, use case definition, architecture modeling, implementation, testing and system validation. The implemented system achieved a satisfactory result, with its essential requisites being totally met, integrating the pedagogical concepts to create an intelligent tutoring system with the characteristics provided for this type of tool. Through testing, it is observed that the learning sequence is tailored to the student's performance. Through the assessment performed by a professional in the area covered by the system, it was considered fit to be used as a support for teaching in the subjects in which it is proposed, meeting its main objective.

Keywords: Computer logics teaching. Natural deduction. Intelligent tutoring systems. Computer education.

LISTA DE FIGURAS

Figura 1	A assimilação ausubeliana.....	25
Figura 2	O modelo 4C/ID	28
Figura 3	Regras básicas de dedução natural para lógica proposicional	32
Figura 4	Regras de dedução natural para lógica de predicados	33
Figura 5	Grafo de conceitos da ferramenta FITS for C++ Programming	34
Figura 6	Interface do sistema tutor J-LATTE	36
Figura 7	Diagrama da arquitetura do sistema	38
Figura 8	Diagrama do Módulo de Domínio.....	40
Figura 9	Esboço da proposta de interface	43
Figura 10	Diagrama de casos de uso do estudante.....	44
Figura 11	Diagrama sequência da resolução de atividades.....	45
Figura 12	Diagrama de classes.....	46
Figura 13	Diagrama entidade-relacionamento	47
Figura 14	Sequenciamento de conceitos utilizado	53
Figura 15	Interface do sistema implementado	56

LISTA DE TABELAS

Tabela 1	Comparação dos trabalhos correlatos.....	37
Tabela 2	Restrições aplicadas às regras de inferência.....	48
Tabela 3	Mensagens de <i>feedback</i> exibidas ao estudante.....	50
Tabela 4	Conjunto de atividades cadastradas no sistema.....	52
Tabela 5	Situação dos requisitos não funcionais.....	57
Tabela 6	Situação dos requisitos funcionais.....	58
Tabela 7	Simulação do sequenciamento de currículo.....	60

LISTA DE ABREVIATURAS E SIGLAS

4C/ID	<i>Four Components Instructional Design</i>
API	<i>Application Programming Interface</i>
CAFe	Comunidade Acadêmica Federada
GPL	<i>GNU General Public License</i>
HTML	<i>HyperText Markup Language</i>
IDE	<i>Integrated Development Environment</i>
IEEE	<i>Institute of Electrical and Electronics Engineers</i>
PDF	<i>Portable Document Format</i>
SciELO	<i>Scientific Electronic Library Online</i>
SQL	<i>Structured Query Language</i>
UML	<i>Unified Modeling Language</i>
UNIPAMPA	Universidade Federal do Pampa

SUMÁRIO

1 INTRODUÇÃO	11
1.1 Justificativa	11
1.2 Problema de pesquisa, objetivos e organização do trabalho.....	15
2 MATERIAL E MÉTODOS	17
2.1 Classificação da pesquisa.....	17
2.2 Revisão da literatura.....	17
2.3 Proposta de ferramenta	18
2.4 Estratégia de desenvolvimento.....	19
2.5 Verificação e validação.....	20
3 REFERENCIAL TEÓRICO	21
3.1 Sistemas tutores inteligentes	21
3.2 Aprendizagem significativa	24
3.3 Design instrucional	25
3.4 Lógica proposicional e lógica de predicados.....	29
3.5 Dedução Natural	31
4 TRABALHOS CORRELATOS	34
5 DESENVOLVIMENTO	38
5.1 Descrição da proposta.....	38
5.2 Requisitos, casos de uso e diagrama de sequência	43
5.3 Arquitetura do sistema	45
5.4 Resolução de exercícios.....	46
5.5 Sequenciamento de currículo.....	50
5.6 Integração com o modelo pedagógico	54
5.7 Testes e validação	55
6 RESULTADOS	56
7 CONSIDERAÇÕES FINAIS	62
REFERÊNCIAS	65
APÊNDICE A – DOCUMENTO DE REQUISITOS	69
APÊNDICE B – CASOS DE USO	71
APÊNDICE C – CASOS DE TESTE	75
APÊNDICE D – QUESTIONÁRIO DE VALIDAÇÃO	82

1 INTRODUÇÃO

1.1 Justificativa

A Lógica Matemática surge no âmbito da Ciência da Computação em dois aspectos diferentes, mas complementares: (i) a formalização do conhecimento referente a sistemas de *hardware* e *software*, expressos em termos de fórmulas de uma lógica apropriada, e (ii) a possibilidade de inferir e verificar propriedades desses sistemas, por meio de sistemas de prova ou de verificações formais (HUTH; RYAN, 2004). Ainda que não diretamente ligada à chamada “lógica de programação”, o processo de prova de teoremas ajuda na consolidação das bases necessárias para a resolução de problemas computacionais, desde a formalização de condições para avaliação em testes condicionais até a organização do pensamento por etapas, necessária à construção de algoritmos (GALAFASSI, 2012). A ciência da programação, define Souza (2002), consiste no desenvolvimento de técnicas que possibilitam a representação do conhecimento e sua manipulação, o que também pode ser formalizado por meio de estruturas matemáticas e lógicas.

Uma lógica é uma linguagem formal (LEWIS; PAPADIMITRIOU, 1997) onde a interpretação das palavras – que, em linguagens que representam lógicas, são denominadas *fórmulas bem formadas* ou simplesmente *fórmulas* – é um mapeamento para o conjunto de valores verdade $\{v, f\}$, onde v significa *verdadeiro* e f significa *falso*. Nas chamadas *lógicas clássicas*, essa interpretação ainda deve obedecer a três axiomas: (i) o princípio da *identidade*, que exige que uma fórmula tenha a ela associada um valor verdade, (ii) o princípio da *não contradição*, que exige que uma fórmula não possa ser verdadeira e falsa simultaneamente, e (iii) o princípio do *terceiro excluído* que afirma que os únicos valores possíveis para uma fórmula sejam verdadeiro ou falso. Esses axiomas, formulados cerca de 400 a.C. pelo filósofo grego Aristóteles, foram mais tarde formalizados dando origem ao que hoje se conhece por axiomas da lógica clássica. Além das lógicas clássicas (proposicional, de primeira ordem, de ordens superiores), extensões *não clássicas* foram propostas ao longo dos anos. Uma lógica é dita não clássica quando suas sentenças deixam de obedecer a um ou mais dos três axiomas acima mencionados. Alguns exemplos de lógicas não clássicas com aplicações em Ciência da Computação são as lógicas modais (STIRLING, 1992; BLACKBURN; RIJKE; VENEMA, 2001), temporais (EMERSON, 1998; HUTH; RYAN, 2004) e *fuzzy* (CHEN; PHAM, 2001).

Existem diversas lógicas que possuem aplicações nas ciências relacionadas à

Computação, tanto de lógicas clássicas quanto de lógicas não clássicas. Entre as lógicas clássicas, as que possuem aplicações mais comuns estão as lógicas Proposicional, de Primeira Ordem e as Monádicas de Segunda Ordem, por questões de decidibilidade computacional. Lógicas, como todas as estruturas formalizadas matematicamente, estabelecem uma linguagem útil para a representação do conhecimento, sem ambiguidades. A partir do conhecimento representado, é possível deduzir formalmente novos conhecimentos, do mesmo modo que a execução de um programa computacional possibilita a construção de novos dados a partir daqueles inicialmente definidos. Usualmente, este objetivo é atingido por meio de sistemas de prova (FITTING, 1996), constituídos a partir de um conjunto finito de axiomas e regras de inferência, que permitem a construção progressiva de novos conhecimentos a partir dos existentes, de forma que se os conhecimentos iniciais eram verdadeiros, todos os outros conhecimentos inferidos são garantidamente verdadeiros. Os sistemas de prova tradicionalmente ensinados nos cursos de graduação são o *Axiomático*, a *Resolução* e a *Dedução Natural*; os dois primeiros por terem fácil implementação computacional – a Resolução é o sistema de prova usado na implementação da linguagem PROLOG – e o último por suas regras possuírem similaridade à maneira com a qual os humanos raciocinam, sendo especialmente útil no desenvolvimento dos processos mentais necessários a uma boa modelagem do conhecimento e à construção de algoritmos.

O ensino sobre lógicas e a correspondente consolidação desse conhecimento, contudo, apresentam-se como desafios nos cursos de graduação. A disciplina de Lógica para Computação, que costuma introduzir este conteúdo nos cursos da área, possui um nível elevado de desistência e baixo rendimento dos alunos. Segundo o levantamento realizado por Galafassi (2012) em uma universidade brasileira, durante o período de 2007/1 a 2012/1 houve uma média de 56% de aprovação na disciplina presencial de lógica. Entre os demais alunos, 24% foram considerados desistentes (tiveram excesso de faltas ou realizaram o cancelamento da disciplina) e 20% foram reprovados, mesmo assistindo a disciplina até o final. Já nos cursos ministrados na modalidade à distância, dados do ano de 2011 apontam que os índices de desistência e reprovação na disciplina são maiores: houve 37% de aprovação, 11% de desistência e 52% de reprovação. Em especial, Galafassi (2012) destaca que as desistências costumam ocorrer quando é apresentado o conteúdo de Dedução Natural na lógica proposicional. Na Unipampa, os números de reprovações e desistência nas disciplinas com esse conteúdo variaram entre 57% e 83% entre os anos de 2015 e 2019 (DIAS; FINGER, 2019).

Entre os motivos para a dificuldade dos discentes com o raciocínio formal destaca-se, conforme Santos e Costa (2005), a ausência de uma base lógico-matemática sólida na educação básica. Essa carência não estimula o desenvolvimento do pensamento lógico nos anos iniciais da formação do aluno, de modo que medidas precisam ser aplicadas para supri-la nos semestres iniciais da graduação. Principalmente nesta etapa da formação, é essencial que o estudante busque construir e ampliar seu aprendizado fora da sala de aula, aponta Janes (2018); dessa forma, terá condições de reservar o espaço das aulas para discutir os pontos mais importantes e desafiadores do tema com o professor. Essa é uma tarefa difícil, visto que utilizando os métodos clássicos de estudo, como livros e materiais didáticos do professor, não há meios de garantir que o estudante esteja aprendendo seu conteúdo por conta própria de maneira correta, sem formar ideias errôneas. Segundo Palazzo *et al.* (2010), quando precisam resolver um exercício com algum grau de dificuldade, os discentes frequentemente não fazem ideia de como começar e desistem da tarefa ou acabam resolvendo o exercício incorretamente, pois não há mecanismo para que percebam instantaneamente onde está seu erro ou como o corrigir.

Uma proposta para lidar com esses problemas é a utilização de tecnologias para assistir o processo de ensino-aprendizagem, fazendo o papel que um professor ou monitor faria. A tecnologia é cada vez mais usada na vida acadêmica na maior parte do mundo, onde professores usam quadros digitais para projetar seu conteúdo e ilustrar aspectos complexos da ciência e engenharia, e estudantes utilizam *tablets* e *laptops* para fazer seus exercícios e acessar materiais de ensino na forma de vídeos, artigos e *podcasts*, afirmam Gross e Rouse (2015) e Croft e Bedi (2004).

O uso de *softwares* educacionais como complemento ao ensino em sala de aula é cada vez mais comum. Uma das categorias desse tipo de ferramenta, como classificam Bertoldi e Ramos (1999), são os *softwares* tutoriais, com a característica de transmitir um conhecimento ao aluno ou complementar o conhecimento previamente adquirido, utilizando para isso materiais pedagógicos instrutivos sobre o tópico em questão. Um sistema tutor inteligente, como destacado por Mitrovic (2003), oferece a vantagem da instrução individualizada sem o ônus de necessitar um tutor humano exclusivo para cada estudante. De acordo com Aravind e Refugio (2019), com seu uso o professor pode desafiar os estudantes a responder uma questão e prover *feedback* imediato (certo ou errado), além de dar dicas úteis àqueles com dificuldades e possibilitar o aprendizado fora da sala de aula.

Segundo Reiser, Anderson e Farrell (1985), o ensino particular – em que cada estudante possui um professor dedicado – é considerado a forma mais eficaz de instrução.

Contudo, limitações práticas como carga de trabalho e tempo impedem que os instrutores proporcionem aos estudantes *feedback* individual em turmas grandes (BUCHANAN, 2000). Na tutoria privada, o tutor pode fornecer material adequado aos conhecimentos reais do estudante e fornecer tanta ajuda quanto necessária. Entretanto, dispor de tantos tutores humanos quanto o número de alunos não é viável do ponto de vista econômico. No âmbito dos sistemas tutores inteligentes, o objetivo é que o sistema simule as atividades de um tutor humano, amenizando este problema ao oferecer instrução individualizada.

O *feedback formativo*, definido por Shute (2008) como uma “informação comunicada ao estudante com a intenção de modificar seu pensamento ou comportamento para melhorar o aprendizado ou desempenho”, produz vantagens durante o tempo de estudo. Seja o *feedback* fornecido por um professor ou um computador, ele visa propiciar conceitos e habilidades específicas mais precisas, evitando a formação de conceitos errôneos pelo estudante devido ao mau entendimento do conteúdo estudado. O fornecimento de *feedback* imediato tem a vantagem de reduzir o desperdício de tempo do estudante em caso de dificuldade na resolução de uma atividade, além de poder evitar que ele desista por não conseguir avançar na solução.

Identifica-se, assim, uma oportunidade para qualificar o ensino de sistemas de prova por meio do desenvolvimento de um sistema de apoio ao aprendizado de Dedução Natural, no contexto da lógica proposicional, em que o aluno possa ser tutorado ao longo do processo. De acordo com seu desempenho, é identificado o perfil de desenvolvimento do aprendizado do estudante e traçado o melhor caminho para suprir suas necessidades, orientando o processo de construção de uma prova. Conforme constatam Ma e Adesope (2014), a utilização de um sistema tutor inteligente tem impacto significativo nos resultados positivos de aprendizagem dos estudantes, com ganhos notáveis quando comparada à instrução conduzida por um professor para grandes grupos ou a outros tipos de instrução computacional sem personalização.

A proposta do sistema é baseada no conceito de aprendizagem significativa, da teoria de aprendizagem de Ausubel (2003). Como explicam Pelizzari *et al.* (2002), a teoria de Ausubel propõe que os conhecimentos prévios dos alunos sejam valorizados, para que possam construir estruturas mentais que permitam descobrir e redescobrir outros conhecimentos, caracterizando, assim, uma aprendizagem eficaz. Com objetivo de qualificar o elemento da aprendizagem autônoma em situação real de educação, a proposta de sistema faz uso de conceitos do *design instrucional*, que é compreendido como o planejamento do ensino-aprendizagem, incluindo atividades, estratégias, sistemas de avaliação, métodos e

materiais instrucionais. Como explicam Filatro e Piconez (2004), essa metodologia tem sido vinculada à produção de materiais didáticos.

1.2 Problema de pesquisa, objetivos e organização do trabalho

O problema de pesquisa deste trabalho divide-se em dois questionamentos. “É possível desenvolver um sistema tutor inteligente capaz de dar *feedback* ao aluno enquanto são realizados os exercícios de prova de teoremas? É possível melhorar os índices de aprendizagem nas disciplinas de lógica para computação com ferramentas de tutoria inteligente?” Assim, são estabelecidos os problemas de pesquisa a curto e a longo prazo, respectivamente.

Este trabalho tem como objetivo desenvolver um sistema multiplataforma de apoio ao processo de ensino-aprendizagem de conteúdos de lógica para computação, que escolha um trajeto de aprendizagem para cada aluno, dependendo do seu nível de conhecimento e que produza apoio e *feedback* imediatos.

São também objetivos desse trabalho:

- Identificar as tecnologias e técnicas envolvidas no desenvolvimento de um sistema tutor inteligente.
- Integrar conceitos de Informática na Educação aplicados com finalidade de utilização no ensino de graduação de Lógica para Computação.
- Utilizar as técnicas de Engenharia de Software para produzir uma ferramenta computacional de ensino.
- Produzir um sistema que pode ser utilizado no ambiente acadêmico.
- Apoiar a produção de conhecimento na área de sistemas tutores inteligentes, com vistas à qualificação futura do ensino de graduação na UNIPAMPA.

As demais partes deste trabalho estão organizadas da seguinte forma: no capítulo 2 são descritos os métodos e o material utilizados para o desenvolvimento do trabalho. No capítulo 3 são mostrados os aspectos teóricos estudados para o desenvolvimento do trabalho, abordando sistemas tutores inteligentes, aprendizagem significativa, design instrucional, lógica proposicional e de predicados e dedução natural. O capítulo 4 apresenta uma discussão sobre os trabalhos correlatos. Já o capítulo 5 contém o desenvolvimento do trabalho, consistindo na apresentação de sua arquitetura e aspectos relevantes de modelagem. O capítulo 6, por sua vez, discute sobre os resultados obtidos com a pesquisa.

Finalmente, o capítulo 7 apresenta as considerações finais sobre este trabalho e possíveis caminhos para sua expansão.

2 MATERIAL E MÉTODOS

2.1 Classificação da pesquisa

Esta pesquisa caracteriza-se como um estudo de natureza aplicada, se tratando de uma proposta de desenvolvimento de *software* para aplicação prática visando a solução de um problema específico. Sua abordagem predominante é qualitativa, visto que as informações referentes aos aspectos pedagógicos, como nesta pesquisa, são baseadas na interpretação de eventos observados e seu significado (LUDKE; ANDRÉ, 2011; BAUER; GASKELL, 2017).

Quanto ao seu objetivo, a pesquisa pode ser classificada como exploratória. Este tipo de pesquisa compreende estudos bibliográficos e análise de exemplos que auxiliam na compreensão do problema. No contexto de seus procedimentos, caracteriza-se como uma pesquisa bibliográfica em sua fase inicial, seguida da fase experimental, com o desenvolvimento e avaliação de um objeto de estudo que segue um planejamento com a formulação do problema (TRIVIÑOS, 1987).

2.2 Revisão da literatura

Na primeira etapa do projeto, foi realizada uma revisão da literatura para compor o referencial teórico. O referencial construído compreende conceitos relacionados ao aprendizado mediado por tecnologias da informação, utilizando as palavras-chave *sistemas tutores inteligentes*, *aprendizagem significativa* e *design instrucional*, além dos conceitos sobre o domínio de aplicação desse projeto *lógica proposicional*, *lógica de predicados* e *dedução natural*. Em inglês, as palavras-chave utilizadas foram *intelligent tutoring systems*, *meaningful learning*, *instructional design*, *propositional logic*, *predicate logic* e *natural deduction*.

No processo de definição das fontes de pesquisa, foram selecionados mecanismos de busca, bibliotecas digitais e periódicos relevantes ao tema proposto. As fontes selecionadas foram: *ScienceDirect*, *IEEE Transactions on Education*, *CAFe*, *Plataforma Supupira*, *SciELO*, *International Conference on Intelligent Tutoring Systems* e *Google Scholar*.

Feita a seleção das fontes de pesquisa e palavras-chave utilizadas, foram especificados os critérios de análise e filtragem do material encontrado para sua inclusão ou

não no referencial teórico. Os critérios utilizados foram a avaliação existente no *Web-qualis* do sistema CAPES, número de citações, relevância ao tema de aplicação e data de publicação.

2.3 Proposta de ferramenta

A partir do problema identificado, foi elaborada uma proposta de ferramenta para atender aos objetivos do projeto. No processo de planejamento desta ferramenta foi realizado o levantamento de requisitos, elaboração dos casos de uso e modelagem da arquitetura.

O requisitos iniciais do sistema foram levantados junto à professora orientadora deste trabalho, docente responsável pela disciplina de Lógica para Computação no Campus Bagé da UNIPAMPA, que propôs as características desejadas para um *software* de apoio ao ensino da construção de provas de Dedução Natural. A partir destes requisitos, agregados aos conhecimentos adquiridos durante a etapa de revisão da literatura, alguns requisitos complementares foram levantados, levando em consideração as características e objetivos comuns desse tipo de aplicação. Com base nas abordagens existentes e particularidades da aplicação deste problema, foram selecionadas as características necessárias para a construção de um sistema tutor inteligente com aplicação no ensino de lógica no contexto da Dedução Natural.

Para a modelagem do sistema foi escolhido o padrão UML, uma linguagem gráfica para visualização, especificação, construção e documentação de artefatos de sistemas complexos de *software* (BOOCH; RUMBAUGH; JACOBSON, 2006). Esta etapa do projeto é útil para evitar erros que podem ocorrer devido à ausência de planejamento da estrutura, além de gerar uma documentação para consultas futuras. Assim, a partir do levantamento de requisitos, foram elaborados os diagramas de casos de uso e de classes do sistema. O *diagrama de casos de uso*, segundo Furlan (1998), descreve a visão externa de um sistema e suas interações com o mundo. Um caso de uso expressa o comportamento do sistema em algumas situações respondendo a requisições de um dos atores envolvidos. Já o *diagrama de classes* consiste, segundo a abordagem orientada a objetos, na identificação das classes que definem o sistema, seus elementos constituintes e suas respectivas relações. O comportamento das classes é descritos por meio de seus atributos e métodos e o estado de cada elemento do sistema depende da sua instanciação em tempo de execução (SILVA; VIDEIRA, 2001). Já o *diagrama de sequência* ilustra uma interação

segundo uma visão temporal, relacionando os objetos que interagem entre si e a sequência de eventos que ocorrem entre eles após uma interação do ator. O *software* utilizado para produzir os diagramas apresentados foi o Astah¹.

Também nesta etapa do projeto foi elaborado o diagrama entidade-relacionamento, que mostra a estrutura do banco de dados da aplicação. Este diagrama é útil para representar os dados e suas relações em alto nível, isto é, em uma forma de fácil entendimento pelos usuários. A ferramenta utilizada para a construção deste diagrama foi o brModelo². Ele oferece a vantagem de gerar, a partir do diagrama entidade-relacionamento, o modelo lógico e o código SQL para a criação, em um sistema de gestão de banco de dados, da estrutura descrita. O modelo lógico representa o diagrama de maneira mais estrutural, como tabelas contendo seus atributos, chaves primárias e relações (chaves estrangeiras).

2.4 Estratégia de desenvolvimento

Para o desenvolvimento do sistema, foi escolhida a linguagem de programação Java³ devido à sua característica de ser uma linguagem multiplataforma e orientada a objetos. Além disso, a linguagem é utilizada para o desenvolvimento de aplicações com interface gráfica para sistemas *desktop*, que é uma necessidade neste projeto. Java é utilizada em ferramentas comumente aplicadas em cursos de computação, como por exemplo a aplicação para modelagem de banco de dados brModelo e o conjunto de ferramentas de síntese lógica Karma⁴. Sendo assim, é habitual que os usuários desse tipo de programa já possuam o ambiente Java instalado em seus sistemas Windows e/ou Linux. Para o processo de desenvolvimento, foi utilizada a IDE NetBeans⁵, que contém ferramentas que auxiliam a implementação de interfaces gráficas em Java. Outra linguagem estudada e considerada para utilização neste projeto foi o JavaScript, tipicamente voltado a programação para *web*. Com a ferramenta Electron⁶, é possível gerar uma aplicação *desktop* a partir dos códigos dessa linguagem. Porém, devido à maior escassez de integração e suporte dessa plataforma, além de não conter uma API tão robusta e não possuir uma ferramenta para auxiliar no desenvolvimento da interface gráfica, não foi a opção escolhida.

Inicialmente, o sistema de gestão de banco de dados escolhido havia sido o Post-

¹<http://astah.net/>

²<http://www.sis4.com/brModelo/>

³https://www.java.com/pt_BR/

⁴<http://www.inf.ufrgs.br/logics/downloads/>

⁵<https://netbeans.org/>

⁶<https://electronjs.org/>

greSQL. Porém, por um erro de planejamento, apenas foi notado nas últimas semanas de implementação o fato de que este banco de dados não suporta ser incorporado a uma aplicação Java, de modo que o usuário precisaria ter o servidor PostgreSQL instalado em sua máquina e saber seu usuário e senha para poder utilizar ambos em conjunto. Essa dependência foi considerada inviável, tornando potencialmente mais difícil a utilização de uma ferramenta que deve ser simples, portanto foi necessário buscar uma alternativa. No momento em que foi necessário gerar a versão para distribuição do sistema, então, o sistema de gestão de banco de dados anterior foi substituído pelo H2, desenvolvido em Java e capaz de ser incorporado a aplicações dessa linguagem. Após passado o período de adaptação na implementação, observou-se que o banco de dados H2 atendeu às demandas do sistema.

O sistema é distribuído sob a licença de código aberto GPL, que permite a criação de códigos derivados, mas exige que tenham a mesma licença para que sejam distribuídos. O projeto está disponibilizado em um repositório *online* na plataforma GitHub⁷, bastante difundida entre desenvolvedores para este fim. O objetivo dessas escolhas é a possibilidade de divulgação da ferramenta na comunidade de *software* livre, visto que seu projeto prevê a inclusão de novas funcionalidades em trabalhos futuros, que podem ser desenvolvidas colaborativamente por outras pessoas.

A metodologia de desenvolvimento prevê entregas incrementais, de modo que as funcionalidades serão implementadas uma por vez de acordo com a prioridade, gerando novas versões do sistema. O histórico dessas versões ficará armazenado na plataforma do repositório *online*.

2.5 Verificação e validação

A verificação do sistema ocorreu através da modelagem e execução de um conjunto casos de teste, projetados após a etapa de modelagem na forma de testes funcionais. Após a conclusão dos testes, foi realizada a etapa de validação, que segundo Sommerville (2011) é o momento em que o software é verificado – após o desenvolvimento – para garantir que é o que o cliente quer. A validação do sistema deu-se por meio da utilização e avaliação do mesmo por especialista na área de domínio da aplicação, que realizou uma análise quali-quantitativa a partir de um conjunto de perguntas.

⁷<https://github.com/igorfonck/logicits>

3 REFERENCIAL TEÓRICO

3.1 Sistemas tutores inteligentes

Os sistemas tutores inteligentes são ferramentas tecnológicas educacionais utilizadas no auxílio à tutoria personalizada de estudantes. Para tal, empregam técnicas de inteligência artificial para sugerir atividades de ensino ou um caminho otimizado de aprendizagem. O objetivo geral dos sistemas tutores inteligentes, definem Brusilovsky *et al.* (1999), é utilizar os conhecimentos sobre o domínio de aplicação, sobre o estudante e sobre as estratégias de ensino para dar suporte ao aprendizado flexível e individualizado.

A informática na educação pode ser apenas uma forma de transferir uma aula presencial ou a leitura de material didático para outro tipo de ambiente. Os sistemas computacionais comuns de aprendizado, como enfatizam Hafidi e Bensebaa (2014), mostram o conteúdo e material educacional da mesma forma a todos os estudantes ou permitem que eles escolham seu próprio caminho através do material. Esse caminho, contudo, não é necessariamente o mais efetivo em termos dos conhecimentos prévios ou das necessidades do estudante. Os sistemas tutores inteligentes buscam resolver esses problemas, apresentando o conteúdo de maneira personalizada de acordo com características identificadas no perfil do aluno.

Como classificado por Brusilovsky *et al.* (1999), existem três funcionalidades elementares dos sistemas tutores inteligentes: (i) sequenciamento de aprendizado, (ii) análise inteligente das soluções do estudante e (iii) suporte interativo à resolução de problemas. Como poderá ser observado no capítulo 4, alguns sistemas já existentes contêm as três características, enquanto outros possuem apenas uma ou duas.

A primeira das funcionalidades listadas por Brusilovsky *et al.* (1999), *sequenciamento de aprendizado*, tem o objetivo de disponibilizar ao estudante a sequência de unidades de conhecimento individualmente planejada mais conveniente a ele, além da sequência de tarefas (exemplos, questões, problemas, etc.) com a complexidade mais adequada à sua capacidade. Em outras palavras, ajuda o estudante a achar o caminho ideal através do material de ensino. Há dois tipos essenciais de sequenciamento: ativo e passivo. O sequenciamento ativo implica a existência de um objetivo de aprendizagem, ou seja, um subconjunto de conceitos do domínio ou tópicos a serem aprendidos. Sistemas com sequenciamento ativo podem gerar o melhor caminho individual para atingir este objetivo. Já o sequenciamento passivo é uma tecnologia reativa e não requer um objetivo de

aprendizagem ativo. Realiza a adaptação do sequenciamento conforme o usuário é capaz ou não de resolver um problema ou responder uma questão (ou questões) corretamente. Seu objetivo é oferecer ao estudante um subconjunto do material de ensino disponível, que pode preencher as lacunas no conhecimento dele. Na maioria dos sistemas tutores inteligentes com sequenciamento é possível distinguir dois níveis: (i) sequenciamento de alto nível ou sequenciamento de conhecimentos, que é estruturado para determinar o próximo sub-objetivo de aprendizagem: próximo conceito, conjunto de conceitos, tópico ou conteúdo a ser ensinado, e (ii) sequenciamento de baixo nível ou sequenciamento de tarefas, que é organizado para determinar a próxima tarefa (problema, exemplo, teste) dentro do sub-objetivo pertinente.

A segunda funcionalidade desse tipo de sistema, chamada *análise inteligente das soluções do estudante*, lida com as respostas finais do estudante aos problemas propostos, sem importar de que forma foram obtidas. Para ser considerado inteligente, um analisador de soluções precisa decidir se a solução está correta ou não, encontrar o que exatamente está errado ou incompleto, e preferencialmente identificar quais conhecimentos faltantes ou incorretos podem ser responsáveis pelo erro. Esta última funcionalidade é conhecida como diagnóstico de conhecimento. Analisadores inteligentes podem entregar ao estudante um *feedback* extensivo de erros e atualizar seu modelo de estudante.

Já a última das três funcionalidades dos sistemas tutores inteligentes, *suporte interativo à resolução de problemas*, fornece ao estudante ajuda inteligente a cada passo da resolução dos problemas, ao invés de aguardar a resposta final. O nível de ajuda pode variar, podendo sinalizar sobre um passo errado, dar uma dica ou mesmo executar o próximo passo pelo estudante. Os sistemas que implementam essa funcionalidade (chamados de tutores interativos) podem observar as ações do estudante, entendê-las, e usar esse entendimento para fornecer ajuda e atualizar o modelo de estudante.

A tutoria individual demonstra ser uma forma bastante efetiva de ensino. Como apresentado por Bloom (1984), estudantes trabalhando com tutores humanos individuais aprendem os conteúdos até quatro vezes mais rápido que aqueles em situação comum de sala de aula, que inclui assistir às aulas, ler textos e trabalhar sozinho em exercícios extraclasse. Os sistemas tutores inteligentes são, em muitos aspectos, similares aos tutores humanos. Baseados na ciência cognitiva e inteligência artificial, eles demonstram bons resultados em aplicações de várias áreas da educação. O estudo realizado por Regian (1997) mostra que tutores inteligentes podem produzir as mesmas melhorias resultantes da tutoria individual e podem reduzir de um terço a metade do tempo necessário para

aprendizado. Atualmente, esses sistemas podem ser encontrados aplicados a áreas como Matemática, Física, Linguagens e Informática. Segundo Woolf *et al.* (2001), a aceitação e popularidade dos sistemas tutores inteligentes vem crescendo devido a resultados como o aumento do desempenho dos estudantes, o desenvolvimento cognitivo independente e o tempo reduzido para o estudante adquirir habilidades e conhecimentos.

Existem dois modelos principais quanto à representação dos conhecimentos do estudante por um sistema tutor inteligente: *model-tracing* e modelagem baseada em restrições. Segundo descrito por Ma e Adesope (2014), o *model-tracing* caracteriza um tutor cognitivo baseado na teoria de arquitetura cognitiva ACT-R (*Adaptive Control of Thought—Rational*) de Anderson e Bellezza (1993). Nesse modelo, a habilidade a ser aprendida é modelada por um conjunto de regras constituídas por operações e as condições que as disparam. As regras formam uma simulação psicologicamente realista de como os humanos resolvem os problemas do domínio e podem ser ativadas para resolvê-los automaticamente (ANDERSON *et al.*, 1995). As operações no modelo de domínio são mostradas na interface, onde o estudante pode selecioná-las para aplicar à resolução do problema. Se ocorre um erro, o estudante recebe *feedback* imediato e pode escolher outra operação. Quando o estudante usa uma regra, um procedimento Bayesiano chamado *rastreamento de conhecimento* é usado para estimar a probabilidade dela ter sido aprendida, sendo o modelo de estudante constituído por essas probabilidades atribuídas às regras no modelo de domínio. Já o segundo modelo, *modelagem baseada em restrições*, é baseado na teoria de Ohlsson (1994) de aprendizagem a partir de erros. Este modelo representa o conhecimento do domínio como restrições lógicas ao relacionar cada restrição a estados que podem surgir na solução de um problema. Uma restrição consiste em três componentes: (i) uma condição de relevância que indica quando a restrição é aplicável, (ii) uma condição de satisfação que testa o estado atual da solução do estudante e (iii) uma mensagem de *feedback* que, caso seja ativada, orienta o estudante quanto ao erro e revisa o princípio que foi violado por ele (MITROVIC; MARTIN; SURAWEERA, 2007). As restrições são similares às regras do rastreamento de modelo, exceto que não podem ser executadas para gerar uma solução do problema. O conhecimento do estudante pode ser modelado baseado nas restrições que foram relevantes, satisfeitas e violadas durante a resolução dos problemas.

Uma extensiva análise foi realizada por Hafidi e Bensebaa (2014) sobre a eficiência dos sistemas tutores inteligentes como ferramenta de melhora no desempenho de estudantes. Seus resultados mostram que estudantes que receberam tutoria inteligente su-

peraram estudantes de aulas convencionais em 92% das avaliações controladas, e o ganho em performance foi grande suficiente para ser considerado de importância substancial em 78% dos 50 estudos. A efetividade média nos 50 estudos foi 66%, que é considerado um efeito moderado a alto para estudos das ciências sociais.

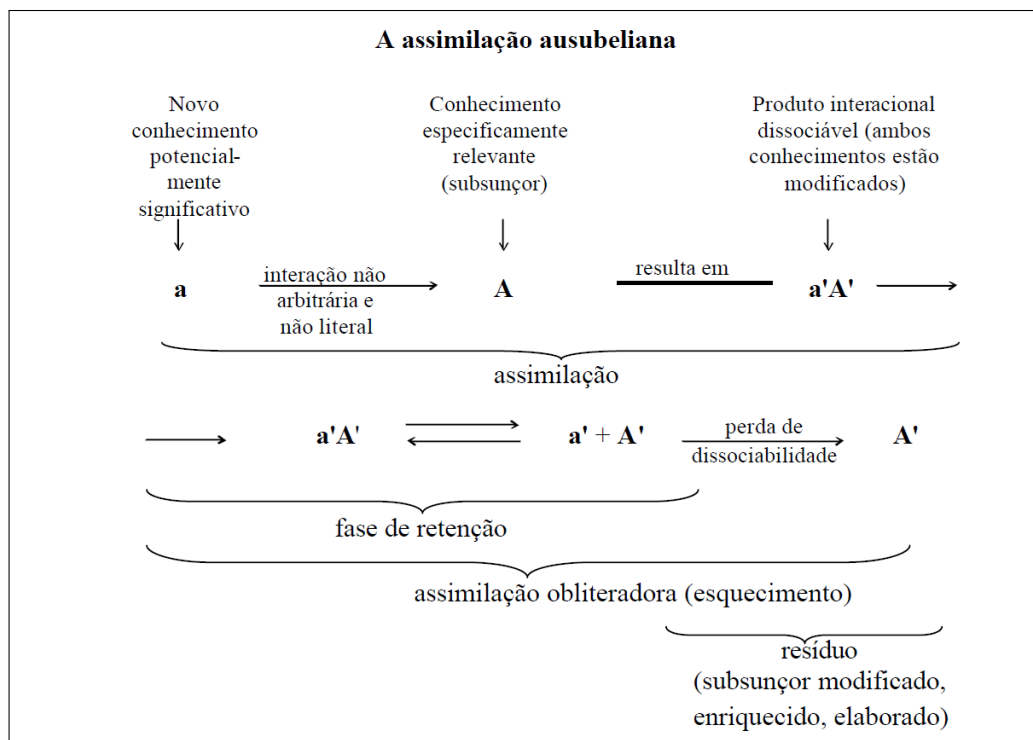
3.2 Aprendizagem significativa

A utilização de tecnologias computacionais na educação visa alcançar um avanço em relação aos instrumentos tradicionais de ensino, em termos de autonomia do estudante e eficiência do aprendizado. O processo de aprendizagem deve ser observado na construção das ferramentas educacionais para que seja possível levar os estudantes à obtenção de uma aprendizagem efetiva. Os modelos que visam descrever como se dá o processo de aprendizagem pelos indivíduos, chamados *teorias de aprendizagem*, são estudados nas áreas da Psicologia e da Educação (CASTRO; SANTOS *et al.*, 2013). O conhecimento e estruturação desse processo de aprendizagem serve como diretriz para definir métodos de organização das matérias e suas experiências de ensino.

A teoria da aprendizagem significativa, de Ausubel (2003), tem como conceito central os fatores necessários para a construção de novos significados. Eles são: material instrucional estruturado de maneira lógica, existência de conhecimento organizado e relacionável ao novo conteúdo, e vontade de relacionar a informação nova com a já existente. Essa estrutura cognitiva pré-existente é denominada *subsunçor*. O processo de assimilação de um novo conceito passa por algumas etapas até que esteja completo. A Figura 1 mostra uma representação esquemática desse processo, onde ocorre a retenção do novo conhecimento ainda associado ao conhecimento pré-existente, até que ocorra a perda de dissociabilidade e assimilação do novo conhecimento. O esquema mostra que a aprendizagem significativa ocorre quando quando ideias expressas simbolicamente interagem de maneira *não arbitrária* – indicando um conhecimento já existente na estrutura cognitiva do sujeito que aprende – e *não literal*. O esquecimento é considerado uma continuação natural da aprendizagem significativa, mas há um resíduo – o subsunçor modificado. Os novos conhecimentos são obliterados, ou subsumidos, mas após o processo estão contidos no subsunçor, o que facilita a reaprendizagem (MOREIRA, 2006).

Em um dos pontos centrais de sua abordagem, como destacado por Nogueira *et al.* (2000), Ausubel afirma que as novas informações no ambiente de aprendizagem devem relacionar-se, de alguma forma, com um elemento relevante da estrutura de conhecimento

Figura 1 – A assimilação ausubeliana



Fonte: Moreira (2006)

prévio do indivíduo. O subsunçor é uma característica particular de cada indivíduo. Logo, pessoas diferentes terão conhecimentos prévios diferentes, e para que os mesmos obtenham uma aprendizagem significativa sobre um determinado tema, as informações a eles oferecidas devem ser diferentes.

Considerando a teoria de aprendizagem significativa, o esperado em um material didático é que ele seja capaz de estruturar e relacionar os conceitos propostos de forma lógica. Em materiais mais interativos e avançados, como sistemas computacionais, a utilização de inteligência artificial pode possibilitar a adaptação do conteúdo ao estudante, buscando gerar um caminho de aprendizagem mais adequado aos conhecimentos prévios de cada indivíduo. Alguns trabalhos que utilizam esta técnica são apresentados e discutidos no capítulo 4.

3.3 Design instrucional

O área de conhecimento concentrada nos métodos e recursos empregados a processos de ensino-aprendizagem é denominada *design instrucional*. Utiliza-se o design instrucional na concepção de cursos e aulas individuais e na construção de materiais di-

dáticos como impressos, vídeos e *softwares*. Segundo a definição de Merrill *et al.* (1996), o design instrucional corresponde a uma prática que permite a criação de experiências de ensino com o intuito de promover a aquisição de conhecimentos e de competências de uma forma mais eficiente e apelativa.

Os sistemas de apoio ao ensino, portanto, devem seguir um modelo de desenvolvimento planejado e estruturado de modo a atingir os objetivos pedagógicos definidos. Nesse sentido, o design instrucional entra como a forma ou método de apresentar as informações e instruções ao aprendiz, apontando alguns aspectos que devem ser observados para garantir a maior eficiência do ensino.

A teoria da carga cognitiva (SWELLER, 1994) foi desenvolvida para fornecer diretrizes para auxiliar na apresentação de informações de uma maneira que estimule atividades de ensino que otimizam o desempenho intelectual. A teoria assume dois tipos de memória que atuam de formas diferentes: (i) a *memória de trabalho*, que possui capacidade limitada e armazena elementos parcialmente independentes, e (ii) a *memória de longo prazo*, sem limites definidos e que contém esquemas que variam em seu nível de inter-relação. Um esquema, nesse contexto, é a estrutura mental que a memória toma forma para compreender, lembrar e relacionar informações, segundo proposto inicialmente por Bartlett (1958).

Estas características da arquitetura cognitiva humana foram usadas para projetar procedimentos instrucionais baseados na ideia de que a carga de memória de trabalho deve ser reduzida enquanto a construção de esquemas permanentes deve ser incentivada. A teoria da carga cognitiva fornece informações pertinentes sobre como o material apresentado influencia na experiência de aprendizado.

A memória de trabalho é considerada aquela da qual os humanos estão conscientes e podem monitorar seu conteúdo. Ela é limitada, podendo apenas manter até sete itens ativos por vez (MILLER, 1956). Quando os elementos precisam ser processados, organizados ou comparados, esse número diminui para dois ou três. A verdadeira habilidade intelectual humana está na memória de longo prazo, de modo que os elementos presentes na memória de trabalho precisam ser elaborados até que se tornem parte de um esquema permanente na memória de longo prazo.

A carga cognitiva associada à apresentação do material de ensino pode ser classificada em três categorias. A *carga cognitiva intrínseca* está relacionada à carga de memória de trabalho imposta pelos diferentes tópicos de estudo. Isso depende do número de elementos que precisam ser processados simultaneamente, enquanto este número depende

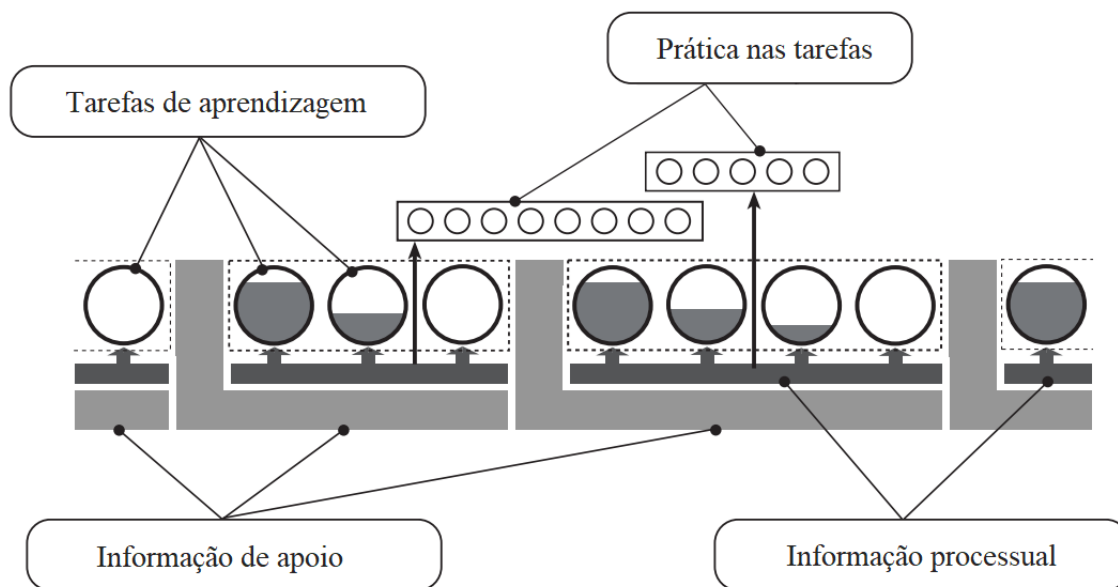
do grau da interatividade entre os elementos. Se há pouca interdependência, a carga é baixa, pois os elementos podem ser aprendidos em série. Esta carga cognitiva não é passível da manipulação, visto que é inerente ao material que está sendo estudado. A *carga cognitiva estranha*, por sua vez, é aquela introduzida devido à maneira como o material é apresentado, que ocupa a memória de trabalho com elementos que não são diretamente relevantes ao aprendizado. Quando diversos elementos que ainda não foram fixados na memória de longo prazo são apresentados em paralelo, por exemplo, é gerada desnecessariamente uma carga cognitiva estranha elevada. Já a *carga cognitiva pertinente* é aquela introduzida pela interface que é diretamente relevante para a compreensão do material estudado e a formação de esquemas mentais. O design instrucional ideal deve diminuir a carga cognitiva estranha enquanto eleva a carga cognitiva pertinente.

A utilização dos modelos de design instrucional ajuda os educadores a elaborar materiais com padrões instrucionais avaliados com sucesso em experiências anteriores. Existem dezenas desses modelos (ANDREWS; GOODSON, 1980), cada um com especificidades mais adequadas a alguns tipos de aplicação. O modelo *ADDIE*, por exemplo, é voltado para a elaboração de aulas e cursos, com maior aplicação em treinamentos e capacitações empresariais (BRANCH, 2009). O modelo *Dick e Carey*, por sua vez, se destaca dos demais por colocar em foco o processo educativo, dando mais atenção ao roteiro a ser desenvolvido pelo estudante e aos resultados esperados (DICK; CAREY; CAREY, 2005). Ambos levam em consideração o público alvo no momento da concepção do material, exigindo um conhecimento prévio e impedindo a adaptabilidade a novos públicos.

O modelo de design instrucional conhecido como 4C/ID é considerado um dos mais influentes (MELO; MIRANDA, 2016). Ele propõe uma metodologia de aprendizagem prática, sendo mais adequado a ferramentas de aprendizagem baseadas em exercícios. Inicialmente, os exercícios são entregues com bastante assistência (informações, exemplos e guias), enquanto progressivamente ela é eliminada das tarefas, até que o aluno tenha condições de resolver o problema sem qualquer ajuda (VAHLDICK; SANTIAGO; RAABE, 2007). Segundo o autor do modelo original em (MERRIËNBOER; KESTER, 2005), o modelo envolve a integração de conhecimentos, competências, atitudes e a capacidade de coordenar diferentes competências em termos qualitativos e, muitas vezes, a transferência do que é aprendido para novas situações. Este modelo instrucional assume os princípios da teoria da carga cognitiva e integra muitos dos resultados experimentais alcançados por ela.

Para implantar as estratégias de aprendizagem, o modelo 4C/ID recomenda a presença de quatro componentes inter-relacionados, cuja representação é mostrada na Figura 2.

Figura 2 – O modelo 4C/ID



Fonte: Adaptado de Merriënboer, Clark e Croock (2002)

As *tarefas de aprendizagem* – representadas pelos círculos grandes na Figura – são consideradas a parte principal do modelo e estão organizadas em “classes de tarefas”, das mais simples para as mais complexas. As classes de tarefas são categorias de tarefas com nível equivalente. Tarefas dentro da mesma classe podem apresentar grande variação quanto ao seu conteúdo e começam oferecendo ao estudante elevado suporte integrado (representado pela cor cinza dentro dos círculos), que desaparece gradualmente até o final da classe (isto é, um processo de *scaffolding*¹). A *informação de apoio* – representada na Figura como barras em forma de L na cor cinza mais clara – oferece auxílio ao aprendizado e desempenho em aspectos não-recorrentes das tarefas de aprendizagem, sendo especificada por classe de tarefas e estando sempre disponível aos estudantes. Permite ao aluno resolver os problemas de forma eficiente, fazendo a ligação entre os seus conhecimentos prévios e os novos através de modelos mentais, estratégias cognitivas e *feedback* cognitivo. A *informação processual* – representada na figura por uma barra horizontal com setas, na cor cinza escura – permite, por meio de um algoritmo, dar informações de

¹Em educação, *scaffolding* refere-se a uma variedade de técnicas de instrução usadas para mover os alunos progressivamente em direção a uma maior compreensão e independência no processo de aprendizagem. O termo é uma palavra inglesa que significa “ter/pôr andaimes”, como os usados na construção civil.

como os aspectos mais recorrentes das tarefas ou itens de prática devem ser executados. É especificada por habilidade recorrente constituinte e organizada em pequenos segmentos de informação e *feedback* corretivo mostrados no momento exato em que são necessários (do inglês *just-in-time information*), desaparecendo conforme os estudantes adquirem perícia. A *prática nas tarefas* – na Figura, representada pelos conjuntos de círculos menores – permite o treino de competências mais rotineiras, visando alcançar o nível necessário de automaticidade pelo estudante através da repetição dos conjuntos de regras complexas. É organizada em sessões que são integradas com tarefas de aprendizagem, buscando incluir itens de prática para todas as aplicações das regras.

3.4 Lógica proposicional e lógica de predicados

O objetivo da Lógica na Ciência da Computação é desenvolver linguagens para modelar as situações que encontramos como profissionais da área, de maneira que seja possível raciocinar sobre elas formalmente. Raciocinar sobre situações significa construir argumentos sobre elas. Busca-se fazer isso formalmente, para que os argumentos sejam válidos e possam ser defendidos rigorosamente ou executados em uma máquina (HUTH; RYAN, 2004).

Para desenvolver argumentos rigorosos, é preciso fazer uso de uma linguagem em que seja possível expressar sentenças na forma de sua estrutura lógica. Uma dessas linguagens é a *lógica proposicional*, que é baseada em proposições, ou sentenças declarativas, cuja única informação associada é o seu valor verdade, ou seja, se são verdadeiras ou falsas.

As lógicas representadas são simbólicas por natureza. As sentenças declarativas são traduzidas em sequências de símbolos combinados por meio de operadores. Isso permite a codificação de sentenças declarativas de forma compacta, mesmo que as sentenças – ou suas combinações – sejam elaboradas, além de permitir o foco na mecânica da argumentação. A Lógica Matemática, desenvolvida por Boole, é dita o estudo das *formas de argumentos*, pois o que interessa no estudo da validade é a *estrutura* do argumento, e não seu conteúdo.

Sentenças declarativas atômicas, ou indecomponíveis, são denominadas *proposições*. A cada uma dessas sentenças atômicas é atribuído um símbolo p, q, r, \dots ou às vezes p_1, p_2, p_3, \dots , denominado *símbolo proposicional* ou, com certo abuso de notação em que o símbolo sintático é mesclado ao seu significado, *proposição*. A partir desses elementos

básicos, podemos codificar sentenças mais complexas de forma composicional, de acordo com as regras abaixo:

\neg : a negação de p é denotada por $\neg p$.

\vee : dados p e r podemos querer afirmar que pelo menos uma delas é verdadeira. Essa sentença declarativa é denotada por $p \vee r$ e é chamada de disjunção de p e r .

\wedge : a fórmula $p \wedge r$ denota a conjunção de p e r , quando se quer afirmar que ambas são verdadeiras.

\rightarrow : para expressar uma implicação entre p e q , sugerindo que q é uma consequência lógica de p , utiliza-se $p \rightarrow q$, onde p é a suposição e q é a conclusão.

\leftrightarrow : a declaração $p \leftrightarrow r$, chamada de bicondicional, expressa que p é verdade se e somente se r é verdade.

Uma sentença de uma lógica é denominada *fórmula bem formada* ou, simplesmente, *fórmula*. Fórmulas mais complexas podem ser elaboradas com a repetição das regras anteriores, como:

$$p \wedge q \rightarrow \neg r \vee q$$

que significa “se p e q então não r ou q ”. Apesar de existirem convenções quanto à associação dos elementos, a leitura dessa forma possui uma potencial ambiguidade. Um computador exigiria a inserção de parênteses, como em

$$(p \wedge q) \rightarrow ((\neg r) \vee q).$$

ou então a definição explícita das regras de associação dos operadores.

A Lógica Proposicional é capaz de representar componentes de sentenças sem estrutura interna que usem os termos *não*, *e*, *ou*, *se... então*, que podem ser traduzidos em operadores. Contudo, possui limitações, visto que o aspecto lógico das linguagens naturais e artificiais vai além disso. Para expressar quantidades como *existem ...*, *todos ...*, *alguns ...*, *apenas ...*, novos operadores precisam ser inseridos. Isso cria a necessidade da utilização de uma linguagem com maior poder de expressão, como a *Lógica de Predicados*, também chamada de *Lógica de Primeira Ordem*.

As sentenças atômicas da Lógica de Predicados têm o formato $P(x, y, z, \dots)$, onde P é o nome do *predicado* e x, y, z, \dots são *argumentos*, que podem ser *variáveis*, *constantes*, outros *predicados* ou *chamadas de função*. Os predicados representam relações entre

elementos de um conjunto, enquanto as variáveis representam objetos ou valores concretos. A essa estrutura, são introduzidos os mesmos operadores da Lógica Proposicional, acrescido dos quantificadores universal e existencial, abaixo descritos:

\forall : Lido como “para todo”, o quantificador universal indica que uma afirmação é verdadeira para todas as coisas.

\exists : Lido como “existe algum” ou “para alguns”, o quantificador existencial indica que uma afirmação é verdadeira para pelo menos um elemento do domínio.

Esses quantificadores sempre vêm associados a uma variável e, em conjunto com as afirmações, pode expressar sentenças como

$$\forall x(P(x) \rightarrow S(x))$$

que poderia ser lido em linguagem natural como “para todo x , se x é P , então x é S ” ou, simplesmente, “todo S é P ”.

Lógicas de ordens superiores possuem um poder ainda maior de representação. Contudo, muitos dos problemas associados a elas são indecidíveis. Na própria Lógica de Primeira Ordem, não é possível determinar algoritmicamente a existência de consequências lógicas. Sistemas de prova para a lógica de primeira ordem podem ser consistentes mas não podem ser completos. Por essa razão, seu uso na verificação formal das propriedades de sistemas é mais restrito. Em geral, na medida do poder de expressão de uma linguagem cresce, maior é a quantidade de problemas indecidíveis associados a ela.

3.5 Dedução Natural

A Dedução Natural é um sistema de prova que permite extrair conclusões a partir de um conjunto de premissas, que são fórmulas de uma lógica construídas em acordo com as regras de formação da linguagem. Isso nos permite inferir fórmulas a partir de outras fórmulas, por meio de regras denominadas *regras de inferência*. Ao aplicar essas regras em sucessão, podemos inferir uma conclusão a partir de um conjunto de premissas. Essa conclusão obtida pode servir como premissa na obtenção de novas conclusões e assim sucessivamente (HUTH; RYAN, 2004).

A partir de um conjunto de fórmulas $\phi_1, \phi_2, \phi_3, \dots, \phi_n$, chamadas de premissas, e uma outra fórmula, ψ , chamada de conclusão, podemos aplicar uma série de regras de inferência às premissas para chegar a outras fórmulas. Ao aplicar mais regras de pro-

vas a essas fórmulas, podemos eventualmente obter a conclusão. Esse procedimento é formalmente denotado por

$$\phi_1, \phi_2, \dots, \phi_n \vdash \psi \quad (1)$$

A expressão apresentada na Equação 1 é chamada de *sequente* e é verdadeira se e somente se uma prova para ela puder ser encontrada.

A Dedução Natural é um sistema de prova que consiste em um conjunto de regras para a introdução e eliminação de operadores lógicos e quantificadores (no caso da lógica de predicados). As regras de Dedução Natural na Lógica Proposicional são mostradas na Figura 3. Na ordem mostrada na figura, as regras são introdução da conjunção, eliminação da conjunção, introdução da disjunção, eliminação da disjunção, introdução da implicação, eliminação da implicação, introdução de negação e eliminação da dupla negação.

Figura 3 – Regras básicas de dedução natural para lógica proposicional

\wedge		\vee		\rightarrow	\neg
$\frac{\phi \quad \psi}{\phi \wedge \psi} \wedge_i$		$\frac{\phi}{\phi \vee \psi} \vee_{i1}$	$\frac{\psi}{\phi \vee \psi} \vee_{i2}$	$\frac{\begin{array}{ c } \phi \\ \vdots \\ \psi \end{array}}{\phi \rightarrow \psi} \rightarrow_i$	$\frac{\begin{array}{ c } \phi \\ \vdots \\ \perp \end{array}}{\neg \phi} \neg_i$
$\frac{\phi \wedge \psi}{\phi} \wedge_{e1}$	$\frac{\phi \wedge \psi}{\psi} \wedge_{e2}$	$\frac{\phi \vee \psi \quad \begin{array}{ c } \phi \\ \vdots \\ \chi \end{array} \quad \begin{array}{ c } \psi \\ \vdots \\ \chi \end{array}}{\chi} \vee_e$		$\frac{\phi \quad \phi \rightarrow \psi}{\psi} \rightarrow_e$	$\frac{\neg \neg \phi}{\phi} \neg_{\neg e}$

Fonte: Adaptado de Huth e Ryan (2004)

Já as regras de Dedução Natural na Lógica de Predicados, mostradas na Figura 4, são uma extensão às anteriores, incluindo os quantificadores existencial e universal. Pela ordem mostrada na figura, as regras são a eliminação do operador universal, introdução do operador universal, introdução do operador existencial e eliminação do operador existencial. Nesta figura, o termo $\phi[t/x]$ significa a substituição de todas as ocorrências livres da variável x na fórmula ϕ pelo termo t . Um termo pode ser qualquer fórmula da Lógica de Predicados.

Uma *prova* é uma sequência de fórmulas obtidas a partir da aplicação de regras

Figura 4 – Regras de dedução natural para lógica de predicados

$$\begin{array}{c}
 \frac{\forall x \phi}{\phi[t/x]} \forall x e \\
 \frac{\boxed{\begin{array}{c} x_0 \\ \vdots \\ \phi[x_0/x] \end{array}}}{\forall x \phi} \forall x i \\
 \frac{\phi[t/x]}{\exists x \phi} \exists x i \\
 \frac{\exists x \phi \quad \boxed{\begin{array}{c} x_0 \quad \phi[x_0/x] \\ \vdots \\ \chi \end{array}}}{\chi} \exists x e
 \end{array}$$

Fonte: Adaptado de Huth e Ryan (2004)

de inferência nas fórmulas já existentes na sequência e usualmente é escrita no formato de uma tabela, com linhas numeradas em que a regra de inferência aplicada é apresentada com as linhas que foram usadas em sua aplicação. As primeiras fórmulas são as premissas do argumento e a última fórmula deve ser sua conclusão. O exemplo abaixo ilustra a prova da transitividade do operador de implicação, representada pelo sequente $p \rightarrow q, q \rightarrow r \vdash p \rightarrow r$.

Linha	Fórmula	Regra aplicada
1.	$p \rightarrow q$	Premissa
2.	$q \rightarrow r$	Premissa
3.	$ p$	Hipótese
4.	$ q$	$\rightarrow e$ 1,3
5.	$ r$	$\rightarrow e$ 2,4
6.	$p \rightarrow r$	$\rightarrow i$ 3-5

No exemplo de prova acima, a conclusão é obtida pela regra da introdução da implicação (que usa raciocínio hipotético) na hipótese que teve início na linha 3 e término na linha 5; por sua vez, a conclusão intermediária obtida na linha 5 é resultado da aplicação de uma regra de eliminação da implicação, nas linhas 2 e 4; a fórmula da linha 4 foi obtida pela aplicação da regra nas linhas 1 e 3.

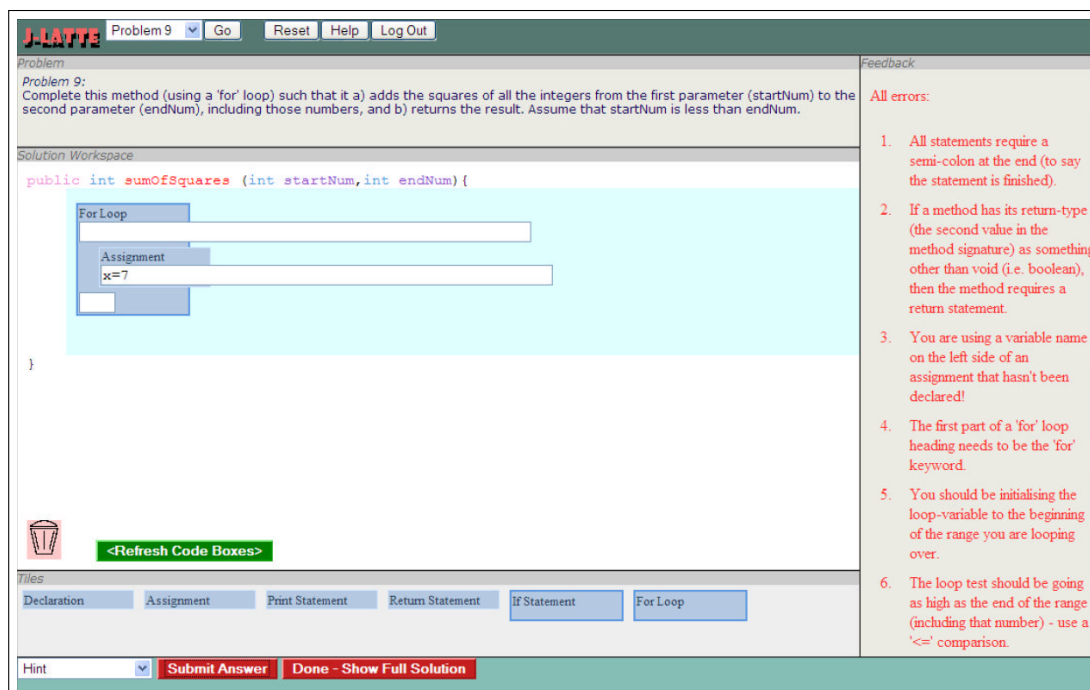
os erros no momento em que são feitos com o objetivo de o estudante poder corrigi-los e evitar desperdício de tempo revisando código após todo o programa ter sido escrito. Sempre que o estudante solicita ajuda ou comete um erro de planejamento ou codificação, o tutor fornece informações sobre o tópico em questão. Utiliza o método *model-tracing* para analisar o raciocínio do estudante conforme ele escreve a solução do problema, identificando qual conceito correto ou equívoco levou à inserção daquela parte de código. Enquanto as regras estão corretas, o sistema age como um simples editor estruturado, mas quando identifica que o estudante necessita, interrompe a resolução oferecendo a assistência relevante. Este sistema atinge um conjunto de objetivos pedagógicos derivados da teoria de aprendizagem ACT*, de Anderson (1983).

O sistema tutor inteligente J-LATTE, apresentado por Holland, Mitrovic e Martin (2009), utiliza a metodologia baseada em restrições para ensinar conceitos simples da linguagem de programação Java. Divide as tarefas entre o modo conceitual, onde o algoritmo é planejado, e o modo de codificação, onde é escrito. Em sua interface, apresenta no topo a descrição do problema atual, na barra da direita as mensagens de *feedback*, na parte central a solução que está sendo montada e na parte inferior as opções disponíveis, como mostra a Figura 6. O sistema de *feedback* é sob demanda, somente fornecido quando solicitado pelo estudante, e mostra as mensagens em vermelho caso ele tenha cometido erros. A metodologia consiste em avaliar se a resposta inserida pelo estudante atende a diversas restrições específicas definidas no modelo do problema.

O *Java Intelligent Tutoring System* (JITS), de Sykes e Franek (2003), possui uma arquitetura dividida em duas funcionalidades. Na primeira, utilizada para problemas mais simples, o professor insere um conjunto de problemas, suas especificações e soluções completas. Na segunda, o professor elabora apenas o problema, a especificação e a saída (resultado esperado), de modo que não há uma solução registrada em código que o sistema possa comparar com a solução do estudante. Dessa forma, a solução do estudante é analisada por um algoritmo especializado em determinar a submissão mais provável do código que ele pretendia. Cada vez que esse algoritmo detecta uma sequência de código não compilável, utiliza um conjunto de transformações para inferir qual a solução válida mais próxima daquela que o usuário inseriu. Assim, é capaz de sugerir as correções necessárias.

O trabalho de Mitrovic (2003) apresenta um sistema tutor inteligente para *web*, o SQLT-Web. O sistema foi desenvolvido baseado em uma versão anterior para *desktop*, o SQL-Tutor, e aponta vantagens da nova plataforma, como facilidade na distribuição e

Figura 6 – Interface do sistema tutor J-LATTE



Fonte: Holland, Mitrovic e Martin (2009)

manutenção. A arquitetura do sistema está organizada entre a interface, o módulo pedagógico, que decide o conteúdo e a frequência de *feedback*, e o módulo de estudante, que analisa as respostas do usuário. Contém uma base de dados com um conjunto de problemas e suas soluções ideais. O sistema não possui modelo de domínio, então para fazer a correção das soluções utiliza um domínio de conhecimento representado na forma de restrições. Utiliza o método de modelagem baseada em restrições para traçar o conhecimento adquirido dos estudantes, onde compara as soluções fornecidas com um conjunto de regras pré-definidas no sistema. Este sistema também elabora uma sequência de aprendizado, onde apresenta os problemas em ordem personalizada de acordo com as ações do estudante e a complexidade dos problemas.

Outro trabalho apresenta uma proposta de suporte ao ensino-aprendizagem de dedução natural. O sistema Heráclito, desenvolvido por Penteadó e Gluz (2011), possui o recurso de editor de provas, como a que será implementada nesse projeto. Contudo, a ferramenta possui baixa usabilidade e pouca documentação, de modo que suas funções mais básicas não podem ser executadas de maneira intuitiva, além de não estar disponível publicamente. Adicionalmente, não há qualquer estrutura de *feedback* para quem está operando o sistema, ponto que se pretende aprimorar com o desenvolvimento deste trabalho.

A Tabela 1 mostra um comparativo com as principais características dos trabalhos correlatos. As características destacadas têm o objetivo comparar (i) a atividade proposta ao estudante, (ii) o método de representação do domínio, (iii) o método de representação dos conhecimentos e (iv) a frequência de *feedback*. Em alguns dos artigos citados, contudo, nem todas essas características estão disponíveis.

Tabela 1 – Comparação dos trabalhos correlatos

Título	Autor	Conteúdo	Características
FITS for C++ Programming	Hooshyar <i>et al.</i> (2016)	C++	Grafo acíclico de conceitos, redes bayesianas, auxílio e <i>feedback</i> personalizados
Lisp-Tutor	Reiser, Anderson e Farrell (1985)	LISP	<i>Feedback</i> imediato, <i>model-tracing</i>
J-Latte	Holland, Mitrovic e Martin (2009)	Java	Baseado em restrições, 2 modos (planejamento e codificação), <i>feedback</i> sob demanda
JITS	Sykes e Franek (2003)	Java	Reconhecimento de intenção do estudante, problemas simples possuem soluções completas
SQL-Tutor	Mitrovic (2003)	SQL	Baseado em restrições, sistema <i>web</i> , decide a sequência de conteúdos para o usuário
Heráclito	Penteado e Gluz (2011)	Dedução natural	Aborda apenas a lógica proposicional

Fonte: Autor (2019)

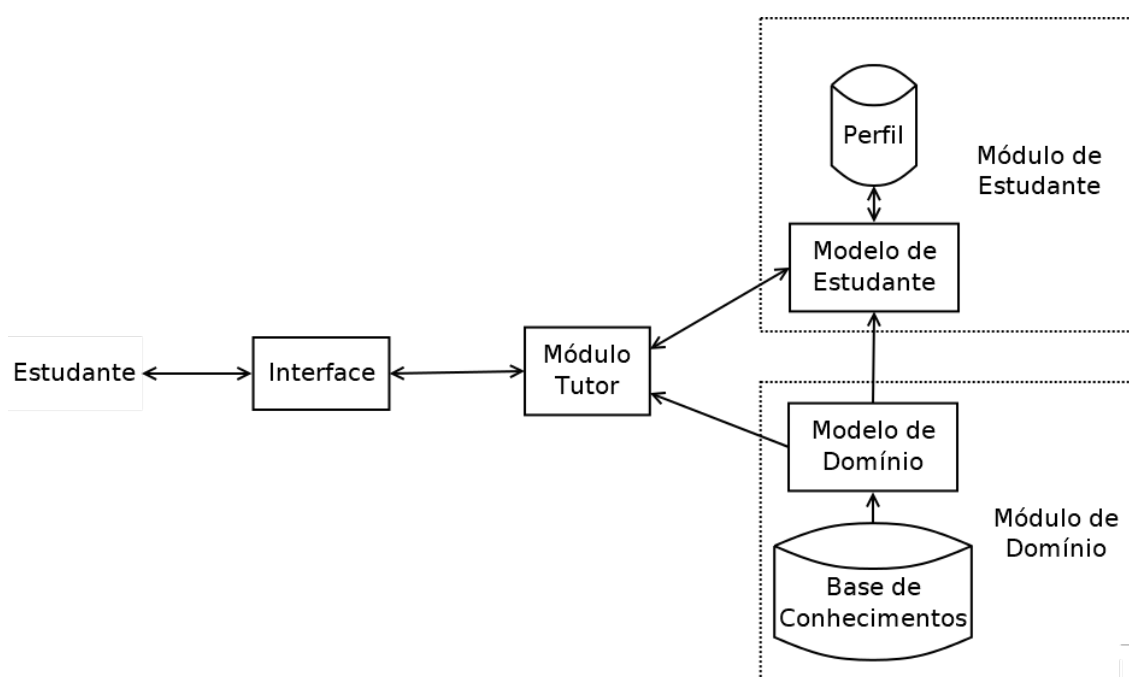
Não foi encontrado na literatura um sistema tutor inteligente para o conteúdo de Dedução Natural que possua descrição detalhada do aspecto técnico de sua implementação e de sua metodologia pedagógica. Em comparação aos trabalhos existentes, este projeto propõe a implementação de um sistema tutor inteligente com modelagem baseada em restrições, *feedback* imediato durante a resolução das atividades e que apresente exercícios de prova de Dedução Natural.

5 DESENVOLVIMENTO

5.1 Descrição da proposta

A arquitetura construída a partir do referencial bibliográfico estudado é apresentada nesta seção. A estrutura do sistema é dividida em três módulos principais: Módulo de Estudante, Módulo de Domínio e Módulo Tutor. A Figura 7 mostra a arquitetura geral e relação entre os módulos.

Figura 7 – Diagrama da arquitetura do sistema



Fonte: Autor (2019)

O diagrama mostra que o estudante interage diretamente com o sistema através da interface, inserindo e recebendo informações. O Módulo Tutor é o responsável por ler os dados inseridos através da interface, bem como atualizá-la com as informações provenientes dos demais módulos. Além disso, este módulo lê informações do Módulo de Estudante para personalização do conteúdo de ensino, bem como atualiza este módulo conforme coleta novas informações. O Módulo de Estudante armazena e recupera as informações do Perfil do aluno, enquanto também recebe a estrutura do Módulo de Domínio para associar às informações de conhecimento do usuário. Este Módulo de Domínio também é responsável por recuperar as informações da Base de Conhecimentos (banco com conteúdos, questões e soluções) e transmitir ao Módulo Tutor.

Conforme especificado na Seção 3.1, existem dois tipos de modelagem dos conhecimentos do estudante em sistemas tutores inteligentes. O método escolhido para esse sistema foi a modelagem baseada em restrições, devido ao fato de as regras de inferência da Dedução Natural serem facilmente implementadas com base em suas restrições de aplicação, tornando-as ideias para este tipo de aplicação. Mais informações sobre as restrições das regras de inferência serão apresentadas na Seção 5.4.

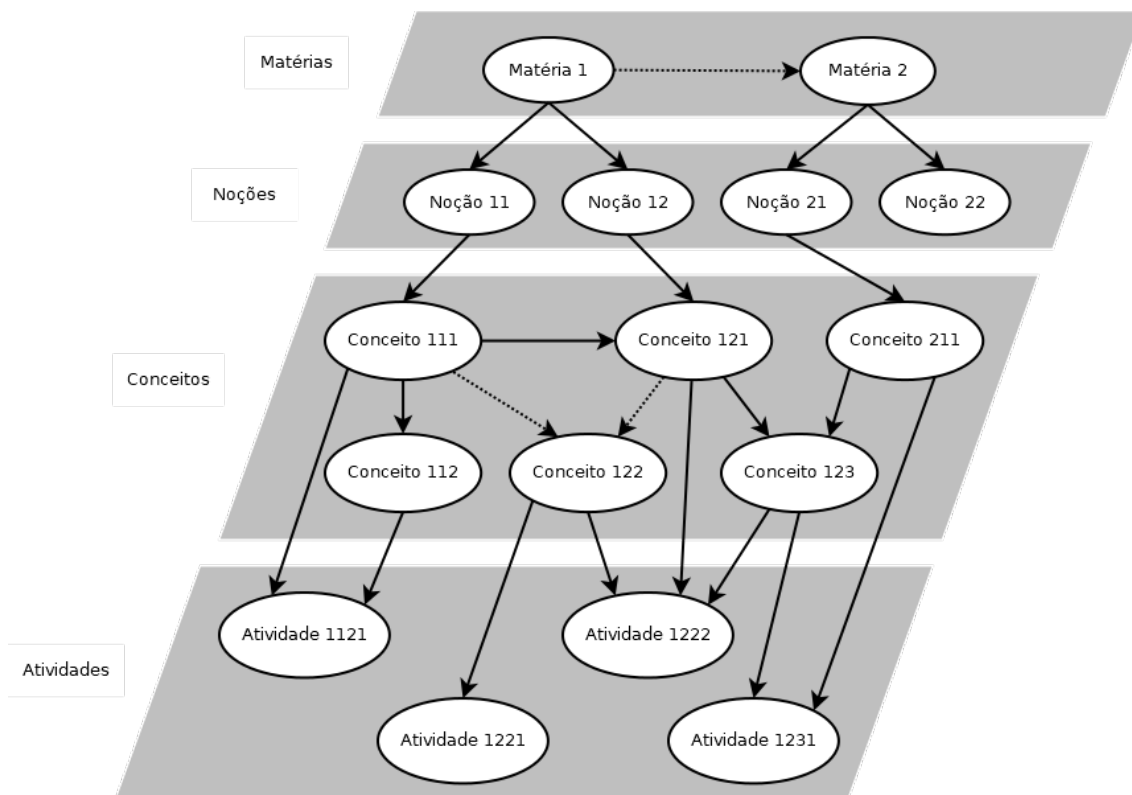
O *Módulo de Estudante* é responsável por armazenar todas as informações, conhecimentos e interações relativos ao usuário do sistema, constituindo o que é chamado de “perfil”. Este perfil pode ser exportado, gerando um arquivo com extensão reconhecível pelo programa e tornando possível que seja importado e acessado em outros computadores com a aplicação instalada. Os conhecimentos são armazenados como um índice para cada conceito, noção e matéria do domínio, baseado nas condições certas e erradas que eles alcançam durante a resolução das atividades. Essa parte do módulo armazena o estado do estudante no curso, incluindo o caminho de aprendizado seguido, a relação de atividades que foram concluídas, os conceitos que foram aprendidos e os que ainda estão a aprender. As interações do usuário durante a resolução das atividades também são armazenadas nesse modelo. Regras aplicadas e removidas, pedidos de ajuda e *feedbacks* recebidos são algumas das informações monitoradas.

O *Módulo de Domínio* permite a organização do campo de ensino e decomposição em elementos. Através de uma representação esquemática, descreve as classes de conhecimento que constituem o domínio. A Figura 8 mostra um exemplo dessa representação. O domínio é constituído por: (i) uma estrutura hierárquica, (ii) as relações entre os itens dessa estrutura, (iii) o material de apoio para cada instância e (iv) as atividades. A estrutura do domínio é dividida em níveis, que se organizam da seguinte forma:

- Uma matéria é constituída por uma ou mais noções.
- Uma noção é constituída por um ou mais conceitos.
- Um conceito é constituído por uma ou mais atividades. Cada conceito está associado também a um ou mais conteúdos ou materiais de apoio.
- As atividades definem a forma como os exercícios são apresentados.
- As instâncias podem estar ligadas a outras do mesmo nível, indicando sequência, dependência ou semelhança. O caminho formado pelas ligações é chamado de sequência pedagógica.

Baseado nesta estrutura, o professor pode realizar a montagem do domínio. No

Figura 8 – Diagrama do Módulo de Domínio



Fonte: Autor (2019)

contexto da dedução natural na lógica proposicional, por exemplo, a matéria é a *lógica proposicional*. As noções são as operações existentes sobre fórmulas ϕ , onde ϕ pode ser uma fórmula composta ou uma proposição. A tabela abaixo apresenta as noções existentes sobre operadores existentes no sistema, com notação prefixa.

$\wedge(\phi_1, \phi_2)$	–	operação de conjunção
$\vee(\phi_1, \phi_2)$	–	operação de disjunção
$\rightarrow(\phi_1, \phi_2)$	–	operação de implicação
$\leftrightarrow(\phi_1, \phi_2)$	–	operação de equivalência
$\neg(\phi_1)$	–	operação de negação

Os conceitos, por sua vez, são as regras aplicadas às fórmulas. A tabela abaixo apresenta os conceitos que devem ser aprendidos pelos alunos, que são as regras de inferência da Dedução Natural.

$\wedge i(\phi_1, \phi_2)$	–	introdução da conjunção
$\wedge e_1(\phi_1, \phi_2)$	–	eliminação da conjunção 1
$\wedge e_2(\phi_1, \phi_2)$	–	eliminação da conjunção 2
$\vee i(\phi_1, \phi_2)$	–	introdução da disjunção
$\vee e(\phi_1, \phi_2, \phi_3)$	–	eliminação da disjunção
$\rightarrow i(\phi_1)$	–	introdução da implicação
$\rightarrow e(\phi_1, \phi_2)$	–	eliminação da implicação
$\neg i(\phi_1)$	–	introdução da negação
$\neg e(\phi_1)$	–	eliminação da dupla negação

As atividades ou exercícios são compostos por: (i) as fórmulas (premissas e conclusão), (ii) a lista de conceitos aos quais estão associados, ou seja, as regras que são empregadas em sua resolução, (iii) a complexidade para cada conceito abordado, (iv) as soluções esperadas, na forma de sequências de regras, e (v) as restrições. As restrições são utilizadas pelo módulo tutor para fornecer o *feedback* adequado. O material de apoio pode ser associado aos itens de diferentes níveis da estrutura do domínio. Isso quer dizer que é possível associar um material de apoio a uma matéria, uma noção e/ou um conceito. O material é composto por uma página HTML, podendo conter texto, imagens, áudio e/ou vídeo.

O *Módulo Tutor* é responsável por avaliar as ações e soluções do estudante, decidindo qual o momento ideal para intervir oferecendo ajuda e qual o conteúdo mais adequado para a ajuda. Além disso, é responsável por selecionar o sequenciamento de currículo e de atividades. O processo de tutoria ocorre durante a resolução de exercícios pelo usuário. O tutor analisa as interações dele com o sistema, registrando interações como a quantidade de regras aplicadas, o número de vezes que uma ação foi desfeita, as ajudas recebidas e os pedidos de ajuda. Baseado nisso, avalia o desempenho do estudante e atribui uma nota àquela atividade, que irá compor a nota do conceito. O fornecimento de *feedback* está associado à matéria, noção, conceito ou atividade. De acordo com as condições de satisfação específicas de cada mensagem, elas são ativadas e mostradas ao estudante. O sequenciamento de currículo busca atender os objetivos cognitivos da teoria de aprendizagem significativa e do *design instrucional*, começando com elementos simples e, gradualmente, incrementando a dificuldade para elementos que interagem entre si. Quando esta interação de elementos é entendida, passa a ser tratada pelo cérebro como um novo elemento simples e pode interagir com novos elementos de complexidade superior. Assim, os exercícios são iniciados pelos mais simples, de modo a fixar o conteúdo

para o estudante, passando a exercícios mais elaborados que exigem a interação dos conceitos abordados anteriormente. Para a estruturação do sequenciamento de currículo, os conceitos ou tópicos a serem dominados pelo estudante possuem relações entre si. Essas relações possuem o atributo de tipo, que pode ser pré-requisito ou associação. O tipo *pré-requisito* é utilizado para a sequência de tópicos, significando que em caso de dependência alta, o estudante só avança para o próximo tópico após obter um bom desempenho neste, e para dependência baixa ele pode avançar obtendo um desempenho moderado. O tipo *associação* é utilizado como segunda opção, para que nos casos onde o estudante está com dificuldades em um tópico seja possível oferecer um tópico semelhante, para então voltar ao tópico anterior com mais embasamento. Duas informações sobre a estrutura do domínio são atribuídas a cada atividade: (i) a lista de conceitos à qual está relacionada e (ii) o grau de complexidade da aplicação de cada tópico nesse exercício. O sequenciamento de atividades ocorre considerando o desempenho já registrado do estudante e a complexidade dos problemas.

A *interface principal* do sistema é o meio pelo qual o estudante interage com os demais módulos. Através dela são apresentados os exercícios e materiais de apoio. Ela é composta por uma área do problema, área de solução e área de *feedback*. As telas secundárias são de apresentação do material de apoio e opções adicionais. A Figura 9 mostra o esboço da interface principal proposta para o sistema, com a disposição na tela dos elementos descritos anteriormente. Esse esboço foi utilizado como base para o desenvolvimento da interface gráfica da aplicação nas etapas posteriores. O material de apoio é apresentado na introdução de cada novo conceito, conforme o modelo de *design instrucional 4C/ID* sugere. O sistema apresenta ao estudante exercícios de Dedução Natural, sequencialmente. Ao introduzir um conceito novo, é apresentado o material de apoio relativo àquele tópico. A sequência de exercícios é ajustada de acordo com as interações do estudante. O sistema apresenta *feedback* aos estudantes conforme necessário, isto é, se estão cometendo um erro ou com dificuldade na resolução do exercício. Ele observa as ações do estudante para identificar suas necessidades. Durante a resolução do exercício, o estudante pode solicitar ajuda na forma de *feedback* ou revisar os conceitos do material de apoio já vistos. Ao iniciar a sessão, o sistema seleciona um problema para o estudante trabalhar. Quando o estudante insere uma solução, o módulo tutor a analisa, identifica erros (se houver), e atualiza o modelo de estudante de acordo com isso. Baseado no modelo de estudante, o módulo tutor gera uma ação pedagógica (*feedback*) apropriada. Quando o problema atual é resolvido ou o estudante solicita outro problema para trabalhar, o módulo

tutor seleciona um novo problema apropriado para o nível do estudante.

Figura 9 – Esboço da proposta de interface

Atividade

$$p \rightarrow q, r \rightarrow s \mid - (p * r) \rightarrow (q * s)$$

1. $p \rightarrow q$	Premissa
2. $r \rightarrow s$	Premissa
3. $\mid p * r$	Hipótese
4. $\mid p$	*e, 3
5. $\mid r$	*e, 3
6. $\mid q$	\rightarrow e, 1, 4
7. $\mid s$	\rightarrow e, 2, 5
8. $\mid q * s$	*i, 6, 7
9. $(p * q) \rightarrow (q * s)$	\rightarrow i, 3-8

Regras de inferência

Feedback

- Feedback A: você deve fazer assim.
- Feedback B: que tal tentar a regra N?
- Você pode pedir uma dica!
- Nova mensagem de feedback!

Fonte: Autor (2019)

5.2 Requisitos, casos de uso e diagrama de sequência

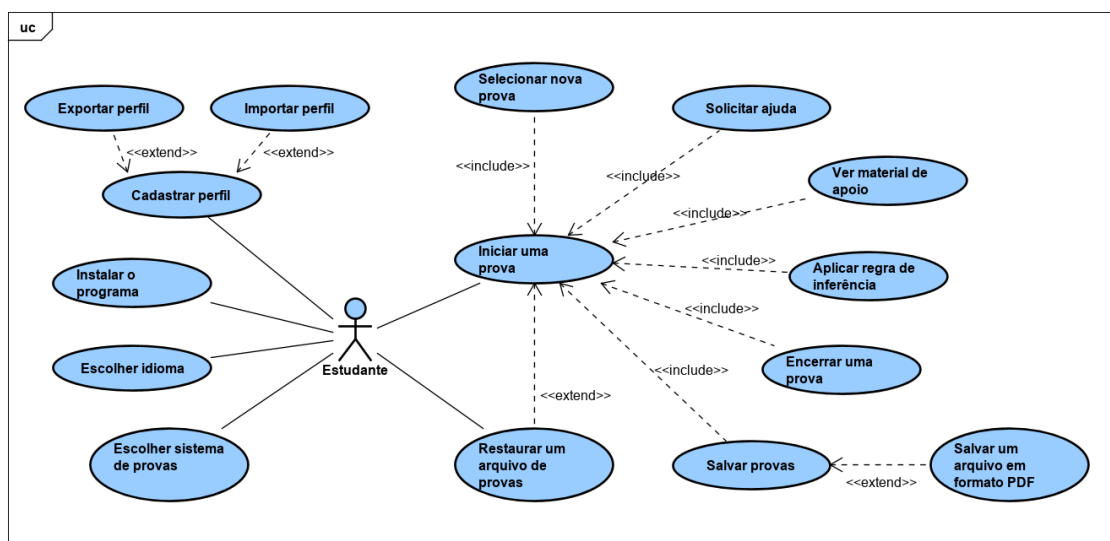
No apêndice A são apresentados os requisitos do sistema, oferecendo uma visão geral da proposta. Os requisitos estão classificados entre funcionais e não funcionais. Os *requisitos funcionais* são aqueles ligados diretamente a funcionalidades do sistema, sejam elas disponíveis diretamente ao usuário ou entre os próprios módulos da arquitetura. Já os *requisitos não funcionais* são relacionados ao ambiente onde o sistema é executado e suas condições de desempenho.

A partir do levantamento dos requisitos, foram modelados os casos de uso do sistema. Nessa seção são apresentados os diagramas de casos de uso conforme o padrão

da UML. Os casos de uso estão detalhados no Apêndice B.

A Figura 10 apresenta o diagrama com os casos de uso do estudante. Nesse diagrama, as relações do tipo *include* significam que o caso de uso depende do anterior, enquanto as relações do tipo *extend* representam que o caso de uso pode acontecer ou não após a ativação do caso que é estendido. O diagrama mostra que o estudante possui as funções principais de instalar o programa, escolher o idioma do sistema, cadastrar seu perfil, escolher sistema de provas que será utilizado, iniciar uma nova prova e restaurar um arquivo de provas. A partir da função do cadastro de perfil, existem as opções de exportar e importar o perfil de usuário. Ao iniciar uma prova, o usuário tem acesso às diversas opções durante a execução do exercício, que são selecionar uma nova prova, solicitar ajuda na resolução, ver o material de apoio referente ao tópico do exercício, aplicar regra de inferência, encerrar a prova e salvar a prova ou exportá-la em formato PDF.

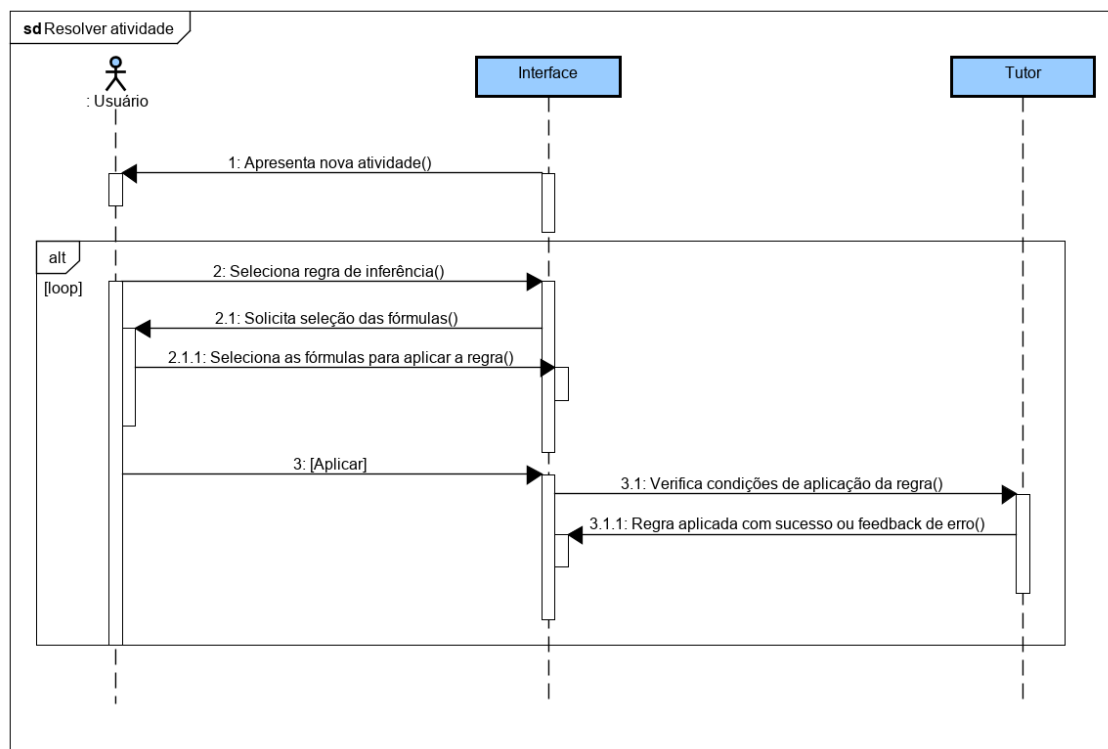
Figura 10 – Diagrama de casos de uso do estudante



Fonte: Autor (2019)

O diagrama de sequência, mostrado na Figura 11, ilustra a sequência de ações envolvidas na resolução de uma atividade pelo usuário. Após a atividade ser apresentada na interface, o usuário utiliza as regras de inferência algumas vezes até chegar à resposta final do exercício. No momento da aplicação de cada regra, o sistema verifica se alguma restrição foi quebrada com as fórmulas selecionadas. Em caso negativo, a regra é aplicada e o resultado mostrado na interface. Em caso positivo, uma mensagem de *feedback* é exibida com explicações sobre a violação. A seção 5.4 descreve de forma detalhada o funcionamento da resolução de exercícios no sistema.

Figura 11 – Diagrama sequência da resolução de atividades



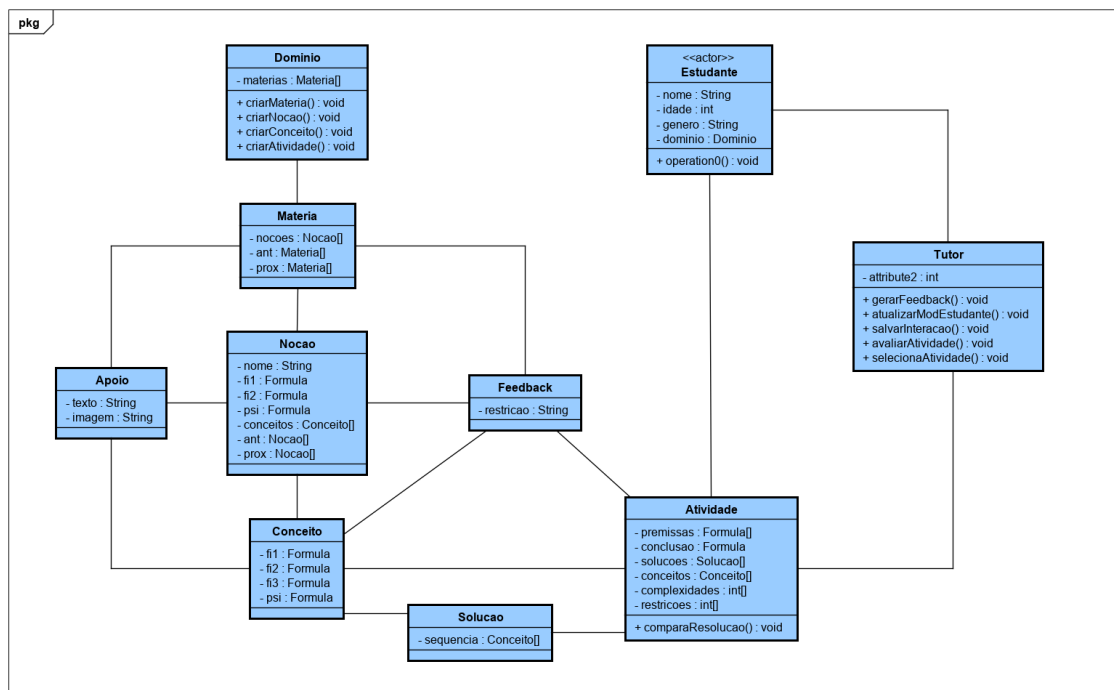
Fonte: Autor (2019)

5.3 Arquitetura do sistema

O diagrama de classes mostrado na Figura 12 apresenta as classes principais do sistema, com as ligações entre classes que interagem entre si, além de seus atributos básicos e métodos. O diagrama apresentado mostra as classes Domínio, Matéria, Noção, Conceito e Atividade, que correspondem à hierarquia do domínio. Além dessas classe, existe a classe Feedback, cujas instâncias estão associadas também a estes níveis, bem como a classe Apoio, também associada aos níveis superiores a atividades. A classe Solução está associada a uma atividade e sua instância consiste em uma sequência de conceitos aplicados às fórmulas daquela atividade. As demais classes são Estudante, que contém as informações do usuário, e Tutor, que agrupa as funcionalidades do módulo com o mesmo nome.

Para o armazenamento consistente das informações do sistema é necessária a utilização de um banco de dados. O diagrama entidade-relacionamento na Figura 13 mostra a estrutura do banco de dados do sistema. As entidades, representadas por retângulos, são relacionadas aos objetos manipulados na estrutura do programa e correspondem a uma

Figura 12 – Diagrama de classes



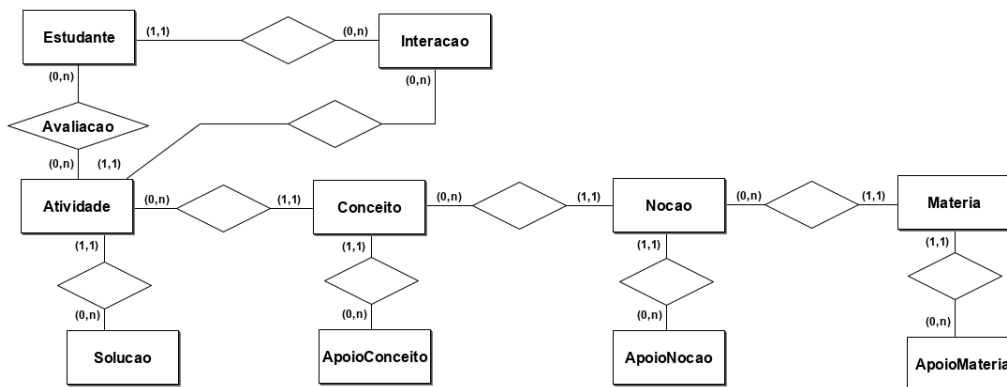
Fonte: Autor (2019)

tabela no banco de dados. As classes armazenadas são Conceito, Nocao e Materia e seus respectivos materiais de apoio (ApoioConceito, ApoioNocao e ApoioMateria), Atividade e suas respectivas soluções (Solucao), além do perfil do usuário em Estudante e suas interações com o sistema na tabela Interacao. A cardinalidade $(0, n)$ para $(1, 1)$ indica que a primeira entidade está associada a um item da segunda, enquanto a segunda pode estar associada a vários itens da primeira. Já a cardinalidade $(0, n)$ para $(0, n)$ indica que um item da primeira entidade pode estar associado a vários itens da segunda e vice-versa, necessitando a criação de uma tabela de relacionamento, que no caso deste diagrama ocorre na tabela Avaliacao. Esta classe armazena a nota do estudante em cada uma das atividades propostas pelo sistema. Os atributos, chaves primárias e chaves estrangeiras foram abstraídos do diagrama para gerar uma representação mais generalista e de fácil entendimento.

5.4 Resolução de exercícios

Para implementar a principal funcionalidade do sistema – a resolução de exercícios de Dedução Natural – foi necessário o conhecimento aprofundado das regras de inferência

Figura 13 – Diagrama entidade-relacionamento



Fonte: Autor (2019)

desse sistema de provas. A implementação das regras em uma linguagem de programação exige a contemplação de todas as restrições e exceções envolvidas em suas aplicações. Cada uma das regras possui condições próprias em que pode ou não ser aplicada. A maioria das regras deve ser aplicada sobre duas fórmulas, porém em alguns casos são utilizadas uma ou três fórmulas.

Em suma, um exercício de Dedução Natural oferece um conjunto de premissas e uma conclusão, que deve ser obtida por meio de uma prova. O estudante deve chegar até a conclusão aplicando uma série de regras de inferência nas premissas e outras fórmulas derivadas das premissas. Na interface, o conjunto de regras de inferência é exibido na forma de botões. A resolução do exercício é implementada com um componente de tabela, onde a primeira coluna corresponde ao número da fórmula, a segunda coluna corresponde ao conteúdo da fórmula e a terceira coluna corresponde ao tipo de fórmula, podendo ser premissa, hipótese ou uma das regras. Inicialmente, a atividade carregada, cujo enunciado é mostrado em um *label* acima da tabela, tem suas premissas transformadas automaticamente nas primeiras linhas da tabela. O usuário deve selecionar uma das regras e escolher as linhas da tabela – de uma a três, dependendo da regra – em que a deseja aplicar. Nesse momento, é observado se as linhas selecionadas não violam as restrições da regra selecionada. As fórmulas, que na interface do sistema são representadas por uma *String* em sua notação infixa, internamente são convertidas em uma notação prefixa e, então, em seu formato de árvore sintática, para que sejam executadas as operações sobre elas mais facilmente. Na Tabela 2 são especificadas as restrições para cada regra.

Alguns métodos auxiliares foram implementados para lidar com ações relacionadas à interface da resolução dos exercícios. O método *selecionar fórmulas* ativa a seleção

Tabela 2 – Restrições aplicadas às regras de inferência

Regra	Nº de fórmulas	Restrições
\wedge_i	2	–
\wedge_e	1	A fórmula selecionada deve ser uma conjunção.
\vee_i	1	–
\vee_e	3	Uma das fórmulas deve ser uma disjunção. Duas das fórmulas devem ser implicações. O conseqüente das implicações deve ser o mesmo. Os antecedentes das impl. devem ser as fórmulas da disjunção.
\rightarrow_i	2	A primeira fórmula deve ser o início da última hipótese aberta. A segunda fórmula deve estar na última hipótese aberta.
\rightarrow_e	2	Uma das fórmulas deve ser uma implicação. A segunda fórmula deve ser igual ao antecedente da implicação.
\neg_i	2	As fórmulas devem estar na última hipótese aberta. A primeira fórmula deve ser o início da hipótese. A segunda fórmula deve ser uma contradição.
\neg_e	1	A fórmula selecionada deve ser uma dupla negação.

Fonte: Autor (2019)

das linhas da tabela em que a regra será aplicada. O número de linhas é recebido como argumento do método, visto que pode variar entre regras diferentes. Por sua vez, o método *limpar linhas* limpa as fórmulas atualmente selecionadas, podendo ser utilizado caso o estudante tenha selecionado uma fórmula equivocadamente ou queira alterar as que selecionou. Já o método *nova linha* insere o resultado da regra de inferência aplicada na tabela. Por fim, o método *fechar configuração* é utilizado para voltar ao painel de botões das regras de inferência, ativado automaticamente após adicionar uma nova linha ou por escolha do usuário ao cancelar a aplicação da regra atualmente selecionada.

Além da resolução padrão das atividades propostas pelo sistema, foi desenvolvida a funcionalidade denominada Modo Livre. Esse modo permite ao usuário inserir sua própria atividade, dividida entre premissas e conclusão, e utilizar as funções do sistema para desenvolver sua resolução. Esse recurso foi inserido com o objetivo de ampliar a utilidade do sistema, visto que a estrutura de resolução de exercícios implementada pode ser utilizada para qualquer atividade, não apenas para aquelas inseridas na base de conhecimentos. Dessa forma, qualquer exercício de dedução natural pode ser inserido,

permitindo aos alunos, por exemplo, resolver uma lista de exercícios da disciplina com o auxílio do *feedback* fornecido pelo sistema.

O *feedback* imediato foi implementado com base nas ações do usuário com o sistema. As mensagens de *feedback* são exibidas quando o estudante tenta executar uma ação que viola as restrições da regra de inferência selecionada ou outra funcionalidade do sistema (por exemplo, inserir uma hipótese com caracteres inválidos). Isso ajuda a guiar o aluno e evitar que adquira conhecimentos incorretos, pois ele não consegue aplicar regras onde não deve. O objetivo disso é ajudar o aluno o mais cedo possível, evitando que ele faça ações incorretas e ajudando a ele formar os conceitos de situações corretas em que as regras podem ser aplicadas.

Quando uma restrição é violada, o sistema produz automaticamente uma mensagem de *feedback*, indicando ao estudante por que motivo a regra não pode ser aplicada na situação selecionada. A mensagem simples é mostrada no painel *Feedback* da interface. Ao selecionar o botão *Solicitar ajuda*, o estudante pode receber uma explicação mais detalhada sobre o último *feedback* recebido, contendo informações sobre a restrição violada e a situação correta de utilização desta ação. A Tabela 3 mostra algumas das mensagens de *feedback* cadastradas no sistema e exibidas ao estudante.

Adicionalmente, foram implementadas algumas funções internas para o desenvolvimento do sistema, mas que não foram disponibilizadas na interface final disponível para o usuário. Uma das funções é o cadastro de novas atividades no banco de dados. Para qualquer atividade do Modo Livre, após a conclusão da atividade era exibido um botão com a opção de registrar atividade. Essa funcionalidade registra as premissas e conclusão da atividade, salva a resolução inserida como a solução padrão, bem como calcula automaticamente a complexidade do exercício e registra todas essas informações no banco de dados. Outra das funções internas implementadas foi a visualização do domínio, onde era possível verificar a lista completa de atividades cadastradas e suas soluções, bem como a ordenação de conceitos utilizada pelo sistema de sequenciamento. Para a versão de distribuição do sistema, optou-se por manter apenas as funcionalidades voltadas ao usuário final – o estudante – e por esse motivo as funções descritas não foram incluídas. Assim, a versão de desenvolvimento possui um conjunto pré-definido de atividades.

Tabela 3 – Mensagens de *feedback* exibidas ao estudante

Gatilho	Mensagem
Aplicar regra \vee_e sem selecionar uma disjunção	Para usar \vee_e , uma das fórmulas precisa ser uma disjunção.
Aplicar regra \vee_e sem selecionar duas implicações	É necessário selecionar uma conjunção (\vee) e duas implicações (\rightarrow) para utilizar a regra \vee_e .
Aplicar regra \vee_e onde as implicações não correspondem à disjunção	Não é possível aplicar a eliminação da disjunção (\vee_e) com as regras selecionadas.
Inserir valores inválidos como segunda fórmula da regra \vee_i	O valor informado para a introdução da disjunção (\vee_i) não é válido.
Aplicar regra \neg_i onde a conclusão não é uma contradição	Para aplicar \neg_i , a segunda fórmula precisa ser uma contradição (\perp).
Aplicar regra \neg_e sem selecionar uma dupla negação	A regra \neg_e deve ser aplicada na dupla negação.
Aplicar regra \rightarrow_i sem selecionar fórmulas na última hipótese	Para usar a regra \rightarrow_i , as fórmulas selecionadas devem estar na última hipótese.
Aplicar regra \rightarrow_e sem selecionar uma implicação	Para usar a regra \rightarrow_e , uma das fórmulas precisa ser uma implicação (\rightarrow)!
Aplicar regra \wedge_e sem selecionar uma conjunção	A regra \wedge_e só pode ser aplicada em uma conjunção (\wedge).
Aplicar uma regra em hipótese que já foi encerrada	Não é possível utilizar fórmulas de uma hipótese encerrada.
Selecionar regra \rightarrow_i quando não há hipótese aberta	É preciso iniciar uma hipótese para utilizar a regra \rightarrow_i .
Selecionar regra \neg_i quando não há hipótese aberta	É preciso iniciar uma hipótese para utilizar a regra \neg_i .
Inserir valor inválido para iniciar uma hipótese	O valor informado para a hipótese não é válido.

Fonte: Autor (2019)

5.5 Sequenciamento de currículo

O sequenciamento de currículo tem o objetivo de selecionar atividades para que o estudante resolva, considerando que as atividades selecionadas sejam adequadas ao seu nível de desenvolvimento nos estudos através dos diferentes conceitos abordados. Para tanto, foi desenvolvida uma série de funções do sistema com a finalidade de quantificar o desempenho do estudante de acordo com suas ações no sistema.

Para a implementação do sequenciamento nesse sistema, utilizou-se um algoritmo simples que considera a três elementos: (i) a nota do estudante nas atividades realizadas, (ii) a nota média do estudante em cada um dos conceitos abordados pelo sistema, e (iii) a quantidade de atividades de cada conceito concluídas pelo estudante, de acordo com seu nível de complexidade e dificuldade. Para dimensionar a complexidade dos conceitos em cada atividade, foi utilizado como parâmetro o número de vezes que a regra de inferência equivalente ao conceito é utilizada em sua solução padrão. Assim, por exemplo, se a regra da eliminação da implicação (\rightarrow_e) é utilizada duas vezes na atividade n , é atribuída complexidade 2 a essa relação. Já para quantificar a dificuldade de uma atividade, são utilizados dois fatores: o número de linhas da solução padrão e o número de hipóteses encadeadas presentes nela. Dessa forma, a dificuldade da atividade é definida por $\lceil \text{numLinhas} * (1 + \text{numHipEncadeadas}/2) \rceil$. O número de linhas (numLinhas) considera tanto as premissas quanto todas as hipóteses e regras de inferência na solução padrão da atividade. Já o número de hipóteses encadeadas (numHipEncadeadas) considera o número de vezes que uma hipótese é iniciada enquanto outra hipótese ainda está aberta. Esse foi considerado um fator relevante a ser contabilizado na dificuldade da atividade, visto que o pensamento hipotético exige um nível cognitivo de abstração maior dos estudantes, sendo este fator agravado quando as hipóteses precisam ser encadeadas. A Tabela 4 mostra alguns exemplos de atividades inseridas no sistema, com suas respectivas solução padrão, complexidade para cada conceito e dificuldade.

Para realizar o cálculo da nota do estudante em uma atividade, foram considerados fatores que apontam que o estudante não está tendo facilidade na execução do exercício. Esses fatores, que chamaremos de erros/ajudas, podem ser: solicitar ajuda, consultar material de apoio, desfazer regras aplicadas ou tentar aplicar uma regra de inferência indevidamente. O aluno obtém uma nota alta ao completar o exercício cometendo poucos ou nenhum erro e solicitando pouca ou nenhuma ajuda. Para realizar o cálculo, é utilizado um contador para registrar o número de erros/ajudas do estudante, conforme suas interações durante a realização da atividade. Os três primeiros erros/ajudas são desconsiderados, servindo como margem para erros eventuais ou cometidos sem intenção. A partir do 4º erro/ajuda, é decrementado um valor de 0,5 da nota a cada ocorrência, que tem valor inicial 10, até o limite de zero. Ao concluir a atividade, essa nota é registrada no perfil do aluno. As notas das atividades não são exibidas ao aluno, sendo utilizadas apenas internamente pelo sistema pelo sistema de sequenciamento.

Quando o estudante completa uma atividade, entra em ação o sistema que define

Tabela 4 – Conjunto de atividades cadastradas no sistema

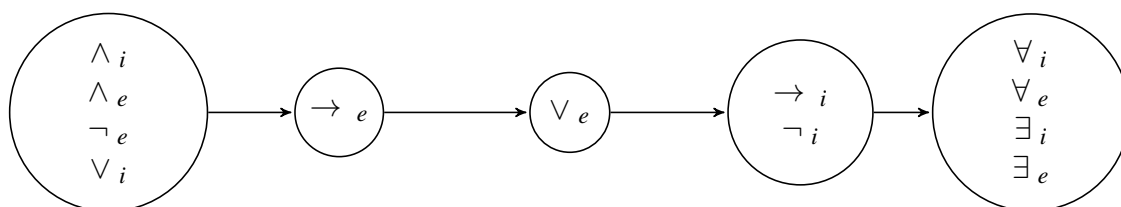
Atividade	Solução	Dificuldade	Complexidade
$(a \rightarrow b) \wedge (c \rightarrow d) \vdash (a \wedge c) \rightarrow (b \wedge d)$	$\wedge_e 1$ $\wedge_e 1$ Hip: $a \wedge c$ $\wedge_e 4$ $\wedge_e 4$ $\rightarrow_e 5, 2$ $\rightarrow_e 6, 3$ $\wedge_i 7, 8$ $\rightarrow_i 4-9$	10	$\wedge_e: 4$ $\rightarrow_e: 2$ $\wedge_i: 1$ $\rightarrow_i: 1$
$a \wedge b \vdash a$	$\wedge_e 1$	2	$\wedge_e: 1$
$(a \wedge b) \rightarrow c, \neg c \vdash \neg(a \wedge b)$	Hip: $a \wedge b$ $\rightarrow_e 1, 3$ $\wedge_i 2, 4$ $\neg_i 3-5$	6	$\rightarrow_e: 1$ $\wedge_i: 1$ $\neg_i: 1$
$a \rightarrow b, \neg b \vdash \neg a$	Hip: a $\rightarrow_e 1, 3$ $\wedge_i 2, 4$ $\neg_i 3, 5$	6	$\rightarrow_e: 1$ $\wedge_i: 1$ $\neg_i: 1$
$p \wedge q, r \vdash p \wedge r$	$\wedge_e 1$ $\wedge_i 3, 2$	4	$\wedge_e: 1$ $\wedge_i: 1$
$p \rightarrow q, q \rightarrow s \vdash \neg s \rightarrow \neg p$	Hip: $\neg s$ Hip: p $\rightarrow_e 1, 4$ $\rightarrow_e 2, 5$ $\wedge_i 3, 6$ $\neg_i 4, 7$ $\rightarrow_i 3, 8$	14	$\rightarrow_e: 2$ $\wedge_i: 1$ $\neg_i: 1$ $\rightarrow_i: 1$

Fonte: Autor (2019)

qual atividade é apropriada a ser desenvolvida a seguir. Primeiramente, é selecionado o conceito que será trabalhado na próxima atividade do aluno. Para isso, leva-se em consideração o número de atividades concluídas em cada conceito e a nota média dessas atividades. Foi estabelecida uma ordem de complexidade dos conceitos, mostrada na Figura 14, baseada em cinco grupos. Os conceitos mais simples são representados no grupo à esquerda – introdução da conjunção (\wedge_i), eliminação da conjunção (\wedge_e), eliminação da negação (\neg_e) e introdução da disjunção (\vee_i) – servindo de pré-requisito ao conceito do segundo grupo – eliminação da implicação (\rightarrow_e). O próximo grupo intermediário,

formado por um conceito – eliminação da disjunção (\vee_e) – é seguido pelo último grupo da lógica proposicional, com os dois conceitos considerados mais complexos – introdução da implicação (\rightarrow_i) e introdução da negação (\neg_i). Já o último grupo representa os conceitos da lógica de predicados – introdução e eliminação do quantificador universal e do quantificador existencial – considerados mais complexos que os conceitos da lógica proposicional. Assim, o Tutor avalia primeiramente se o estudante concluiu atividades suficientes dos conceitos básicos. Durante a fase de desenvolvimento do sistema, esse valor foi ajustado para que seja necessário resolver pelo menos 3 exercícios do conceito antes de avançar, sendo que pelo menos 1 deles deve ser considerado de dificuldade média ou alta. As atividades com dificuldade de valor até 5 foram consideradas de nível fácil, para valores de 6 a 11 nível médio e, para valores de 12 ou mais, nível difícil. Caso o estudante ainda não tenha concluído o número mínimo de atividades de um dos conceitos básicos, este é selecionado. Caso já tenha realizado o número mínimo, é avaliada a nota média dele nesses conceitos. Se uma das notas for menor que 6, entende-se que ele não está preparado para avançar a conceitos mais complexos, então este conceito é selecionado para seguir sendo trabalhado. Caso a nota nos conceitos seja maior ou igual a 6, passa-se ao próximo grupo de conceitos da sequência, onde a mesma avaliação de número de atividades e nota média é realizada.

Figura 14 – Sequenciamento de conceitos utilizado



Fonte: Autor (2019)

Após ser selecionado o conceito que deve ser trabalhado, o Tutor seleciona qual das atividades dentro desse conceito será mostrada a seguir. Por padrão, é selecionada a atividade de menor dificuldade dentre as que ainda não foram realizadas pelo estudante. Caso todas as atividades deste conceito já tenham sido realizadas, é selecionada aquela em que ele obteve a menor nota, permitindo substituir a nota anterior caso obtenha uma mais alta.

É possível exportar o perfil do estudante para que seja carregado em outro computador ou alternar entre perfis na mesma máquina. Para tanto, foi implementado um menu com as opções *Exportar perfil*, *Importar perfil* e *Redefinir perfil*. A função de exportação

cria um arquivo com extensão própria do sistema – chamada LITS, em alusão ao nome da aplicação, LogicITS (*Logic Intelligent Tutoring System*) – contendo a sequência de atividades realizadas pelo estudante e suas avaliações. O formato do arquivo LITS é o próprio *backup* gerado pelo banco de dados H2, apenas para as tabelas pertinentes. Com a opção de redefinir perfil, todas as informações sobre sobre atividades realizadas são excluídas, retornando o estado do sistema àquele de sua primeira utilização. A opção de importar perfil, por sua vez, carrega um arquivo com a extensão LITS e retoma o estado do sistema no momento em que aquele arquivo foi gerado, sobrescrevendo o andamento atual do estudante.

5.6 Integração com o modelo pedagógico

Para evidenciar a aplicação das metodologias pedagógicas discutidas, é necessário fazer a relação entre os elementos projetados na arquitetura apresentada e como os objetivos pretendem ser atingidos. Foram utilizados conceitos da teoria de aprendizagem significativa e do modelo de design instrucional 4C/ID.

A aprendizagem significativa concentra-se principalmente na teoria dos subsunçores, que são os conhecimentos anteriores do aluno que se relacionam com o novo conhecimento que é apresentado. Este conceito é aplicado através do sequenciamento de atividades, onde os exercícios propostos são incrementados gradualmente em termos de complexidade. Destarte, o sistema se baseia na fixação dos conteúdos antes da apresentação dos novos, seguindo uma ordem lógica de navegação através dos tópicos definida no domínio da aplicação.

Cada componente do modelo de design instrucional 4C/ID possui também seus elementos correspondentes à arquitetura do sistema. As *tarefas de aprendizagem*, ou atividades-fim, correspondem ao domínio da aplicação, que também é organizado em categorias de tarefas com nível equivalente conforme especificado no modelo. Da mesma forma, as atividades inicialmente são apresentadas com grande nível de suporte, que vai desaparecendo conforme o estudante avança naquele tópico. Outro componente do 4C/ID, a *informação de apoio*, aparece no sistema na forma de material de ensino sobre todos os tópicos abordados na aplicação. Assim, permite ao aluno resolver os problemas de forma eficiente, fazendo a ligação entre os seus conhecimentos prévios e os novos através de modelos mentais, estratégias cognitivas e *feedback* cognitivo. O terceiro componente, *informação processual*, é aquele que permite detectar aspectos da resolução pelo indivi-

duo e aparece na forma de *feedback* imediato durante a resolução das provas. Já o último componente, a *prática nas tarefas*, refere-se especificamente à aplicação das regras pelo aluno, visando alcançar o nível necessário de automaticidade através da repetição dos conjuntos de regras complexas.

5.7 Testes e validação

Foram realizados testes funcionais para confirmar que a execução do sistema está ocorrendo de acordo com o esperado. Os casos de teste, detalhados no apêndice C, foram desenvolvidos e executados para cada funcionalidade implementada no sistema. Os testes foram realizados nas plataformas Windows 10 e Linux Mint. Nos casos de teste são listadas uma série de funcionalidades presentes nos menus e interface principal do sistema, como exportar/importar perfil, aplicar as regras de inferência e solicitar ajuda. Os casos de teste estão especificados de acordo com as seguintes características: (i) identificador do caso de teste, no formato #00, (ii) ação, que descreve a funcionalidade sendo testada, (iii) pré-condição, que é o estado do sistema necessário para a execução do teste, (iv) passos, que descreve em etapas cada ação do usuário para a execução do teste, (v) resultado esperado, que descreve a saída ou ação gerada pelo sistema após a execução de teste e (vi) pós-condição, que descreve o estado do sistema após a execução do teste. O sistema passou em todos os casos de teste, confirmando que é funcional.

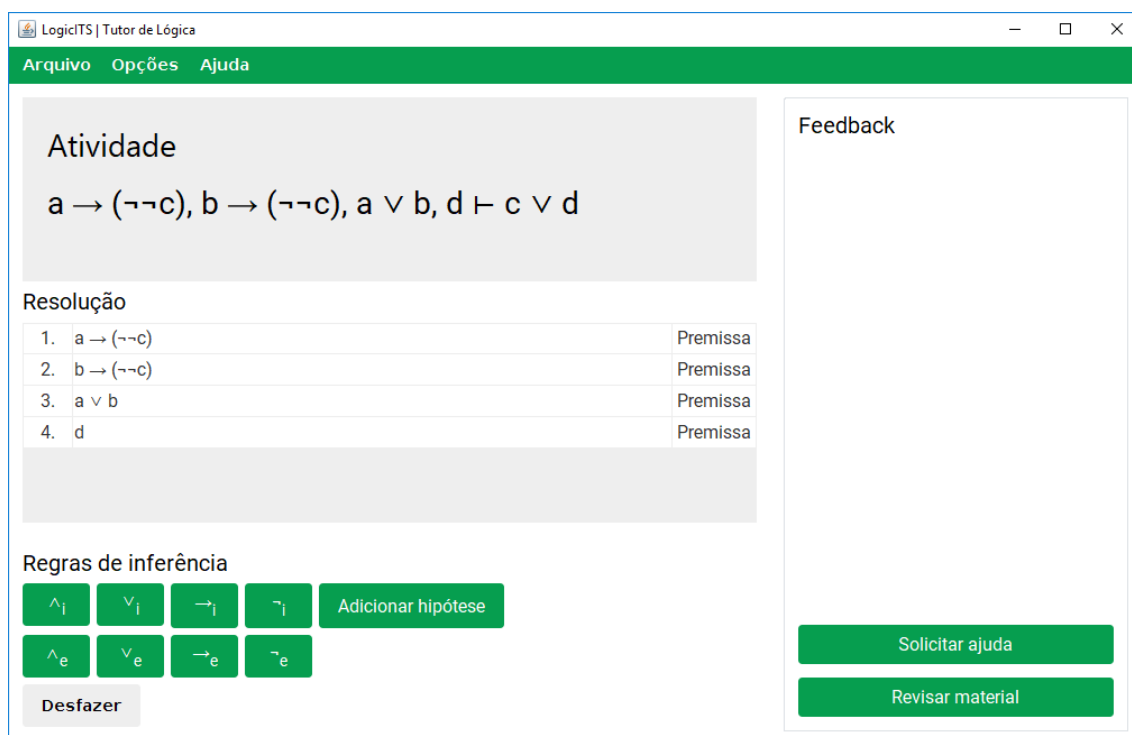
Para a validação do sistema produzido, foi realizada uma avaliação qualitativa por duas docentes responsáveis pela disciplina de Lógica para Computação em cursos de graduação. A avaliação consistiu na execução de um conjunto sugerido de atividades, seguido da resposta de um questionário.

6 RESULTADOS

Neste capítulo, são apresentados os resultados obtidos com este projeto, avaliando sua aderência aos objetivos iniciais propostos e sua usabilidade final.

O sistema desenvolvido, cuja interface pode ser vista na Figura 15, contém as funcionalidades definidas de um sistema tutor inteligente, que são o sequenciamento de currículo, a ajuda na resolução de problemas e a análise das soluções do estudante. Além disso, a metodologia pedagógica do sistema foi embasada nas teorias da aprendizagem significativa e do modelo 4C/ID do *design* instrucional, garantindo que o aspecto educacional da ferramenta utilize conceitos conhecidos e testados. Na etapa de testes do sistema, foi executada uma série de casos de teste para garantir seu correto funcionamento em diferentes máquinas e sistemas operacionais. Os resultados dos testes mostram que o sistema é operacional em todas as suas funcionalidades nos ambientes testados. A avaliação dos resultados do projeto consiste na verificação dos requisitos elencados cumpridos, da validação realizada por profissionais da área de ensino de Lógica para Computação, além da simulação do funcionamento do sistema para diferentes perfis de alunos.

Figura 15 – Interface do sistema implementado



Fonte: Autor (2019)

A seguir, é mostrada a relação de requisitos elicitados que foram atendidos ou não

com o sistema desenvolvido. A numeração dos requisitos é em relação às listas apresentadas no Apêndice A. Verificando a Tabela 5, pode-se perceber que todos os requisitos não funcionais do sistema foram atendidos. Esses requisitos dizem respeito à compatibilidade entre sistemas operacionais, à metodologia de programação que permita inclusão de novas funcionalidades com facilidade e à usabilidade e interface do sistema. O desenvolvimento modular e orientado a objetos facilita a possibilidade de expansão da ferramenta.

Tabela 5 – Situação dos requisitos não funcionais

Nº do requisito não funcional	Prioridade	Situação
1	Importante	Atendido
2	Importante	Atendido
3	Essencial	Atendido

Fonte: Autor (2019)

Na Tabela 6, que trata dos requisitos funcionais, é mostrado que a maioria dos requisitos foram atendidos. Alguns deles, contudo, foram atendidos parcialmente ou não atendidos. É importante destacar que os requisitos de mais alta prioridade, considerados essenciais, foram todos implementados. Os requisitos essenciais referem-se às funções básicas do sistema para que cumpra seu objetivos, sendo especificamente referentes às funções de armazenamento do perfil do estudante, sequenciamento do currículo, apresentação de exercícios de dedução natural, apresentação de material de apoio e acompanhamento das resoluções com oferecimento de ajuda.

Entre os requisitos considerados importantes, três foram atendidos e dois foram parcialmente atendidos. Os requisitos importantes atendidos referem-se à representação das fórmulas lógicas em formato de suas árvores sintáticas, capacidade de salvar e carregar as atividades já realizadas e capacidade de exportar e importar o perfil do usuário. O primeiro requisito importante parcialmente atendido refere-se a realização de cálculos tanto sobre a Lógica Proposicional quanto a Lógica de Predicados, onde o último foi parcialmente implementado. A Lógica de Predicados foi implementada em nível de domínio, estando cadastrada na base de conhecimentos e sequenciamento de currículo, porém não teve suas regras de inferência implementadas devido ao escopo de tempo para realização do projeto, onde outras funcionalidades foram priorizadas. Sendo assim, a ferramenta já possui a estrutura necessária para a execução de cálculo sobre a Lógica de Predicados, bastando implementar as regras de negócio especificamente referentes às quatro regras

Tabela 6 – Situação dos requisitos funcionais

Nº do requisito funcional	Prioridade	Situação
1	Importante	Parcialmente atendido
2	Desejável	Parcialmente atendido
3	Importante	Atendido
4	Importante	Parcialmente atendido
5	Essencial	Atendido
6	Essencial	Atendido
7	Desejável	Não atendido
8	Importante	Atendido
9	Desejável	Parcialmente atendido
10	Essencial	Atendido
11	Essencial	Atendido
12	Essencial	Atendido
13	Importante	Atendido

Fonte: Autor (2019)

adicionais dessa lógica na Dedução Natural. Já o segundo requisito importante parcialmente atendido refere-se à geração de um instalador para cada sistema operacional, onde optou-se por distribuir o sistema na forma de um executável. Esta opção, contudo, não causa grande impacto sobre os objetivos e resultados da ferramenta, pois é perfeitamente utilizável, tanto quanto seria na forma de um instalador.

Quanto aos requisitos de mais baixa prioridade, marcados como desejáveis, dois deles foram parcialmente atendidos e um não foi atendido. O primeiro deles que foi parcialmente atendido diz respeito ao formato de menus do programa, onde a implementação final foi adaptada conforme se julgou necessário às funcionalidades implementadas, não seguindo o padrão planejado de itens de menu (Arquivo, Sistemas de prova, Idioma, Ajuda, Sair). O outro requisito desejável parcialmente implementado refere-se à possibilidade de salvar provas em PDF, onde em alternativa a isso foi implementada a função de salvar provas em formato de imagem. Esta opção tornou a implementação mais simples e viável em termos de tempo. Para que seja implementado futuramente o suporte à geração de PDF, poderá ser utilizada uma biblioteca do Java que contenha essa funcionalidade. Já o único requisito não atendido refere-se à implementação de opções de idiomas no sis-

tema, com opções de português, inglês e espanhol. Este requisito não foi implementado por ser considerado de baixa prioridade, de modo que as demais funções do sistema receberam mais atenção durante o tempo destinado à implementação. Sua implementação seria simples em termos de programação – bastando disparar um evento para alterar os textos contidos em botões, menus e materiais de apoio –, porém trabalhosa em termos da necessidade de traduzir todo o conteúdo do sistema para os dois idiomas estrangeiros.

Foram realizadas simulações do desempenho de estudantes para demonstrar a personalização do sequenciamento de currículo. Para isso, foram elaborados três perfis de alunos com características de desempenho diferentes, cujos caminhos através das atividades do sistema são mostrados na Tabela 7. O Perfil A simula um aluno com pouco ou nenhum conhecimento sobre dedução natural, que frequentemente tenta aplicar as regras de inferência de maneira equivocada e necessita revisar o material de apoio, resultando em uma nota mais baixa na avaliação das atividades. O Perfil B simula um aluno com mais conhecimento das regras e que adapta-se facilmente ao uso do sistema, cometendo poucos erros e necessitando pouca revisão do material. Já o Perfil C representa um estudante com desempenho intermediário, que obtém pontuação média nas primeiras atividades, mas encontra dificuldades logo que um novo conceito é apresentado, recebendo avaliações mais baixas. Para cada perfil, foi realizada uma sequência de sete atividades com desempenhos diferentes, de modo que a sequência de atividades selecionadas pelo sistema torna-se diferente para cada um dos casos. A segunda coluna da tabela mostra a atividade selecionada pelo Tutor precedida por seu número de identificação no sistema (#), enquanto a terceira coluna contém a quantidade de erros/ajudas do estudante na atividade – como explicado na Seção 5.5, são considerados na avaliação tanto os erros que o estudante comete, tentando aplicar as regras de inferência de maneira equivocada, quanto as solicitações de ajuda e revisão do material de apoio. A quarta coluna mostra a nota obtida em cada atividade, conforme a fórmula utilizada pelo Tutor – a cada erro/ajuda, desconsiderando os 3 primeiros, é descontado um valor de 0,5 da nota inicial, que é 10. A última coluna mostra o próximo conceito a ser trabalhado pelo estudante, mediante seu desempenho geral em todas as atividades anteriores. Todas as simulações iniciaram do mesmo estado, um perfil em branco onde o primeiro conceito selecionado pelo Tutor é a introdução da conjunção (\wedge i).

Observando os resultados das simulações, percebe-se que o sistema é capaz de personalizar o caminho dos usuários de acordo com seu desempenho, levando em consideração variáveis como a nota do estudante nas atividades realizadas, a quantidade de

Tabela 7 – Simulação do sequenciamento de currículo

Perfil	[#] Atividade	Erros	Nota	Próximo conceito
A	[6] $p \wedge q, r \vdash p \wedge r$	16	3.5	\wedge_i
	[23] $a \wedge b \vdash b \wedge a$	14	4.5	\wedge_i
	[17] $(p \wedge q) \rightarrow r, p, q \vdash r$	10	6.5	\wedge_i
	[24] $\vdash (p \wedge q) \rightarrow (q \wedge p)$	12	5.5	\wedge_i
	[5] $a \rightarrow b, \neg b \vdash \neg a$	9	7	\wedge_e
	[3] $a \wedge b \vdash a$	6	8.5	\wedge_e
	[15] $(p \wedge q) \wedge r, s \wedge t \vdash q \wedge s$	15	4	\neg_e
B	[6] $p \wedge q, r \vdash p \wedge r$	5	9	\wedge_i
	[23] $a \wedge b \vdash b \wedge a$	6	8.5	\wedge_i
	[5] $a \rightarrow b, \neg b \vdash \neg a$	4	9.5	\wedge_e
	[15] $(p \wedge q) \wedge r, s \wedge t \vdash q \wedge s$	2	10	\neg_e
	[25] $\neg\neg a, \neg\neg b \vdash a \wedge b$	1	10	\neg_e
	[8] $a \rightarrow \neg\neg c, b \rightarrow \neg\neg c, a \vee b \vdash c \vee d$	6	8.5	\neg_e
	[26] $a \rightarrow b, \neg\neg a \wedge (b \rightarrow c)$	3	10	\vee_i
C	[6] $p \wedge q, r \vdash p \wedge r$	13	5	\wedge_i
	[23] $a \wedge b \vdash b \wedge a$	11	7	\wedge_i
	[17] $(p \wedge q) \rightarrow r, p, q \vdash r$	5	9	\wedge_i
	[5] $a \rightarrow b, \neg b \vdash \neg a$	6	8.5	\wedge_e
	[3] $a \wedge b \vdash a$	17	3	\wedge_e
	[24] $\vdash (p \wedge q) \rightarrow (q \wedge p)$	14	4.5	\wedge_e
	[15] $(p \wedge q) \wedge r, s \wedge t \vdash q \wedge s$	8	7.5	\neg_e

Fonte: Autor (2019)

atividades e a dificuldade delas. A simulação abrange um conjunto pequeno de atividades – sete para cada perfil – e mesmo assim é possível observar as mudanças na sequência de aprendizagem de acordo com as ações do usuário. O Perfil B, que possui mais facilidade na resolução dos exercícios, abrange um maior número de conceitos em menor tempo, visto que, por ter bom desempenho, é permitido avançar para os próximos tópicos mais rapidamente. Os Perfis A e C, que possuem desempenhos intermediários, percorrem uma sequência diferente de atividades entre si, de modo que é possível observar que o Perfil C fica mais tempo retido no conceito \wedge_e (eliminação da conjunção) por obter notas menores nas atividades relacionadas a ele.

Para a validação do sistema, foi contatada a docente responsável pela disciplina de Lógica para Computação onde é trabalhado o conteúdo de Dedução Natural no Campus Alegrete da UNIPAMPA. A docente respondeu a um questionário, apresentado no apêndice D, após ser sugerida uma série de passos para utilização do sistema, também

apresentados no mesmo apêndice. Na primeira questão, que aborda os aspectos positivos do sistema, foi citado que “a ajuda e o material resumido são de grande ajuda”. Na segunda questão, que aborda aspectos que podem ser melhorados ou acrescentados, foi sugerido incluir dicas sobre os operadores presentes nas premissas para cada atividade, com a justificativa de que “a maior dificuldade dos alunos no ensino de lógica é justamente olhar todas as regras e saber qual ou quais ele pode usar naquelas premissas”. Sobre a terceira questão, a respondente concorda totalmente que “o sistema tem potencial para incentivar e facilitar o estudo extraclasse pelos alunos de graduação nas disciplinas relacionadas à Lógica para Computação”. No que se refere à quarta questão, a professora concorda parcialmente com a afirmação “o sistema tem potencial para melhorar o desempenho dos alunos e reduzir desistências nas disciplinas com esses conteúdos”. Quanto à asserção “o sistema é uma alternativa útil aos alunos para o estudo sem dependência do professor, com a capacidade de avaliar seus próprios resultados”, foi assinalado que a avaliadora concorda totalmente. No campo aberto a comentários adicionais, foi colocada a resposta: “Gostei muito do sistema desenvolvido, da simplicidade e o modo como facilita o estudo dos alunos, excluindo a ideia de fazer todas as deduções lógicas em papel”. A partir das respostas obtidas na avaliação, observa-se que o sistema alcançou grau satisfatório como ferramenta útil de apoio ao ensino, ao mesmo tempo que ainda possui potencial para aprimoramentos.

7 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver um sistema multiplataforma de apoio ao processo de ensino-aprendizagem de conteúdos de Lógica para Computação, especificamente sobre Dedução Natural, que escolha um trajeto de aprendizagem para cada aluno, dependendo do seu nível de conhecimento e que disponibilize apoio e *feedback* imediatos. Foram cumpridos os objetivos de identificar as tecnologias e técnicas envolvidas no desenvolvimento de um sistema tutor inteligente, além de integrá-las aos conceitos de Informática na Educação para criar uma ferramenta apta a ser utilizada em disciplinas de graduação. Utilizando as técnicas de Engenharia de Software, foi possível desenvolver um sistema que aplica os conhecimentos da lógica para computação para estas finalidades.

O estudo aprofundado do referencial teórico foi essencial para obter os conhecimentos necessários à elaboração da proposta de um sistema tutor inteligente, visto que as informações referentes a esse tipo de sistema não são comumente conhecidas ou trabalhadas nos cursos de graduação. Existe muito conteúdo acerca do assunto, de modo que as várias tecnologias e aplicações precisaram ser analisadas para se chegar à escolha das mais viáveis e adequadas ao contexto deste trabalho. Uma dificuldade encontrada na etapa do levantamento bibliográfico sobre sistemas tutores inteligentes foi que há muitas referências sobre esse tipo de ferramenta e suas características, mas poucos fazem referência à maneira como são implementados em termos práticos, explicitando o algoritmo utilizado. Dessa forma, o desenvolvimento do sistema precisou ser planejado sem haver uma base muito clara além das definições dos conceitos utilizados nesse tipo de sistema. A partir dos estudos realizados durante o levantamento bibliográfico, foi elaborada a proposta de sistema e modelagem inicial da arquitetura. Estes documentos serviram como base para a implementação que viria a seguir.

A etapa de implementação, a mais longa do projeto, exigiu aplicar conhecimentos de diversas funcionalidades da linguagem de programação Java. O desenvolvimento da interface, a implementação da resolução dos exercícios, o sequenciamento de currículo e a importação/exportação de perfil são exemplos de funcionalidades distintas entre si e que necessitaram a utilização de recursos específicos da linguagem. A principal dificuldade encontrada no desenvolvimento foi com relação à distribuição da aplicação. Como descrito no capítulo 2, o sistema de gestão de banco de dados precisou ser substituído no momento em que foi necessário gerar uma versão para a realização dos testes por terceiros. A substituição em um estágio tão avançado do desenvolvimento gerou certo transtorno,

necessitando de revisão nos códigos, que poderia ter sido evitado caso o planejamento tivesse sido mais cuidadoso.

O propósito do desenvolvimento deste projeto foi sua aplicação prática em disciplinas de Lógica para Computação nos cursos de graduação. Acredita-se, a partir das análises e avaliações preliminares realizadas, que sua utilização seja viável, com potencial para auxiliar o processo de aprendizagem dos alunos nos conteúdos abordados. O sistema, além de sugerir um conjunto de atividades ao estudante, permite a inserção de novas atividades para que sejam resolvidas na ferramenta. Isso permite, por exemplo, que o estudante insira as atividades de sua lista de exercícios no sistema e as resolva, contando com o auxílio da ferramenta para garantir a aplicação correta dos procedimentos da resolução.

São identificadas algumas oportunidades de melhoramento e expansão da ferramenta, sugeridas para trabalhos futuros. A primeira delas seria implementar os requisitos elencados primeiramente e que não foram atendidos neste projeto, sendo o principal deles as atividades em Lógica de Primeira Ordem, seguido do suporte a múltiplos idiomas. Outra funcionalidade adicional seria a personalização de domínio, que permite a um professor, por exemplo, inserir através da aplicação novas atividades, soluções, materiais de apoio e *feedbacks*. Opções adicionais, como gravar o estado de uma prova durante sua execução para ser retomada mais tarde, exportar resolução das provas em formato PDF, também seriam importantes.

Uma sugestão para projetos futuros é a concentração de esforços no algoritmo de sequenciamento de currículo. Para este projeto, foi implementado um algoritmo que realiza um sequenciamento simples. Apesar de não ser o escopo desse projeto, técnicas mais sofisticadas poderiam ser empregadas, utilizando por exemplo conceitos de Inteligência Artificial para produzir um caminho realmente otimizado ao estudante. Como demanda um sistema tutor inteligente, o algoritmo desenvolvido leva em consideração variáveis relacionadas às interações do estudante e dificuldade das atividades para fornecer um caminho otimizado dependendo do desempenho do estudante. Além disso, da maneira como foi desenvolvido, o sistema fornece uma API capaz de manipular todas as suas variáveis, de modo que uma implementação de um novo algoritmo de sequenciamento torna-se mais fácil.

No questionário de validação, a docente consultada sugeriu possibilidades de aprimoramento de alguns aspectos de como são apresentados os *feedbacks*. Seria interessante dar dicas específicas sobre a atividade, sugerindo quais regras podem ser usadas, além de

informar o aluno imediatamente quando ele seleciona uma regra que não pode ser usada. Atualmente, esse *feedback* aparece um pouco atrasado, apenas quando o aluno seleciona a quantidade máxima de fórmulas para a regra selecionada e realiza a ação de aplicar.

O processo de validação poderia ter sido melhor se houvesse a oportunidade de aplicar o uso da ferramenta na prática, em uma turma de alunos da disciplina de Lógica para Computação. Assim, seria realizada uma comparação entre o desempenho de aprendizagem de estudantes com e sem o auxílio do sistema em um mesmo ambiente. Contudo, não foi possível fazer o teste, visto que não houve a disciplina de Lógica no Campus Bagé da Unipampa no semestre de implementação do trabalho, e uma aplicação em outros lugares implicaria em dificuldades físicas adicionais. Os testes dos alunos poderiam ainda servir como uma validação adicional do sistema, contando nesse caso com a validação dos especialistas – os docentes – e dos próprios estudantes que são os usuários finais do sistema.

Ao utilizar o sistema na prática, novas possibilidades de aprimoramentos poderão ser identificadas. Através da obtenção de estatísticas de uso, podem ser verificados aspectos como a eficiência do sequenciamento de currículo, os tipos de *feedback* mais eficazes, as maiores dificuldades dos discentes que não são supridas pelo sistema, entre outros. A partir dessas informações, seria possível gerar uma nova leva de requisitos com vistas a suprir as demandas apontadas.

Com a finalização do projeto, pode-se concluir que sua execução estabeleceu as bases de um sistema tutor inteligente que pode continuar a ser desenvolvido com o intuito de, além do apoio aos estudantes de graduação, mostrar que na UNIPAMPA se faz pesquisa em Educação de qualidade, com suporte computacional e que ajude outros estudantes com os desafios impostos no ensino de Lógica. Do ponto de vista pessoal, foi importante para agregar conhecimentos acerca de temas como Informática na Educação e desenvolvimento de sistemas computacionais. Além disso, o processo de elaborar todas as etapas de uma pesquisa desse tipo representa uma contribuição significativa na formação de um Engenheiro de Computação.

REFERÊNCIAS

- ANDERSON, J. R. **The Architecture of Cognition**. [S.l.]: Harvard University Press, 1983.
- ANDERSON, J. R.; BELLEZZA, F. **Rules of the mind**. Hillsdale, NJ: Erlbaum Associates, 1993.
- ANDERSON, J. R. *et al.* Cognitive tutors: Lessons learned. **The journal of the learning sciences**, Taylor & Francis, v. 4, n. 2, p. 167–207, 1995.
- ANDREWS, D. H.; GOODSON, L. A. A comparative analysis of models of instructional design. **Journal of instructional development**, Springer, v. 3, n. 4, p. 2–16, 1980.
- ARAVIND, V. R.; REFUGIO, C. Efficient learning with intelligent tutoring across cultures. **World Journal on Educational Technology: Current Issues**, ERIC, v. 11, n. 1, p. 30–37, 2019.
- AUSUBEL, D. P. Aquisição e retenção de conhecimentos: uma perspectiva cognitiva. **Plátano**, Lisboa, v. 1, 2003.
- BARTLETT, F. **Thinking: An experimental and social study**. [S.l.]: Basic Books, 1958.
- BAUER, M. W.; GASKELL, G. **Pesquisa qualitativa com texto, imagem e som: um manual prático**. [S.l.]: Editora Vozes Limitada, 2017.
- BERTOLDI, S.; RAMOS, E. M. F. **Avaliação de Software Educacional – Impressões e Reflexões**. Dissertação (Mestrado) — Universidade Federal de Santa Catarina, 1999.
- BLACKBURN, P.; RIJKE, M. de; VENEMA, Y. **Modal Logic**. Cambridge, UK: Cambridge University Press, 2001. (Cambridge Tracts in Theoretical Computer Science, v. 53). 554p. ISBN 0 521 52714 7 (paperback).
- BLOOM, B. S. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. **Educational researcher**, Sage Publications Sage CA: Thousand Oaks, CA, v. 13, n. 6, p. 4–16, 1984.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML: guia do usuário**. [S.l.]: Elsevier Brasil, 2006.
- BRANCH, R. M. **Instructional design: The ADDIE approach**. [S.l.]: Springer Science & Business Media, 2009.
- BRUSILOVSKY, P. *et al.* Adaptive and intelligent technologies for web-based education. **Ki**, Citeseer, v. 13, n. 4, p. 19–25, 1999.
- BUCHANAN, T. The efficacy of a world-wide web mediated formative assessment. **Journal of Computer Assisted Learning**, Wiley Online Library, v. 16, n. 3, p. 193–200, 2000.
- CASTRO, L. dos S.; SANTOS, R. da S. *et al.* Educação e teorias da aprendizagem: um foco na teoria de Vygotsky. **Revista da Universidade Vale do Rio Verde**, v. 11, n. 1, p. 551–559, 2013.

CHEN, G.; PHAM, T. T. **Introduction to Fuzzy Sets, Fuzzy Logic, and Fuzzy Control Systems**. [S.l.]: CRC Press, 2001.

CROFT, R.; BEDI, S. eBooks for a distributed learning university: The Royal Roads University case. **Journal of library administration**, Taylor & Francis, v. 41, n. 1-2, p. 113–137, 2004.

DIAS, B.; FINGER, A. Busca por estratégias para diminuição de retenção em disciplinas de lógica matemática. In: **V Workshop-Escola de Informática Teórica**. [S.l.: s.n.], 2019.

DICK, W.; CAREY, L.; CAREY, J. O. **The systematic design of instruction**. [S.l.]: Citeseer, 2005.

EMERSON, E. A. Temporal and modal logic. In: LEEUWEN, J. van (Ed.). **Handbook of Theoretical Computer Science**. Amsterdam: Elsevier, 1998. v. 2 (Formal models and semantics), cap. 16, p. 995–1072. ISBN 0-444-88074-7.

FILATRO, A.; PICONEZ, S. C. B. Design instrucional contextualizado. **São Paulo: Senac**, 2004.

FITTING, M. **First-order logic and automated theorem proving**. 2. ed. New York: Springer, 1996. (Graduate texts in Computer Science). 326p. ISBN 0-387-94593-8.

FURLAN, J. D. **Modelagem de objetos através da UML - The Unified Modeling Language**. [S.l.]: Makron books, 1998.

GALAFASSI, F. F. P. **Agente pedagógico para mediação do processo de ensino-aprendizagem da dedução natural na lógica**. Dissertação (Mestrado) — Universidade do Vale do Rio dos Sinos, 2012.

GROSS, A. L.; ROUSE, L. J. An interactive map of plate boundaries for geoscience teachers. **Abstracts with Programs—Geological Society of America**, v. 47, n. 2, p. 11, 2015.

HAFIDI, M.; BENSEBAA, T. Developing adaptive and intelligent tutoring systems (AITS): A general framework and its implementations. **International Journal of Information and Communication Technology Education (IJICTE)**, IGI Global, v. 10, n. 4, p. 70–85, 2014.

HOLLAND, J.; MITROVIC, A.; MARTIN, B. J-LATTE: a Constraint-based Tutor for Java. University of Canterbury. Computer Science and Software Engineering, 2009.

HOOSHYAR, D. *et al.* Applying an online game-based formative assessment in a flowchart-based intelligent tutoring system for improving problem-solving skills. **Computers & Education**, Elsevier, v. 94, p. 18–36, 2016.

HUTH, M.; RYAN, M. **Logic in Computer Science: Modelling and reasoning about systems**. [S.l.]: Cambridge university press, 2004.

JANES, V. Learning outside the classroom. **Community Care**, 2018.

LEWIS, H. R.; PAPADIMITRIOU, C. H. **Elements of the Theory of Computation**. [S.l.]: Prentice Hall PTR, 1997.

LUDKE, M.; ANDRÉ, M. E. Pesquisa em educação: abordagens qualitativas. **Em Aberto**, v. 5, n. 31, 2011.

MA, W.; ADESOPE, O. O. Intelligent tutoring systems and learning outcomes: A meta-analysis. In: **Journal of Educational Psychology**. [S.l.]: American Psychological Association, 2014.

MELO, M.; MIRANDA, G. L. Efeito do modelo 4C/ID sobre a aquisição e transferência de aprendizagem: revisão de literatura com meta-análise. **RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação**, Associação Ibérica de Sistemas e Tecnologias de Informação (AISTI), n. 18, p. 114–130, 2016.

MERRIËNBOER, J. J. V.; CLARK, R. E.; CROOCK, M. B. D. Blueprints for complex learning: The 4C/ID-model. **Educational technology research and development**, Springer, v. 50, n. 2, p. 39–61, 2002.

MERRIËNBOER, J. J. V.; KESTER, L. The four-component instructional design model: Multimedia principles in environments for complex learning. **The Cambridge handbook of multimedia learning**, Cambridge University Press New York, p. 71–93, 2005.

MERRILL, M. D. *et al.* Reclaiming instructional design. **Educational Technology**, v. 36, n. 5, p. 5–7, 1996.

MILLER, G. A. The magical number seven, plus or minus two: Some limits on our capacity for processing information. **Psychological review**, American Psychological Association, v. 63, n. 2, p. 81, 1956.

MITROVIC, A. An intelligent SQL tutor on the web. **International Journal of Artificial Intelligence in Education**, IOS Press, v. 13, n. 2-4, p. 173–197, 2003.

MITROVIC, A.; MARTIN, B.; SURAWEEERA, P. Intelligent tutors for all: The constraint-based approach. **IEEE Intelligent Systems**, IEEE, n. 4, p. 38–45, 2007.

MOREIRA, M. A. Aprendizagem significativa: da visão clássica à visão crítica (meaningful learning: from the classical to the critical view). In: **Conferência de encerramento do V Encontro Internacional sobre Aprendizagem Significativa**. [S.l.: s.n.], 2006.

NOGUEIRA, J. de S. *et al.* Utilização do computador como instrumento de ensino: Uma perspectiva de aprendizagem significativa. **Revista Brasileira de Ensino de Física**, v. 22, n. 4, p. 517, 2000.

OHLSSON, S. Constraint-based student modeling. In: GREER *et al.* (Ed.). **Student Modelling: The Key to Individualized Knowledge-Based Instruction**. [S.l.]: Springer, 1994. p. 167–189.

PALAZZO, D. J. *et al.* Patterns, correlates, and reduction of homework copying. **Physical review special topics-Physics education research**, APS, v. 6, n. 1, p. 010104, 2010.

PELLIZZARI, A. *et al.* Teoria da aprendizagem significativa segundo Ausubel. **Revista PEC**, v. 2, n. 1, p. 37–42, 2002.

PENTEADO, F.; GLUZ, J. C. Sistema heráclito: Suporte a objetos de aprendizagem interativos e dialéticos voltados ao ensino de dedução natural na lógica proposicional. In: **Anais do XXII SBIE**. [S.l.]: XVII WIE, 2011.

REGIAN, J. Functional area analysis of intelligent computer-assisted instruction (report of TAPSTEM ICAI-FAA Committee). **Brooks AFB, TX**, 1997.

REISER, B. J.; ANDERSON, J. R.; FARRELL, R. G. Dynamic Student Modelling in an Intelligent Tutor for LISP Programming. In: **IJCAI**. [S.l.: s.n.], 1985. v. 85, p. 8–14.

SANTOS, R. P.; COSTA, H. A. X. TBC-AED e TBC-AED/WEB: Um desafio no ensino de algoritmos, estruturas de dados e programação. In: **IV WEIMIG - Workshop de Educação em Computação e Informática do Estado de Minas Gerais**. [S.l.: s.n.], 2005.

SHUTE, V. J. Focus on formative feedback. **Review of educational research**, Sage Publications, v. 78, n. 1, p. 153–189, 2008.

SILVA, A. M. R. da; VIDEIRA, C. A. E. **UML, metodologias e ferramentas CASE: linguagem de modelação UML, metodologias e ferramentas CASE na concepção e desenvolvimento de software**. [S.l.: s.n.], 2001.

SOMMERVILLE, I. **Engenharia de Software**. 6. ed. [S.l.]: Pearson Prentice Hall, 2011. ISBN 9788579361081.

SOUZA, J. N. de. **Lógica para ciência da computação**. [S.l.]: Elsevier Brasil, 2002.

STIRLING, C. Modal and temporal logics. In: ABRAMSKY, S.; GABBAY, D. M.; MAIBAUM, T. S. E. (Ed.). **Handbook of Logic in Computer Science**. Oxford: Oxford Science Publications, 1992. v. 2 (Background: Computational Structures), p. 477–563.

SWELLER, J. Cognitive load theory, learning difficulty, and instructional design. **Learning and instruction**, Elsevier, v. 4, n. 4, p. 295–312, 1994.

SYKES, E. R.; FRANEK, F. An intelligent tutoring system prototype for learning to program JavaTM. In: **Proc. 3rd IEEE Intl. Conf. Advanced Technologies**. [S.l.: s.n.], 2003.

TRIVIÑOS, A. N. S. Pesquisa qualitativa. **Introdução à pesquisa em ciências sociais: a pesquisa qualitativa em educação**. São Paulo: Atlas, p. 116–173, 1987.

VAHLDICK, A.; SANTIAGO, R. de; RAABE, A. L. A. Aplicação das técnicas de projeto instrucional 4C/ID na produção de objetos de aprendizagem em conformidade com o SCORM usando um software livre como ferramenta de autoria. **RENOTE**, v. 5, n. 2, 2007.

WOOLF, B. P. *et al.* Growth and maturity of intelligent tutoring systems: A status report. **Smart Machines in Education: The coming revolution in educational technology**, p. 99–144, 2001.

APÊNDICE A – DOCUMENTO DE REQUISITOS

Lista de requisitos não funcionais

Nº	Prioridade	Descrição
1	Importante	O sistema deve ser compatível com sistemas operacionais <i>Linux</i> e <i>Windows</i> .
2	Importante	A programação deve permitir a inclusão de novos menus e novos sistemas de prova com facilidade.
3	Essencial	O sistema deve apresentar boa usabilidade, com uma interface simples que torne fácil a execução de suas funções principais.

Lista de requisitos funcionais

Nº	Prioridade	Descrição
1	Importante	Para cada sistema operacional, deve haver um instalador executável, de modo que o usuário tenha que fazer <i>download</i> de apenas um arquivo para instalar o <i>software</i> em sua máquina.
2	Desejável	O formato dos menus deve seguir um padrão no formato <i>drop-down</i> e com ícones: Arquivo (para gerenciamento dos arquivos, como abrir e salvar), Sistemas de prova, Idioma (configurar idioma de exibição dos menus), Ajuda (com itens específicos para cada assunto), Sair.
3	Importante	As fórmulas lógicas devem ser representadas no formato de suas árvores sintáticas, para facilitar as operações entre elas.
4	Importante	O programa deverá implementar cálculos tanto sobre lógica proposicional quanto sobre lógica de primeira ordem. Deve haver possibilidade de estender o sistema para que contemple outras lógicas.
5	Essencial	O sistema deve ser capaz de salvar o perfil de um estudante, registrando suas interações.
6	Essencial	O sistema deve ser capaz de oferecer um caminho de aprendizado adequado ao desempenho do estudante.
7	Desejável	O sistema deve ter opção de escolha de idioma, com opções de português, inglês e espanhol.
8	Importante	O sistema deve ser capaz de salvar e restaurar trabalho já realizado.
9	Desejável	O sistema deve ser capaz de exportar as provas realizadas em formato PDF.
10	Essencial	O sistema deve apresentar exercícios de dedução natural aos estudantes.
11	Essencial	O sistema deve apresentar ao estudante material de apoio referente aos conceitos abordados nos exercícios.
12	Essencial	O sistema deve ser capaz de acompanhar a resolução dos exercícios pelo estudante, oferecendo ajuda quando necessário.
13	Importante	O sistema deve ser capaz de exportar o perfil do usuário em um formato de arquivo próprio, bem como ser capaz de importar perfis desse tipo.

APÊNDICE B – CASOS DE USO

CASO DE USO 1 – ESCOLHER O IDIOMA DO *SOFTWARE*

- Deve haver um menu para escolha do idioma dos menus do programa. O idioma escolhido deve ficar indicado por meio de uma marcação ao lado do item de menu.
- Quando um novo idioma for escolhido, todos os textos que aparecem nas telas devem trocar de idioma ao mesmo tempo.

CASO DE USO 2 – INICIAR UMA PROVA

- O usuário deverá clicar no menu de provas e escolher o item iniciar nova prova. Se um arquivo de provas já tiver sido aberto, a prova será feita sobre a lógica e o sistema de prova já determinados para aquele arquivo.
- Após a escolha, o usuário deverá entrar com o argumento que deverá ser provado, que inclui uma lista de premissas (separadas por vírgula) e uma conclusão. O programa deverá validar cada uma dessas fórmulas, para garantir que estão corretas do ponto de vista sintático. Se a fórmula estiver mal construída ela deverá ser escrita em fonte vermelha e aparecer um indicativo de erro (similar ao que aparece em uma fórmula errada no *Microsoft Excel*TM), que deve indicar o erro e também um link para “ajuda para este erro”, com explicações sobre o erro cometido.
- Após a inserção do argumento, o sistema deve construir, na janela de prova, o início da prova, com as premissas numeradas e indicadas dessa forma.
- O programa deverá então entrar no modo “execução de prova”, onde deverá ficar aguardando a aplicação das regras de inferência pelo usuário.

CASO DE USO 3 – APLICAR REGRA DE INFERÊNCIA

- A aplicação das regras de inferência deve ser feita com a indicação, pelo usuário, de qual regra deseja aplicar. A lista das regras de inferência do sistema de prova utilizado deverão estar na área da janela ao lado da prova.
- Quando o usuário indicar a regra desejada, por meio de um clique com o mouse, deverá abrir uma janela *pop-up* móvel para que ele entre com o número das linhas, na prova, nas quais estão as fórmulas que serão usadas na aplicação da regra de inferência. O número de elementos (linhas) deve ser igual ao número de fórmulas que são premissas da regra de inferência.
- No caso das linhas indicadas não corresponderem à estrutura das fórmulas espera-

das (por exemplo, caso a linha indicada para aplicação da regra de eliminação da conjunção não contenha uma conjunção), deverá ser apresentada uma janela explicativa do erro.

- O usuário deverá dar OK para fechar a janela explicativa.
- Caso as linhas indicadas correspondam, em estrutura, às fórmulas necessárias, a regra de inferência deverá ser aplicada, criando uma (ou duas, para os casos de eliminação da conjunção e da equivalência) nova linha na prova, numerada e com indicação de qual regra de inferência foi aplicada em quais linhas da prova.
- Hipóteses podem ser construídas, com a opção “criar hipótese”.
- Todas as opções de regras de inferência na área visível também devem estar disponíveis nos menus do sistema.

CASO DE USO 4 – ENCERRAR UMA PROVA

- Quando a fórmula encontrada na aplicação de uma regra de inferência for igual à conclusão do argumento, então deve aparecer uma janela *pop-up* dizendo que a prova foi encerrada e perguntando se deseja fechar o arquivo de provas ou fazer uma nova prova.
- Se o usuário desejar fazer uma nova prova, deve aparecer outra janela *pop-up* para que ele escolha a lógica e o sistema de prova. A lógica e o sistema usados na última prova devem aparecer como valor padrão.

CASO DE USO 6 – SALVAR UMA PROVA EM FORMATO PDF

- Solicitar ao usuário quais provas devem ser salvas no formato PDF.
- Perguntar o nome do arquivo PDF que será salvo.
- No caso do arquivo já existir, perguntar se quer substituir.

CASO DE USO 7 – RESTAURAR UM ARQUIVO DE PROVAS

- Abrir e colocar em memória todas as provas salvas.
- O usuário deve poder navegar pelas provas salvas, e o sistema deverá mostrá-las.
- Deve ser possível dar nome para a prova ou o sistema deve usar o argumento a ser provado como nome da prova.

CASO DE USO 8 – ESCOLHER SISTEMA DE PROVA

- O sistema de prova a ser usado deverá estar disponível no menu. Se o usuário escolher mudar o sistema no meio de uma prova, o programa deverá gerar uma

janela de *pop-up* e perguntar se o usuário deseja descartar a prova em andamento, reiniciar a prova (mantendo o argumento) ou cancelar.

- Se o usuário desejar descartar a prova, passa-se ao procedimento de iniciar uma nova prova, no novo sistema de prova escolhido. Esse procedimento deve ser adicionado ao registro de interações do programa.
- Se o usuário desejar reiniciar, mantém-se o argumento, mantêm-se as linhas iniciais da prova que representam as premissas e todas as linhas referentes a aplicações de regras de inferência são eliminadas. A prova continua no novo sistema de prova escolhido.
- Se o usuário escolher cancelar, nada acontece.

CASO DE USO 9 – INSTALAR O PROGRAMA

- A instalação do programa deverá ser construída por meio de uma sequência de menus que peçam ao usuário que indique as opções desejadas. Cada opção padrão já deve vir preenchida (no estilo *next-next-finish*). A primeira opção deverá ser a escolha da língua de instalação (que deve ser usada como padrão quando o usuário iniciar a execução do programa).
- O sistema deverá salvar as opções de configuração do usuário na instalação e a cada novo uso. Para isso pode fazer uso de arquivos de configuração, gravados na área do sistema em que o programa foi instalado.
- Todas as interfaces usuais de abertura e exploração de arquivos presentes no sistema operacional usado deverão estar disponíveis para o usuário do programa.
- Mostrar uma janela final indicando se o software foi instalado com sucesso.

CASO DE USO 10 - SELECIONAR NOVA PROVA

- O estudante pode selecionar uma nova prova, caso não queira resolver a selecionada atualmente. O abandono da prova atual será salvo no registro de interações do programa.

CASO DE USO 11 - SOLICITAR AJUDA

- O estudante pode solicitar ajuda, na forma de *feedback*, caso esteja com dificuldade na resolução da atividade atual.

CASO DE USO 12 - VER MATERIAL DE APOIO

- O estudante pode revisar o material de apoio associado à atividade em execução.

- O material de apoio é acessado através de um botão destacado na interface.
- O material de apoio é exibido por meio de uma janela *pop-up*.

CASO DE USO 13 - CADASTRAR PERFIL

- O estudante deve criar seu perfil na primeira utilização do programa.
- O estudante pode alterar seu perfil no menu de configurações.

CASO DE USO 14 - EXPORTAR E IMPORTAR PERFIL

- O usuário pode exportar seu perfil em um arquivo para que seja possível acessá-lo em outra máquina.
- O usuário pode importar seu perfil para restaurar as informações contidas no momento da exportação.

APÊNDICE C – CASOS DE TESTE

Caso de teste #01

Ação	Aplicar a introdução da conjunção entre duas fórmulas.
Pré-condição	A prova contém n linhas, onde $n \geq 2$.
Passos	1º: Selecionar botão [Introdução da conjunção]. 2º: Selecionar primeira fórmula (ϕ). 3º: Selecionar segunda fórmula (ψ). 4º: Selecionar botão [Aplicar].
Resultado esperado	Nova linha com a conjunção entre as fórmulas, no formato $\phi \wedge \psi$.
Pós-condição	A prova contém $n + 1$ linhas.

Caso de teste #02

Ação	Aplicar a eliminação da conjunção em uma fórmula.
Pré-condição	A prova contém n linhas, onde $n \geq 1$ e uma das linhas é uma conjunção (\wedge).
Passos	1º: Selecionar botão [Eliminação da conjunção]. 2º: Selecionar uma conjunção ($\phi \wedge \psi$). 3º: Selecionar uma das fórmulas (ϕ ou ψ) como resultado.
Resultado esperado	Nova linha com uma das fórmulas ϕ ou ψ .
Pós-condição	A prova contém $n + 1$ linhas.

Caso de teste #03

Ação	Aplicar a introdução da disjunção entre duas fórmulas.
Pré-condição	A prova contém n linhas, onde $n \geq 1$.
Passos	1º: Selecionar botão [Introdução da disjunção]. 2º: Selecionar primeira fórmula (ϕ). 3º: Digitar a segunda fórmula (ψ). 4º: Selecionar botão [Ok].
Resultado esperado	Disjunção entre as fórmulas, no formato $\phi \vee \psi$.
Pós-condição	A prova contém $n + 1$ linhas.

Caso de teste #04

Ação	Aplicar a eliminação da disjunção entre três fórmulas.
Pré-condição	A prova contém n linhas, onde $n \geq 3$, uma das linhas é uma disjunção (\vee) e duas das linhas são implicações (\rightarrow).
Passos	1º: Selecionar botão [Eliminação da disjunção]. 2º: Selecionar uma disjunção ($\phi \vee \psi$), uma implicação ($\phi \rightarrow \chi$) e uma implicação ($\psi \rightarrow \chi$). 3º: Selecionar botão [Aplicar].
Resultado esperado	Nova linha com a fórmula χ .
Pós-condição	A prova contém $n + 1$ linhas.

Caso de teste #05

Ação	Aplicar a introdução da implicação entre duas fórmulas.
Pré-condição	A prova contém n linhas, onde $n \geq 1$, e contém uma hipótese aberta.
Passos	1º: Selecionar botão [Introdução da implicação]. 2º: Selecionar primeira fórmula (ϕ). 3º: Digitar a segunda fórmula (ψ). 4º: Selecionar botão [Aplicar].
Resultado esperado	Implicação entre as fórmulas, no formato $\phi \rightarrow \psi$.
Pós-condição	A prova contém $n + 1$ linhas e a hipótese é encerrada.

Caso de teste #06

Ação	Aplicar a eliminação da implicação entre duas fórmulas.
Pré-condição	A prova contém n linhas, onde $n \geq 2$, uma das linhas é uma implicação (\rightarrow) e uma das linhas é o antecedente da implicação.
Passos	1º: Selecionar botão [Eliminação da implicação]. 2º: Selecionar uma implicação ($\phi \rightarrow \psi$) e uma fórmula correspondente a seu antecedente (ϕ). 3º: Selecionar botão [Aplicar].
Resultado esperado	Nova linha com a fórmula ψ .
Pós-condição	A prova contém $n + 1$ linhas.

Caso de teste #07

Ação	Aplicar a introdução da negação entre duas fórmulas.
Pré-condição	A prova contém uma hipótese aberta com n linhas, onde $n \geq 2$, e a última linha é uma contradição (\perp).
Passos	1º: Selecionar botão [Introdução da negação]. 2º: Selecionar uma fórmula (ϕ) e uma contradição (\perp) dentro da mesma hipótese aberta. 3º: Selecionar botão [Aplicar].
Resultado esperado	Nova linha com a fórmula $\neg\phi$.
Pós-condição	A prova contém $n + 1$ linhas e a hipótese é encerrada.

Caso de teste #08

Ação	Aplicar a eliminação da negação em uma fórmula.
Pré-condição	A prova contém n linhas, onde $n \geq 1$, e uma das linhas é uma dupla negação ($\neg\neg$).
Passos	1º: Selecionar botão [Eliminação da negação]. 2º: Selecionar uma dupla negação ($\neg\neg\phi$).
Resultado esperado	Nova linha com a fórmula ϕ .
Pós-condição	A prova contém $n + 1$ linhas.

Caso de teste #09

Ação	Iniciar uma hipótese.
Pré-condição	A prova contém n linhas, onde $n \geq 0$.
Passos	1º: Selecionar botão [Adicionar hipótese]. 2º: Digitar fórmula da hipótese. 3º: Selecionar botão [Ok].
Resultado esperado	Nova linha com a fórmula digitada.
Pós-condição	A prova contém $n + 1$ linhas e uma nova hipótese aberta.

Caso de teste #10

Ação	Exportar o perfil do usuário.
Pré-condição	–
Passos	1º: Selecionar item de menu [Arquivo] > [Exportar perfil]. 2º: Escolher diretório e nome do arquivo no seletor de arquivos. 3º: Selecionar botão [Salvar]. 4º: Selecionar botão [Ok] na caixa de mensagem de sucesso.
Resultado esperado	O arquivo de perfil é gerado no diretório escolhido.
Pós-condição	–

Caso de teste #11

Ação	Importar um perfil de usuário.
Pré-condição	Um arquivo de perfil foi exportado anteriormente.
Passos	1º: Selecionar item de menu [Arquivo] > [Importar perfil]. 2º: Selecionar botão [Ok] na caixa de confirmação. 2º: Selecionar o arquivo de perfil no seletor de arquivos. 3º: Selecionar botão [Abrir]. 4º: Selecionar botão [Ok] na caixa de mensagem de sucesso.
Resultado esperado	O progresso atual do estudante é excluído. O progresso registrado no perfil importado é carregado para o sistema.
Pós-condição	A atividade atual é recarregada de acordo com o novo perfil do sistema.

Caso de teste #12

Ação	Redefinir o perfil de usuário.
Pré-condição	–
Passos	1º: Selecionar item de menu [Arquivo] > [Redefinir perfil]. 2º: Selecionar botão [Ok] na caixa de confirmação. 4º: Selecionar botão [Ok] na caixa de mensagem de sucesso.
Resultado esperado	O progresso atual do estudante é excluído. Um perfil em branco é carregado no sistema.
Pós-condição	A atividade atual é recarregada de acordo com o novo perfil do sistema.

Caso de teste #13

Ação	Sair do programa.
Pré-condição	–
Passos	1º: Selecionar item de menu [Arquivo] > [Sair] ou botão [Fechar] (X) da janela do programa. 2º: Selecionar botão [Ok] na caixa de confirmação.
Resultado esperado	A execução do sistema é encerrada.
Pós-condição	–

Caso de teste #14

Ação	Iniciar Modo Livre.
Pré-condição	Não estar no Modo Livre.
Passos	1º: Selecionar item de menu [Opções] > [Modo Livre]. 2º: Inserir as premissas e conclusão da nova atividade, conforme instruções da mensagem exibida. 3º: Selecionar botão [Ok] na caixa de mensagem.
Resultado esperado	A nova atividade é exibida no Modo Livre.
Pós-condição	O sistema entra no Modo Livre, indicado pela coloração amarela nos botões e menus.

Caso de teste #15

Ação	Encerrar Modo Livre.
Pré-condição	O sistema deve estar no Modo Livre.
Passos	1º: Selecionar item de menu [Opções] > [Encerrar Modo Livre] ou botão [Encerrar Modo Livre], exibido após concluir uma atividade.
Resultado esperado	O sistema retorna ao modo normal, carregando a próxima atividade selecionada pelo Tutor.
Pós-condição	O sistema sai do Modo Livre, retornando para a coloração verde nos botões e menus.

Caso de teste #16

Ação	Chegar à resposta da atividade.
Pré-condição	–
Passos	1º: Inserir a sequência de regras de inferência necessárias para chegar à conclusão da atividade atual.
Resultado esperado	A caixa de mensagem de conclusão de atividade é exibida.
Pós-condição	O painel com as regras de inferência é ocultado e o painel de conclusão de atividade é mostrado.

Caso de teste #17

Ação	Abrir próxima atividade.
Pré-condição	O usuário chegou à resposta de uma atividade.
Passos	1º: Selecionar botão [Próxima atividade], exibido após concluir uma atividade.
Resultado esperado	A próxima atividade é carregada no sistema.
Pós-condição	A nova atividade exibida em sua condição inicial.

Caso de teste #19

Ação	Abrir material de ajuda.
Pré-condição	–
Passos	1º: Selecionar item de menu [Ajuda] > [Exibir ajuda]
Resultado esperado	A tela de Ajuda é exibida.
Pós-condição	–

Caso de teste #20

Ação	Abrir material de apoio.
Pré-condição	–
Passos	1º: Selecionar botão [Revisar material] na interface principal.
Resultado esperado	A tela de Material de Apoio é exibida.
Pós-condição	–

Caso de teste #21

Ação	Ver informações do menu "Sobre".
Pré-condição	–
Passos	1º: Selecionar item de menu [Ajuda] > [Sobre].
Resultado esperado	A tela de informações Sobre é exibida.
Pós-condição	–

APÊNDICE D – QUESTIONÁRIO DE VALIDAÇÃO

Para a realização da validação, foram sugeridos alguns passos de utilização do sistema com o objetivo de tornar o processo da validação o mais simples possível, cobrindo todas as funcionalidades principais. As etapas sugeridas para o teste foram:

1. Resolver as 3 primeiras atividades do sistema.
2. Tentar aplicar as regras de inferência de maneira inadequada, observando os *feedbacks* e as alterações no botão de Ajuda.
3. Inserir uma atividade no Modo Livre, por exemplo: Premissas: “a, $a > b$ ”; Conclusão: “b”.

O questionário de avaliação consistiu em um total de seis questões, três discursivas e três objetivas. As questões 3, 4 e 5 são objetivas e possuem as opções *concordo totalmente*, *concordo parcialmente*, *discordo parcialmente* e *discordo totalmente*. A resposta da questão 6 foi sinalizada como opcional.

1. Com base na utilização do sistema, quais características positivas podem ajudar os estudantes no estudo de Dedução Natural?
2. Quais características do sistema podem ser melhoradas ou acrescentadas?
3. Sobre a afirmação: "O sistema tem potencial para incentivar e facilitar o estudo extraclasse pelos alunos de graduação nas disciplinas relacionadas à lógica para computação."
4. Sobre a afirmação: "O sistema tem potencial para melhorar o desempenho dos alunos e reduzir desistências nas disciplinas com esses conteúdos."
5. Sobre a afirmação: "O sistema é uma alternativa útil aos alunos para o estudo sem dependência do professor, com a capacidade de avaliar seus próprios resultados."
6. Possui comentário ou considerações adicionais sobre o sistema?

Com as questões 1 e 2, o objetivo foi realizar um levantamento sobre as impressões positivas obtidas na utilização inicial do sistema, bem como as impressões negativas, como funcionalidades mal utilizadas ou ausentes. Já as questões discursivas – 3, 4 e 5 – têm a finalidade de obter uma impressão sobre a viabilidade do sistema de alcançar seus objetivos a longo prazo quando aplicado em situações reais de sala de aula. Os objetivos a longo prazo são que o sistema seja capaz de incentivar o estudo extraclasse de maneira eficaz, com a possibilidade de o desempenho dos estudantes em disciplinas

de lógica para computação. Finalmente, a questão 6 tem o objetivo de deixar um campo aberto para eventuais comentários que não se adequavam aos campos anteriores, bem como justificativas sobre alguma resposta dada anteriormente.