UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

CRISTIANO GALAFASSI

EVOLOGIC: SISTEMA TUTOR INTELIGENTE PARA ENSINO DE LÓGICA BASEADO EM COMPUTAÇÃO EVOLUTIVA

Porto Alegre
Abril de 2021

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
CENTRO INTERDISCIPLINAR DE NOVAS TECNOLOGIAS NA EDUCAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA NA EDUCAÇÃO

CRISTIANO GALAFASSI

EvoLogic: Sistema Tutor Inteligente para Ensino de Lógica Baseado em Computação Evolutiva

Tese apresentada ao Programa de Pós-Graduação em Informática na Educação do Centro Interdisciplinar de Novas Tecnologias na Educação da Universidade Federal do Rio Grande do Sul, como requisito para obtenção do título de Doutor em Informática na Educação.

Orientadora:
Dra. Rosa Maria Vicari

Coorientador:
Dr. Eliseo Berni Reategui

Porto Alegre
Abril de 2021

CIP - Catalogação na Publicação

Galafassi, Cristiano
EvoLogic: Sistema Tutor Inteligente para Ensino de Lógica Baseado em Computação Evolutiva / Cristiano Galafassi. -- 2021.
100 f.
Orientadora: Rosa Maria Vicari.

Coorientadora: Eliseo Berni Reategui.

Tese (Doutorado) -- Universidade Federal do Rio Grande do Sul, Centro de Estudos Interdisciplinares em Novas Tecnologias na Educação, Programa de Pós-Graduação em Informática na Educação, Porto Alegre, BR-RS, 2021.

1. Sistemas Tutores Inteligentes. 2. Algoritmos Genéticos. 3. Model Tracing. I. Vicari, Rosa Maria, orient. II. Reategui, Eliseo Berni, coorient. III. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da UFRGS com os dados fornecidos pelo(a) autor(a).

RESUMO

O uso de sistemas computacionais na educação vem demonstrando ser de grande valia para auxiliar no processo de ensino-aprendizagem. Uma das ferramentas disponíveis são os Sistemas Tutores Inteligentes, que tem o objetivo de dar suporte ao aluno em um determinado conteúdo. Em meados de 1980, surge o Cognitive Tutor®, para testar a arquitetura cognitiva ACT-R, se diferenciando dos Sistemas Tutores Inteligentes por possuir um viés cognitivo, destinado a acompanhar o processo cognitivo atrelado a resolução de problemas. Desde seu surgimento, o Cognitive Tutor® passou por inúmeras atualizações além de novos tutores cognitivos terem surgido. Contudo, todos recaem sobre o problema de disponibilizar um conjunto limitado de exercícios nativos, demandaram um grande esforço do professor para modelar novas atividades para os alunos ou apresentam um processo cognitivo limitado a certas formas de resolver o problema. Nesse sentido, o objetivo desse estudo é propor um Sistema Tutor Inteligente capaz de acompanhar o processo cognitivo do aluno no domínio de Dedução Natural em Lógica Proposicional, permitindo que seja possível identificar a linha de raciocínio dos alunos e que seja possível fornecer *feedback* de qualidade possibilitando que o aluno tenha flexibilidade na escolha dos exercícios que deseja resolver. Os agentes são modelados no contexto de um Sistema Tutor Inteligente denominado EvoLogic, utilizando uma arquitetura multiagente, suportada por cinco estruturas cognitivas estudadas ao longo dos anos. O sistema é composto por três agentes, destacando-se o agente Pedagógico, que tem a função de identificar as características do processo cognitivo do aluno (Modelo de Aluno), e o agente Especialista baseado em Algoritmos Genéticos. O objetivo deste estudo é a criação do EvoLogic, um Sistema Tutor Inteligente que possa acompanhar o processo cognitivo do aluno, flexibilizando a escolha dos exercícios e possuindo um *model tracing* adequado ao raciocínio do aluno. Sendo assim, optou-se pela utilização de um algoritmo genético como especialista do domínio de ensino dada a sua grande adaptabilidade a diversos contextos e problemas, facilitando uma possível portabilidade futura para outros domínios de ensino. Para validar o sistema proposto, foi criado um ambiente simulado com dados de um experimento realizado em outro sistema, especialista em Lógica. Os resultados apontam que o agente Especialista foi capaz de obter as diferentes soluções para os problemas estudados e que o agente Pedagógico pode identificar características do processo cognitivo dos alunos e acompanhar seu raciocínio ao resolverem os exercícios.

Palavras-chave: Algoritmos Genéticos. Sistema Multiagente. *Model Tracing*. Agente Especialista.

EvoLogic: Intelligent Tutor System for Teaching Logic Based on Evolutionary Computing

ABSTRACT

The use of computer systems in education has proven to be valuable to assist in the teaching-learning process. One of the available tools are the Intelligent Tutoring Systems, which aims to support the student in a certain content. In the mid-1960s, the Cognitive Tutor® appeared, to test the cognitive architecture ACT-R, differentiating itself from Intelligent Tutoring Systems in that it has a cognitive bias, designed to follow the cognitive process related to problem solving. Since its inception, Cognitive Tutor® has undergone numerous updates in addition to new cognitive tutors having emerged. However, they all fall on the problem of making a limited set of native exercises available, demanded a great effort from the teacher to model new activities for the students or present a cognitive process limited to certain ways of solving the problem. In this sense, the objective of this study is to propose an Intelligent Tutor System capable of monitoring the student's cognitive process in the domain of Natural Deduction in Propositional Logic, allowing it to be possible to identify the students' line of reasoning and to provide quality feedback enabling that the student has flexibility in choosing the exercises he wants to solve. The agents are modeled in the context of an Intelligent Tutoring System called EvoLogic, using a multi-agent architecture, supported by five cognitive structures studied over the years. The system is composed of three agents, especially the Pedagogical agent, which has the function of identifying the characteristics of the student's cognitive process (Student Model), and the Specialist agent based on Genetic Algorithms. The objective of this study is the creation of EvoLogic, an Intelligent Tutoring System that can follows the student's cognitive process, making the choice of exercises more flexible and having a model tracing consistent to the student's reasoning. Therefore, we opted for the use of a genetic algorithm as a specialist in the teaching domain, given its great adaptability to different contexts and problems, facilitating possible future portability to other teaching domains. To validate the proposed system, a simulated environment was created with data from an experiment carried out in another system, specialized in Logic. The results show that the Specialist agent was able to obtain the different solutions to the studied problems and that the Pedagogical agent can identify characteristics of the students' cognitive process and monitor their reasoning when solving the exercises.

Keywords: Genetic Algorithms. Multiagent System. Model Tracing. Specialist Agent.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ACT	Adaptive Control of Thought
ACT-R	Adaptive Control of Thought-Rational
AG	Algoritmo Genético
DNPL	Dedução Natural em Lógica Proposicional
EaD	Ensino a Distância
EBL	Example Based Learning
FIFO	First In, First Out
FIPA-ACL	Foundation for Intelligent Physical Agents - Agent Communication Language
STI	Sistema Tutor Inteligente
UML	Unified Modeling Language

SUMÁRIO

1. INTRODUÇÃO

Em 2014 foi aprovado o Plano Nacional de Educação através da Lei 13005 de junho de 2014 (Brasil, 2014) que versa sobre o ensino país. Dentre as 20 metas definidas no plano, a meta 12 versa sobre o compromisso que o país possui com a democratização do acesso à educação superior, garantida a manutenção da qualidade. Para atingir tal objetivo, as estratégias sugerem elevar a taxa bruta de matrícula na educação superior em 50% e a taxa líquida em 33% da população de 18 a 24 anos, assegurando a qualidade da oferta e a expansão para, pelo menos, 40% das novas matrículas, no segmento público.

Com base nos dados disponíveis no InepData[1], o número total de vagas oferecidas no ensino superior, público e privado, passou de 5.145.973 para 10.793.807 (referente aos anos de 2009 e 2017, respectivamente), representando um aumento superior a 100%. Os benefícios do aumento no número de alunos no ensino superior estimulam a inovação, a produção de bens e serviços especializados e favorecem a promoção da justiça social. Contudo, o ingresso não garante esse ganho se vier acompanhado das condições que contribuam para a permanência dos estudantes, do ingresso à formatura.

Junto a esse aumento no número de vagas e ingressos, pode-se verificar um elevado índice de reprovação e evasão, em especial nas turmas de semestre iniciais. Aliado a isso, o cenário de pandemia causado pelo Sars-Cov-2 (Corona Vírus – Covid-19) no ano de 2020 impactou diversos setores da vida cotidiana, incluindo todos os níveis de ensino. Ensino a Distância (EaD) é uma realidade em diversos cursos de graduação, tanto públicos quanto privados. Contudo, o novo cenário impôs que até mesmo os cursos superiores pensados para ocorrerem de forma presencial, tenham de ser abordados de forma remota. Isso também ocorreu com a educação infantil e os ensinos fundamental e médio. Apesar de existir um grande debate acerca das disparidades sociais e do acesso aos materiais didáticos (que não serão discutidos neste trabalho), inúmeros veículos de notícia apontam que o ensino híbrido tende a ser inserido, de alguma forma, no cenário educacional pós pandemia.

Em muitos casos, esses problemas podem ser mitigados através do uso de ambientes computacionais de aprendizagem que forneçam suporte o professor e o aluno no processo de ensino-aprendizagem. Esses ambientes, em sua maioria, possuem limitações quanto a representar um modelo de aprendizagem que expresse o que ocorre durante as interações entre alunos e professores. De acordo com Galafassi (2019b), tradicionalmente, os ambientes computacionais reproduzem o modelo de sala de aula presencial, especialmente, no que tange ao uso do modelo de trabalho individual e em grupo. Isso se dá, em parte, pelas restrições dos softwares e pelo desconhecimento em como modelar vários destes aspectos computacionalmente, em especial, o diagnóstico cognitivo do aluno.

Nesse contexto, os Sistemas Tutores Inteligentes (STI) surgiram como uma alternativa para lidar com essas questões, possibilitando a criação de modelos de representação do conhecimento do aluno, a implementação de estratégias de ensino-aprendizagem e o desenvolvimento de algoritmos especialistas no domínio de ensino. Weitekamp *et al.* (2020) destacam que enquanto um aluno pode aprender apenas uma forma de resolver um problema, sendo suficiente para ele, um STI necessita aprender todas as formas de resolver o mesmo problema para ser capaz de auxiliar os demais alunos.

[1] http://portal.inep.gov.br/inep-data

Os STIs se apresentam como uma possível abordagem para esse cenário pois, de acordo com Paviotti et al. (2012), permitem:

- monitorar, acompanhar e avaliar o que o aluno faz no ambiente de aprendizagem (ou seja, páginas e documentos acessados/baixados);
- analisar os textos produzidos dos alunos;
- gerenciar fontes de conhecimento, por exemplo, construindo representações de conhecimento por meio de mapas conceituais ou taxonomias;
- representar o conhecimento formal e não formal do aluno à medida que ele se desenvolve dentro do curso;
- avaliar e fornecer e feedback formativo.

Essas características possibilitam o planejamento eficaz do processo de ensino/aprendizagem e visam a aprendizagem personalizada, centrada no aluno. Contudo, ao considerar o número de alunos envolvidos no EaD (completo ou híbrido), a possibilidade de realizar plenamente as atividades necessárias para acompanhar os alunos, realizando um planejamento adequado, exige uma enorme carga de trabalho.

Na década de 80 surgiu uma tentativa de criar um modelo do processo cognitivo do aluno, bem como acompanhar seu aprendizado, através da arquitetura cognitiva ACT-R (*Adaptive Control of Thought-Rational*) (ANDERSON, 1983). O ACT-R apresenta diversas estruturas cognitivas que são utilizadas para modelar o comportamento do sistema, simulando o processo cognitivo. Junto à proposição da arquitetura surgiu o primeiro Sistema Tutor Cognitivo (especialização dos Sistemas Tutores Inteligentes), denominado Cognitive Tutor® (ANDERSON et al., 1989), com o objetivo de testar e validar a arquitetura ACT-R. Esse tutor possuía um modelo de aluno ideal, baseado em regras de produção, no contexto de programação em LISP, que comparava os passos do aluno com seu modelo e fornecia *feedback* ao aluno.

1.1 Justificativa

Em sua concepção, os Sistemas Tutores Cognitivos (ou os modelos cognitivos existentes em alguns STIs) podem auxiliar o aluno em sua aprendizagem, pois buscam acompanhar o seu processo cognitivo e prover *feedbacks* condizentes com a situação atual da resolução do problema. No entanto, de acordo com Murray (2003), os STIs são notoriamente difíceis de se construir e demandam muito tempo. Os autores também apresentam um exemplo onde o professor irá modelar o processo cognitivo dos alunos no contexto de ensino de adição de várias colunas (adição de grandes números). Ao utilizar regras de produção, o professor precisa considerar todos os possíveis passos do aluno, escrevendo regras de produção para adicionar números, carregar dígitos 1 e colocar suas respostas em locais predefinidos em uma interface apropriada.

De acordo com Weitekamp et al., (2020), processos similares ao descrito anteriormente, podem levar de 200 a 300 horas de tempo do desenvolvedor por hora de tempo de instrução. Em outras palavras, a principal dificuldade está em modelar os problemas a serem resolvidos pelo aluno, uma vez que a complexidade por trás da tarefa exige que, em muitos casos, o professor (ou o designer de conteúdo) invista muito tempo para modelar o design instrucional. Além do tempo de desenvolvimento, alguns STIs apresentam restrições quanto à fidelidade cognitiva, ao não modelar todos os possíveis cenários por onde o aluno pode seguir. Dependendo das características do problema a ser abordado, podem existir diferentes linhas de

raciocínio que o aluno pode seguir e é importante que o STIs seja capaz de acompanhar e fornecer os *feedbacks* necessários em cada uma delas.

Apesar do grande tempo investido, os desafios associados à satisfação das necessidades do aluno dentro das restrições do design podem resultar em comprometimentos na flexibilidade e na fidelidade cognitiva. Para endereçar esse problema, este trabalho apresenta os objetivos a seguir.

1.2 Objetivos

Com base no cenário atual, o objetivo desse estudo é propor e desenvolver o STI EvoLogic, composto pelos agentes Interface, Modelo de Aluno e Especialista, para ensino de Dedução Natural em Lógica Proposicional (DNPL), que siga os passos do aluno na resolução de problemas (*model tracing*) e obtenha subsídios (na forma de diferentes soluções possíveis para o mesmo exercício) para fornecer *feedbacks* que guiem o aluno em seu próprio processo cognitivo.

1.2.1 Objetivos Específicos

Dentre os objetivos específicos, estão:

- Modelar o STI com um sistema multiagente, suportado por elementos das arquiteturas cognitivas que mais se adequem ao objetivo deste trabalho;
- Desenvolver um agente Interface capaz de interagir, através da troca de mensagens, com o provador de teoremas, possibilitando a adaptação e integração com diversas ferramentas existentes.
- Desenvolver um agente Especialista, baseado em computação evolutiva (Algoritmo Genético - AG), capaz de resolver qualquer exercício proposto pelo aluno no contexto de Dedução Natural em Lógica Proposicional;
- Desenvolver um agente Modelo de Aluno que, utilizando os subsídios gerados pelo agente Especialista, possa identificar e acompanhar o processo cognitivo do aluno (*model tracing*) sem que haja a necessidade de modelar, para cada exercício, todo o processo cognitivo. Ressalta-se que é importante garantir a flexibilidade, para o aluno em tentar resolver exercícios que não façam parte da base de dados do EvoLogic;
- Propor estratégias pedagógicas baseadas na teoria de Aprendizagem Baseada em Exemplos, explorando as capacidades e características do agente Especialista;
- Validar e avaliar o desempenho do EvoLogic comparando-o com o outro STI para ensino de lógica.

1.3 Organização

O presente trabalho está organizado no que segue. No Capítulo 2 são apresentados os conceitos que permeiam as áreas de Inteligência Artificial e Educação, se iniciando por definições conceitos acerca da IA e como ela se entrelaça com áreas de Educação e Psicologia, dando origem aos Sistemas Tutores Inteligentes, que também são abordados. Após a discussão

sobre os STIs, são apresentados os conceitos relacionados com as Arquiteturas Cognitivas, que suportam a criação dos Sistemas Tutores Cognitivos e também do STI proposto neste trabalho. Ainda acerca do referencial teórico, também se discute o tema de Computação Evolutiva, a qual engloba uma série de métodos, em especial, os Algoritmos Genéticos, tema central do agente Especialista aqui proposto. Além disso, neste capítulo, são apresentados os trabalhos relacionados a esta pesquisa, iniciando pelos Sistemas Tutores Cognitivos, enfatizando as características dos seus *model tracing*. Em seguida, aborda-se os trabalhos relacionados com o domínio de ensino desta pesquisa, Dedução Natural em Lógica Proposicional. Esses trabalhos se dividem entre provadores de teoremas e STIs, dentre os quais são comparadas e evidenciadas suas características.

No Capítulo 3 são apresentados os detalhes técnicos acerca do STI EvoLogic, discutindo sua arquitetura multiagente, suportada por estruturas cognitivas. Detalha-se a forma de comunicação do STI com o ambiente externo (através do agente Interface), as estratégias pedagógicas e o *model tracing* do agente Pedagógico, e o funcionamento do Algoritmo Genético que compõem o agente Especialista.

No Capítulo 4 apresenta-se o planejamento de experimentos. Discute-se, nesse contexto, a possibilidade de avaliar as capacidades do agente Especialista e do *model tracing*, utilizando dados reais, coletados em um experimento prévio.

No Capítulo 5 são mostrados os resultados obtidos acerca do agente Especialista, analisando o comportamento do algoritmo e a capacidade dele na obtenção de diferentes soluções para um mesmo problema (característica fundamental para que o *model tracing* possa ser realizado de forma autônoma). Em seguida, avalia-se o processo de *model tracing*, i.e., a capacidade do agente Pedagógico em acompanhar os passos do aluno, identificar sua linha de raciocínio e responder as necessidades do aluno conforme as estratégias pedagógicas.

Por fim, no Capítulo 6, apresenta-se as conclusões, evidenciando as contribuições desta pesquisa e sugere-se trabalhos futuros para melhorar a ferramenta.

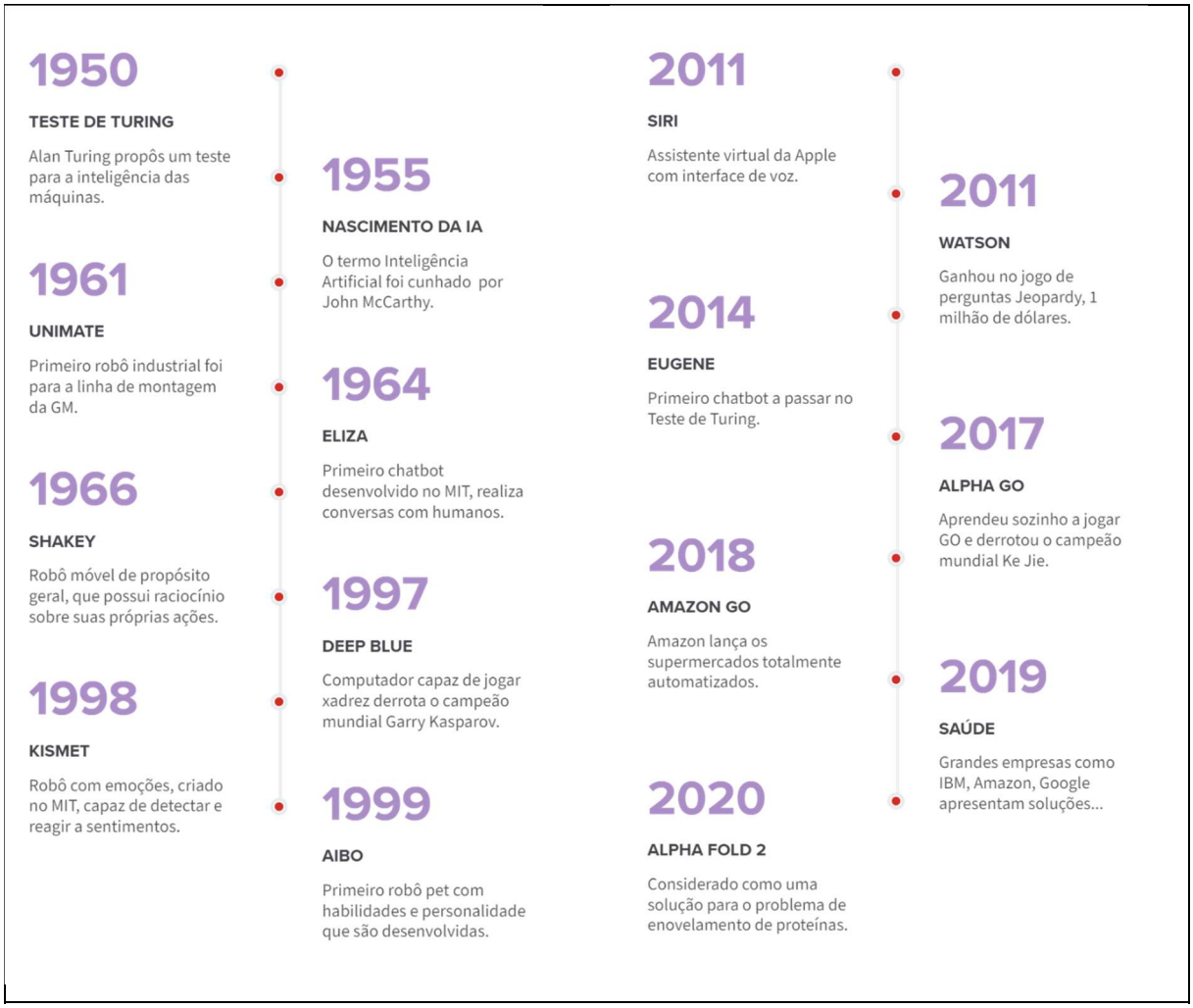
2. REFERENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos principais que permeiam essa proposta. Neste trabalho utilizaram-se referencias como Self, Woolf e Nwana, que embora sejam referências um tanto antigas, apresentam ideias atuais. Ressalta-se que a arquitetura de um STI não passou por mudanças significativas nos últimos anos. As mudanças estão centradas na forma como estes vêm sendo desenvolvidos, utilizando uma ou mais técnicas de IA. De forma geral, todas as tecnologias aqui mencionadas nos sistemas computacionais educacionais, são as mesmas que fazem parte de várias áreas da computação, muitas delas advindas da IA, porém, aplicadas à educação.

2.1 IA & Educação

A IA é uma área de pesquisa e desenvolvimento da Ciência da Computação. Segundo os textos publicados por Alan M. Turing (1950) pode-se dizer que a IA teve seu início durante a segunda guerra mundial e vem evoluindo, promovendo inovações até os dias de hoje. Uma linha do tempo da IA é mostrada na Figura 2.1, onde são relacionados alguns dos avanços mais significativos desde o seu surgimento.

Figura 2.1 - Linha do tempo da Inteligência Artificial



Recentemente, nos anos 2000, as notícias que se refere à IA ultrapassaram os limites da academia e chegaram a diversos meios de comunicação, atingindo o público em geral. Contudo, a IA não chega ao grande público apenas através dos jogos que marcaram o período. Cada vez mais a IA se aproveita do grande poder de processamento que a evolução do hardware proporciona, como a capacidade das máquinas de traduzir qualquer idioma tanto em texto quanto em voz (em tempo real) ou de ser capaz de diagnosticar câncer (IBM Watson). Por outro lado, muitos condenam o uso de algoritmos fechados para sentenciar réus (FORD, M. 2015), autor que estuda que tipo de regras as máquinas precisam utilizar, nos algoritmos que operam de forma autônoma, na área de tomada de decisões jurídicas) e ou uso bélico da IA (ANGWIN et al., 2016).

Como pode ser visto na Figura 2.1, a IA engloba uma enorme variedade de subáreas, indo desde as mais gerais até as mais específicas como, por exemplo, aprender, perceber o ambiente e responder eficientemente a ele, jogar xadrez, provar teoremas matemáticos, escrever poesia, dirigir um carro em uma rua movimentada, fazer traduções simultâneas e até mesmo diagnosticar doenças.

Tendo como um dos principais objetivos (o qual vem desde a sua origem), o de ultrapassar a barreira da compreensão de como se dá o raciocínio humano, McCarthy (2007) e Rich e Knight (1994), definiram a IA como sendo "o estudo de como fazer computadores realizarem coisas que, atualmente, os humanos fazem melhor". Desta forma, a IA busca, entre outros objetivos, desenvolver sistemas que possam simular a capacidade humana de raciocínio, percepção e tomada de decisão para a resolução de problemas. Em alguns casos, busca-se desenvolver sistemas que possam ser considerados inteligentes. Vale ressaltar que o conceito de sistema inteligente remete a uma discussão filosófica extensa e que não se adequa ao contexto desta proposta. Ao longo do tempo, outros autores também buscaram definições de IA que são apresentadas em quatro categorias (RUSSEL e NORVIG, 2010):

1. Pensando Humanamente:

"[A automação de] atividades que nós associamos com o pensamento humano, atividades como a tomada de decisões, resolução de problemas, aprendendo ..." (Bellman, 1978);

"O novo e empolgante esforço para fazer computadores pensar ... máquinas com mentes, no sentido completo e literal" (Haugeland, 1985).

2. Pensando Racionalmente:

"O estudo das faculdades mentais através do uso de modelos computacionais" (Charniak e McDermott, 1985).

"O estudo dos cálculos que realizam é possível perceber, raciocinar e agir" (Winston, 1992).

3. Atuando Humanamente:

"A arte de criar máquinas que executam funções que exigem inteligência quando realizada por pessoas" (Kurzweil, 1990).

"O estudo de como fazer computadores fazerem coisas que, no momento, as pessoas fazem melhor" (Rich e Knight, 1994).

4. Atuando Racionalmente

"Inteligência Computacional é o estudo do design de agentes inteligentes" (Poole et al., 1998).

"IA ... está preocupada com o comportamento inteligente em artefatos" (Nilsson, 1998).

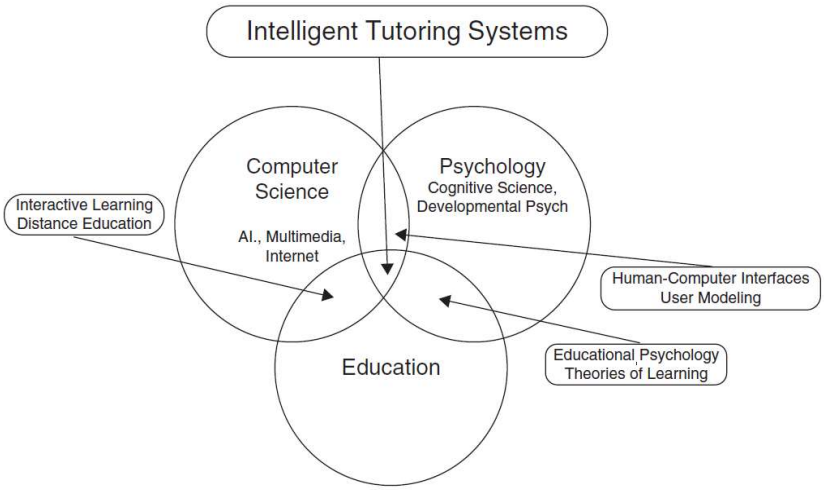
Historicamente, todas as quatro abordagens da IA foram seguidas, cada uma delas por diferentes pesquisadores e com diferentes métodos. Uma abordagem centrada no ser humano deve ser em parte, uma ciência empírica, envolvendo observações e hipóteses sobre o comportamento humano. Já uma abordagem racionalista envolve uma combinação de matemática e engenharia.

De acordo com Woolf (2009), a IA relaciona-se com aquisição e manipulação de dados para produzir conhecimento e reproduzir um comportamento inteligente. A IA se preocupa em criar modelos computacionais que contemplem as mais variadas atividades como: falar, aprender, andar e brincar, e em replicar as tarefas do senso comum como: compreender a linguagem escrita ou falada, reconhecer padrões visuais, servir de apoio para a educação, entre outras.

Neste último campo, além de fornecer ferramentas para impactar diretamente na forma de aprender, a IA aplicada à educação agrega ainda outros valores importantes. De forma mais específica, a IA pode contribuir decisivamente para uma maior motivação dos alunos quanto à rotina de aprender, proporcionar autonomia, auxiliar o trabalho do professor colocando ferramentas à disposição dos docentes e, ao mesmo tempo, otimizar a dinâmica de suas aulas presenciais e a distância. Uma característica do campo da IA e da educação é usar a inteligência para raciocinar sobre o ensino e a aprendizagem. Representar o quê, quando e como ensinar requer o embasamento de várias áreas de atuação, incluindo a Ciência da Computação, a Psicologia e a Educação.

Muitos dos métodos e conceitos que permeiam as áreas de Ciência da Computação, Psicologia e Educação são complementares e fornecem, de forma coletiva, uma cobertura quase completa do campo da IA e da educação. A interligação entre essas áreas pode ser representada na Figura 2.2 e também demonstra onde os Sistemas Tutores Inteligentes, que são um campo de pesquisa e desenvolvimento interdisciplinar da IA.

Figura 2.2 - Interligação entre as áreas da computação, educação e psicologia e onde convergem os STIs da IA.



Fonte: Woolf (2009).

A IA tem o foco em raciocinar sobre a inteligência e em aprender, seja sobre um conceito ou sobre um aluno, tendo como apoio as ferramentas voltadas ao ensino como, por exemplo, os

STIs. A Psicologia, particularmente a ciência cognitiva[2], aborda como as pessoas pensam e aprendem, e a educação se concentra em como melhor apoiar este ensino e aprendizagem.

A aprendizagem e o ensino de seres humanos consistem em atividades complexas, sendo imprescindível que, para desenvolver um sistema computacional para o ensino (um dos objetivos da IA), ele seja apoiado em teorias subjacentes da aprendizagem (os objetivos da educação e da ciência cognitiva[3]).

Visto isso, pode-se verificar que a Ciência Cognitiva possui grande influência em diversas áreas do conhecimento e investiga como entidades inteligentes, sejam elas humano ou computador, interagem com seu ambiente, adquirem conhecimento, lembram e usam o conhecimento, tomam decisões e resolvem problemas. A modelagem cognitiva na área da aprendizagem tem contribuído com teorias pedagógicas, teorias de aprendizagem, design instrucional e melhor distribuição instrucional (ANDERSON et al., 1995). Os resultados da ciência cognitiva, incluindo métodos empíricos, fornecem uma compreensão mais profunda da cognição humana, acompanhando assim a aprendizagem humana e apoiando a aprendizagem flexível.

Apesar da Ciência Cognitiva abranger uma área mais extensa que a IA, foi a partir do desenvolvimento da IA, nas últimas décadas, que toda a ideia de uma ciência da mente se desenvolveu. A IA proporcionou o passo fundamental para se tentar relacionar mentes e computadores e estabelecer o que passamos a chamar de "modelo computacional da mente" (TEIXEIRA, 1998). É importante salientar que, para Gardner (1996, pg. 19), a ciência cognitiva é "um esforço contemporâneo, com fundamentação empírica, para responder questões epistemológicas de longa data – principalmente aquelas relativas à natureza do conhecimento, seu desenvolvimento e seu emprego". Nesse sentido, a Figura 2.3 traz a representação da Ciência Cognitiva e a inter-relação da mesma com outras disciplinas (e áreas) na visão de Teixeira (1998) e Gardner (1996).

Figura 2.3 - Representação da Ciência Cognitiva de acordo com os autores.

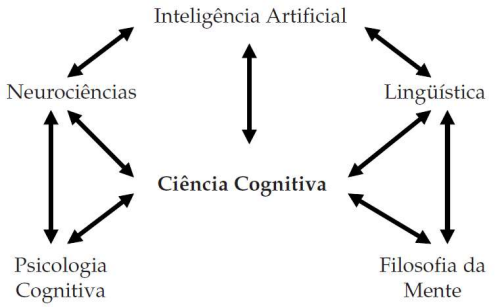
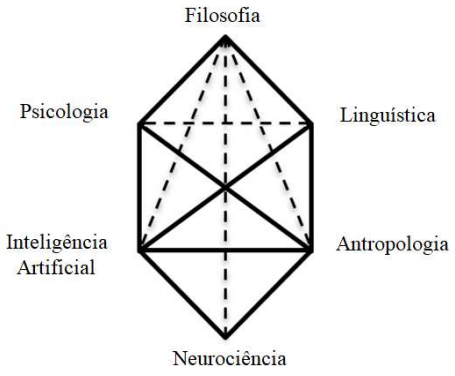


Diagrama das inter-relações entre algumas disciplinas que compõem a Ciência Cognitiva.

Fonte: Teixeira (1998).



Linhas cheias – fortes vínculos interdisciplinares.
Linhas tracejadas - fracos vínculos interdisciplinares.

Fonte: Gardner (1996).

[2] Neste trabalho a ciência cognitiva a que se faz referência, é a que traz a ideia de se poder estudar os processos mentais à luz de um modelo computacional.

[3] O desafio de simular computacionalmente processos mentais humanos requeria a contribuição de todos aqueles que, direta ou indiretamente, estivessem envolvidos com o estudo da mente: psicólogos, linguistas, filósofos, neurólogos, etc. Este esforço interdisciplinar levou à consolidação do que mais tarde ficou conhecido como Ciência Cognitiva (TEIXEIRA, 1998).

Na Figura 2.3, pode-se perceber que a IA é uma área fundamental na Ciência Cognitiva. Nesse contexto, um dos campos de pesquisa da IA se preocupa em projetar sistemas que exibam características inteligentes, como aprender, raciocinar, resolver problemas e entender a linguagem.

Segundo Galafassi (2019b), para apoiar a IA nessa tarefa, a educação possui um papel fundamental, uma vez que ela se preocupa em compreender e apoiar o ensino, principalmente em escolas. Ela se concentra em como as pessoas ensinam e como a aprendizagem é impactada pela comunicação, desenho do curso e currículo, avaliação e motivação. Um objetivo de longo prazo da educação é o de produzir um ensino acessível, eficiente e eficaz. Dentro de cada teoria da aprendizagem, conceitos como memória e estratégias de aprendizagem são abordados de forma diferente. Teorias específicas são frequentemente desenvolvidas para domínios específicos, como o ensino de ciências. Os métodos de educação incluem formas de melhorar a aquisição, manipulação e utilização do conhecimento e as condições sob as quais a aprendizagem ocorre. Os educadores podem avaliar as características da retenção de conhecimento usando ciclos de projeto e testes. Eles geralmente geram uma intervenção - uma circunstância ou ambiente para apoiar o ensino e, em seguida, testam se ela tem um efeito de aprendizado duradouro.

Para desenvolver ferramentas que apoiem o ensino e a aprendizagem, diversas tecnologias e arquiteturas computacionais podem ser utilizadas. Dentre elas, ressalta-se o uso de agentes de software que, devido a sua dinamicidade, suportam o desenvolvimento de muitos sistemas tutores inteligentes. A próxima seção apresenta os conceitos que permeiam os STIs, os quais servem de inspiração para a criação dos sistemas tutores cognitivos, apresentados na seção seguinte.

2.2 Sistemas Tutores Inteligentes

Os Sistemas de Instrução Assistida por Computador, que surgiram na década de 60, e serviram como base para os STIs, eram dotados de modelos com teorias cognitivas e estratégias de ensino-aprendizagem mais flexíveis. Essas técnicas buscam ajudar no processo de ensino e aprendizagem procurando entender como os alunos aprendem (SELF, 1990; GLUZ et al., 2013). De acordo com Nwana (1990), conceitualmente, STIs são programas de computador projetados para incorporar técnicas de IA, a fim de produzir um sistema de ensino-aprendizagem que:

- Conhecem o que ensinam;
- Sabem como ensinar;
- Acompanham o processo de ensino-aprendizagem do aluno.

Nos três aspectos que caracterizam os STIs, ainda de acordo com Nwana (1990), o trabalho de concepção, projeto e desenvolvimento de um STI envolve uma equipe multidisciplinar. Esta premissa é verdadeira para todos os sistemas que utilizam estratégias pedagógicas baseadas em alguma teoria educacional. A equipe, muitas vezes, é composta por cientistas da computação, pedagogos, designers de interface e especialistas no conteúdo que será abordado.

Em seu desenvolvimento são utilizadas várias tecnologias da IA dependendo de sua finalidade. No entanto, independentemente da tecnologia utilizada, esses sistemas apresentam como componentes básicos as seguintes estruturas:

- Modelo de aluno[4] (conhecimento do aluno sobre o tema de quem vai aprender e, em muitos casos, também o estado emocional do estudante);
- Modelo pedagógico[5] (estratégias e táticas pedagógicas – como será ensinado);
- Modelo especialista[6] (conhecimentos sobre o que será ensinado), e;
- Modelo de Interface (meio para interação do aluno com o sistema – representa os métodos de comunicação entre alunos e computadores, como interfaces gráficas, agentes animados, etc.).

Conforme Galafassi (2019b), esses sistemas alcançam altos níveis de detecção acerca do conteúdo educacional onde um aluno pode apresentar problemas de aprendizagem (estado cognitivo). Também é possível interferir de uma forma especializada, por exemplo, fazendo uso de um estilo de aprendizagem específico, associado a abordagens direcionadas ao estado cognitivo e emocional desse aluno, e em como apresentar o conteúdo (o que se vai ensinar) da forma mais adequada para cada indivíduo.

Com estas características um STI pode oferecer ajuda em qualquer momento, durante a realização do exercício: seja possibilitando ao aluno resolver problemas propostos passo a passo (validando o aprendizado), bem como propor possíveis soluções de problemas a partir de qualquer ponto da resolução, seja através de exemplos ou dicas pedagógicas que apontam o próximo passo.

Do ponto de vista tecnológico, pesquisas em STIs estão cada vez mais ligadas ao modelo de aluno, através do uso de tecnologias como *Learning Analytics*, Afetividade, e ao modelo de aptidões de cada estudante. A integração desses modelos, que já existem de forma isolada em diferentes STIs depende apenas do avanço da tecnologia voltada para a computação.

Como mencionado anteriormente, a IA aplicada a esses sistemas educacionais está voltada para o ensino personalizado (contrapondo o ensino colaborativo) e tem sua maior aplicação em STIs (SELF, 1990; GLUZ et al., 2013), dentre outros. Mais recentemente, aplicações da IA à educação se fazem presentes também na chamada aprendizagem ativa e suas implementações, como acontece na proposta "sala de aula invertida[7]", modelo que tem suas raízes no ensino híbrido, e nos "*Fab labs*[8]".

---

[4] O modelo de aluno contém as informações sobre quais são os conhecimentos do aluno, seu estilo cognitivo, suas habilidades e dificuldades. O modelo de aluno serve para uma melhor adaptação do tutor ao aluno. Este módulo pode levar o STI a classificar diversos níveis de aluno, desde o mais inicial, até o mais especialista de um determinado domínio de ensino.

[5] Estratégias e táticas de ensino-aprendizagem correspondem ao módulo pedagógico e determinam como o conhecimento é apresentado, as formas e maneiras de expor o conteúdo, as estratégias e táticas didáticas que melhor se adaptam ao aluno.

[6] No geral, são sistemas especializados em um determinado conteúdo. Podem ser utilizados em todos os níveis do ensino formal e não formal e em aulas presenciais e semipresenciais. É uma herança dos tradicionais sistemas especialistas de aplicações da IA, muito populares na década de 1970.

[7] A proposta do *flipped classroom*, é que o aluno absorva o conteúdo através do meio virtual e ao chegar à sala presencial ele já esteja ciente do assunto a ser desenvolvido (mais informações podem ser encontradas no endereço: https://www.edools.com/sala-de-aula-invertida/.

[8] *Fab Labs* são um espaço em que pessoas de diversas áreas compartilham a paixão por empreendedorismo e inovação. Nestes *labs* todos se reúnem para realizar projetos de fabricação digital de forma colaborativa. Mais informações podem ser vistas no FAB LAB LIVRE SP, que é um exemplo de *Fab Lab*, no endereço: http://fablablivresp.art.br/o-que-e.

2.3 Tutores Cognitivos

Um Tutor Cognitivo é um tipo particular de STI que utiliza um modelo cognitivo para fornecer *feedback* aos alunos enquanto eles estão trabalhando na resolução de um determinado problema. Esse *feedback* informará imediatamente os alunos sobre as características de suas ações (correção, inexatidão) na interface do tutor. Além das informações acerca da corretude da ação, os tutores cognitivos fornecem orientação passo a passo à medida que o aluno desenvolve uma habilidade complexa de solução de um dado problema (VANLEHN, 2006). Tipicamente, esses tutores fornecem formas de apoio como: (a) um ambiente de solução de problemas; (b) feedback passo a passo sobre o desempenho do aluno; (c) mensagens de feedback específicas para erros; (d) dicas do próximo passo específicas do contexto a pedido do aluno e (e) seleção individualizada de problemas (ALEVEN, 2010).

Os tutores cognitivos realizam duas das principais tarefas características do ensino humano: (1) monitoram o desempenho do aluno e fornece instruções individuais específicas ao contexto; e (2) monitoram o aprendizado do aluno e seleciona atividades apropriadas de solução de problemas (KOEDINGER et al., 2006).

O nome Tutor Cognitivo geralmente se refere a um tipo particular de STI produzido pela Carnegie Learning, aplicado no domínio de matemática para o ensino médio, com base na teoria ACT-R da cognição humana de John Anderson (ANDERSON et al., 1995). No entanto, os tutores cognitivos foram originalmente desenvolvidos para testar a teoria do ACT-R desde o início dos anos 80 e se aplicam também para outras áreas e domínios de ensino, como programação e ciências (ANDERSON et al., 1995). Os tutores cognitivos podem ser implementados nas salas de aula como parte do aprendizado combinado, que mesclam atividades de livros didáticos e de software.

O Tutor Cognitivo da Carnegie Learning se utiliza de um modelo cognitivo e são baseados no *model tracing* e no *knowledge tracing*. *Model tracing* significa que o Tutor Cognitivo verifica todas as ações executadas pelos alunos, como digitar um valor ou clicar em um botão, enquanto o *knowledge tracing* é usado para calcular as habilidades que os alunos aprenderam medindo-as em um gráfico de barras chamado *Skillometer*. O *model tracing* e o *knowledge tracing* são essencialmente usados para monitorar o progresso do aprendizado dos alunos e orientá-los a corrigir o caminho para a solução de problemas, fornecendo *feedbacks*.

Em geral, um Tutor Cognitivo se baseia em um modelo cognitivo (também chamado de arquitetura cognitiva), o qual busca modelar o conhecimento do domínio de ensino da mesma forma que o conhecimento é representado na mente humana. O modelo cognitivo permite que os STIs resolvam problemas como o aluno faria (CORBETT et al., 1997). Em outras palavras, o modelo cognitivo se assemelha a um sistema especialista que é composto por uma infinidade de soluções para os problemas apresentados aos alunos. Conforme Corbett et al. (2010), o modelo cognitivo é usado para rastrear a solução de cada aluno através de problemas complexos, permitindo ao tutor fornecer *feedback* e dicas, passo a passo, e manter um modelo do conhecimento do aluno com base em seu desempenho. Vale ressaltar que o modelo cognitivo pode possuir as resoluções de problemas pré-determinados, nativamente em sua base de dados, ou pode resolver os problemas que o aluno iniciar, permitindo que ele e o professor possam propor novos problemas.

2.3.1 Arquitetura Cognitiva

As arquiteturas cognitivas são modelos de representação que surgiram no início na década de 1950 com o objetivo de criar programas que pudessem raciocinar sobre problemas em diferentes domínios, obter *insights*, adaptar-se a situações desconhecidas e refletir sobre si mesmos. Da mesma forma, o macro-objetivo das pesquisas em arquiteturas cognitivas é modelar a mente humana, eventualmente nos permitindo construir inteligência artificial no nível humano.

Segundo Russell e Norvig (2010), a IA pode ser modelada de quatro maneiras diferentes: sistemas que pensam como humanos, sistemas que pensam racionalmente, sistemas que agem como humanos e sistemas que agem racionalmente. As arquiteturas cognitivas existentes exploraram todas as quatro possibilidades. Por exemplo, nos modelos que buscam mimetizar o pensamento humano, os erros cometidos por um sistema inteligente devem coincidir com os erros normalmente feitos por seres humanos em situações semelhantes. Isso contrasta com os sistemas de pensamento racional que são necessários para produzir conclusões consistentes e corretas para uma determinada tarefa. Uma distinção semelhante é feita para máquinas que agem como seres humanos ou agem racionalmente. Além disso, os autores ressaltam que as máquinas, de qualquer um desses grupos, não devem pensar como seres humanos, apenas suas ações ou comportamentos devem ser semelhantes. Em outras palavras, busca-se apenas simular o comportamento humano e não reproduzir a forma de pensamento humana.

As arquiteturas cognitivas possuem um papel importante nessa proposta, uma vez que servem de subsídio para planejamento do sistema cognitivo a ser proposto. Desse modo, nessa seção serão apresentados os conceitos que tangem as arquiteturas cognitivas, bem como alguns trabalhos que buscam implementar determinada característica. Nessa seção serão mostradas a maior parte dos conceitos existentes por trás das arquiteturas cognitivas, contudo, vale ressaltar que nem todos eles serão implementados nesta proposta.

2.3.1.1 Percepção

A percepção é o processo que converte uma entrada bruta (texto, estímulo de sensores, entre outros) para um formato interno, de modo que se possa raciocinar sobre o que foi percebido. Os cinco mais comuns aos seres humanos (e mimetizados nas arquiteturas cognitivas) são: visão, audição, olfato, tato e paladar. Outros sentidos humanos incluem: *propriocepção* (cinestesia), *termocepção* (percepção de alterações de térmicas no ambiente), *nociocepção* (percepção de estímulos aversivos), senso de tempo, entre outros. As arquiteturas cognitivas implementam alguns desses mecanismos de percepção, bem como outras modalidades que não possuem correlação entre os sentidos humanos, como dados simbólicos (ou seja, entrada via teclado ou interface gráfica do usuário) e vários sensores, como: LiDAR[9], laser, bússola, sonares, etc. Dependendo de suas capacidades cognitivas, um sistema inteligente pode processar várias quantidades e tipos de dados como entrada perceptiva.

Vale ressaltar que, apesar de apresentar e discutir esses mecanismos em separado, o cérebro humano recebe um fluxo constante de informações de diferentes sentidos e as integra a uma representação coerente e consistente do ambiente. Essa ideia também se aplica às arquiteturas cognitivas. Em geral, nem todos os tipos de mecanismos de percepção estão presentes em um

[9] LiDaR (Light Detection and Ranging) é um sensor a laser que permite estimar a distância de objetos.

único modelo, sendo que muitas arquiteturas implementam apenas duas modalidades diferentes simultaneamente (por exemplo, visão e audição, visão e entrada simbólica ou visão e sensores de alcance). Com poucas exceções, essas arquiteturas são incorporadas e executam essencialmente o que é conhecido como integração de recursos na ciência cognitiva (Zmigrod e Hommel, 2013) ou fusão de dados de sensores em robótica (Khaleghi et al., 2013).

Independentemente de seu design ou finalidade, um sistema inteligente não pode existir isoladamente e requer uma entrada para produzir um comportamento. Nesse sentido, apresenta-se alguns tipos de métodos de entradas de dados (percepções), separando-as em: Visão, Audição, Simbólica e Outros.

2.3.1.1.1. Visão

O processamento visual, segundo Marr (2010), é composto por três etapas: inicial, intermediária e final. A visão inicial é orientada por dados e envolve o processamento da imagem captada e extração de elementos simples, como cor, luminescência, forma, movimento, etc. A etapa intermediária agrupa elementos em regiões, que são posteriormente processadas na etapa final para reconhecer objetos e atribuir significado a eles usando o conhecimento disponível. Além disso, de acordo com Tsotsos (2011), os mecanismos de atenção visual, sistemas de emoção e de recompensa também influenciam todas as etapas do processamento visual, de modo que a percepção e a cognição estão interconectadas em todos os níveis de processamento visual. É importante mencionar que essas etapas de processamento de imagem, mencionadas acima, são implicitamente implementadas em técnicas de *deep learning* (aprendizado profundo). Contudo, de acordo com Kotseruba e Tsotsos (2019), poucas arquiteturas cognitivas implementam tal técnica, são elas: CogPrime (GOERTZEL et al., 2013), LIDA (MADL et al., 2015), SPA (STEWART e ELIASMITH, 2013) e BECCA (ROHRER, 2013). Ainda assim, de acordo com Stokes e Biggs (2014), embora na robótica vários sensores não visuais (por exemplo, sonares, sensores de distância ultrassônicos) e sensores proprioceptivos (por exemplo, giroscópios, bússolas) sejam usados para resolver tarefas visuais, como navegação, prevenção de obstáculos e pesquisa, a entrada visual ainda é responsável por mais da metade todas as modalidades de entrada implementadas.

2.3.1.1.2. Audição

A audição é comumente utilizada para percepção de comandos de som ou voz. Segundo Kotseruba e Tsotsos (2019), a audição, nas arquiteturas, é puramente funcional, sendo que muitas arquiteturas recorrem ao uso de softwares de transcrição de fala em texto, em vez de desenvolver modelos de audição. Entre as poucas arquiteturas que modelam a percepção auditiva estão ART, ACT-R, SPA e EPIC.

A utilização de softwares especialistas em processamento de fala e comunicação ajuda a desenvolver um senso de complexidade e realismo nas arquiteturas, por exemplo, realizando interações em um ambiente lotado (CORTEX – robô vendedor (ROMERO-GARCÉS et al., 2015b)). Uma aplicação prática envolve o uso do reconhecimento de fala para solicitar livros de uma biblioteca pública de Nova York (FORR (EPSTEIN et al., 2012)). Outros sistemas que usam software de processamento de fala pronto para uso incluem o PolyScheme (TRAFTON et al., 2005) e ISAC (PETERS et al., 2001a).

De acordo com Kotseruba e Tsotsos (2019), grande parte do esforço é destinado ao processamento de linguagem natural, ou seja, informações linguísticas e semânticas transportadas pelo discurso, e pouca atenção é dada ao conteúdo emocional (por exemplo, volume, velocidade da fala, e entonação). Algumas tentativas nesse sentido foram feitas no contexto de robótica social. Por exemplo, o robô social Kismet não entende o que está sendo dito, mas pode determinar aprovação, proibição ou suavização com base nos contornos prosódicos do discurso (BREAZEAL e ARYANANDA, 2002). A arquitetura Ymir também possui um analisador de prosódia combinado com um reconhecedor de fala baseado em gramática que pode entender um vocabulário limitado de 100 palavras (THORISSON, 1999). Até o próprio som pode ser usado como uma sugestão, por exemplo, os robôs BBD podem se orientar em direção à fonte de um som alto (SETH et al., 2004).

2.3.1.1.3. Simbólica

A categoria simbólica pode ser determinada por diferentes métodos de entrada que não constituem sensores ou simulações físicas. Nesse contexto temos as percepções em forma de comandos, dados de texto ou eventos desencadeados através de uma interface com o usuário. A entrada em forma de texto pode assumir diferentes formatos. Por exemplo, arquiteturas que executam tarefas de planejamento e inferência lógica tipicamente recebem os comandos de texto escritos em termos de predicados primitivos usados na arquitetura, de modo que nenhuma análise adicional seja necessária (por exemplo, NARS (WANG, 2013), OSCAR (POLLOCK, 1993b), MAX (KUOKKA, 1989), Homer (VERE e BICKMORE, 1990)). Contudo, a entrada pode ser fornecida em qualquer outro formato (por exemplo, matrizes binárias) e podem ser utilizadas em aplicações de categorização e classificação (por exemplo, HTM (LAVIN e AHMAD, 2015), CSE (HENDERSON e JOSHI, 2013), ART (CARPENTER et al., 1991)).

2.3.1.2 Outros

Kotseruba e Tsotsos (2019) e argumentam que outras modalidades sensoriais, como tato, olfato e propriocepção, são implementadas com uma variedade de sensores (físicos e simulados). Muitas plataformas robóticas são equipadas com sensores sensíveis ao toque (similares a pára-choques) para uma parada de emergência, evitando colisões com obstáculos (por exemplo, RCS (ALBUS et al., 2006), AIS (HAYES-ROTH et al., 1993), CERA-CRANIUM (ARRABALES et al., 2009), DIARC (TRIVEDI et al., 2011)). Da mesma forma, a propriocepção, ou seja, a detecção das posições relativas das partes do corpo e do esforço exercido, pode ser simulada ou fornecida por sensores de força (3T (FIRBY et al., 1995), iCub (PATTACINI et al., 2010), MDB (BELLAS et al., 2010), Subsumption (BROOKS, 1990)), *feedback* conjunto (SASE (HUANG e WENG, 2007), RCS (BOSTELMAN et al., 2006)) e acelerômetros (Kismet (BREAZEAL e BROOKS, 2004), 3T (WONG et al. 1995)), entre outros.

2.3.1.3 Atenção

A atenção possui um papel importante na cognição humana, sendo responsável por filtrar as informações relevantes das irrelevantes recebidas nos receptores sensórias do corpo (visão, audição, tato, entre outros). No contexto de arquiteturas cognitivas, a atenção é amplamente

pesquisada no contexto da visão, uma vez que poucos trabalhos se utilizam de módulos auditivos e, mesmo assim, de acordo com Kotseruba e Tsotsos (2019), esses módulos ainda podem ser considerados rudimentares (OMAR (DEUTSCH et al., 1997), iCub (RUESCH et al., 2008), EPIC (KIERAS et al., 2016) e MIDAS (GORE et al., 2009)).

Tsotsos (2011) propõem uma classificação dos mecanismos de atenção separados em três classes:

- Seleção: consiste em selecionar uma informação (um único detalhe ou objeto). Esses mecanismos agregam a capacidade de extrair um objeto de uma imagem, características específicas desse objeto de interesse ou a até mesmo a identificação de um evento ocorrido;
- Restrição: consiste em selecionar algumas informações (um conjunto de objetos). De modo geral, esse mecanismo tem a função de limitar o espaço de busca, filtrando e reduzindo o campo de visão;
- Supressão: consiste em suprimir temporariamente os recursos ao redor do objeto de interesse, reduzindo a incidência de estímulos irrelevantes para a tarefa a ser realizada.

Vale ressaltar que esses mecanismos podem ser combinados para atender as demandas das arquiteturas. Uma vez que o agente desenvolvido nesta proposta não possui mecanismos de atenção baseados em visão, sugere-se a leitura de Tsotsos (2011) para se obter maiores detalhes acerca desses conceitos.

2.3.1.4 Seleção de Ações

A seleção de ações é a estrutura responsável por determinar, a qualquer momento, o que deve ser feito em seguida. Nesse sentido, pode-se dividir esse mecanismo em "o quê", que envolve a tomada de decisão sobre o que deve ser feito, e "como", relacionado a como executar determinada ação (OZTURK, 2009). De forma geral, os mecanismos de seleção de ações são separados em três categorias:

- Planejadas: possuem planos de ações ou objetivos a serem atingidos;
- Reativas: as arquiteturas são projetadas para responderem aos estímulos recebidos ou percebidos no ambiente;
- Dinâmicas: congregam diversos mecanismos (tanto planejados quanto reativos), permitindo maior flexibilidade e capacidade de modelar muitos fenômenos típicos do comportamento humano.

Na literatura, essa distinção nem sempre é explícita, onde a seleção de ações pode estar relacionada a um objeto, realização de uma tarefa ou comandos motores. Por exemplo, no caso da arquitetura MIDAS, o mecanismo de seleção de ações envolve tanto a seleção do objetivo a ser atingido quanto as ações que devem ser realizadas para atingir o objetivo (TYLER et al., 1998). Pode-se entender que essa estrutura atua em diferentes níveis, um estratégico (determinação de um objetivo) e operacional (como atingir o objetivo). Além desse exemplo, outros trabalhos mais recentes como DIARC (BRICK et al., 2007), DSO (NG et al., 2012), MIDCA (PAISNER et al., 2013), COGNET (ZACHARY et al., 2016), entre outros apresentam mecanismos de seleção similares.

2.3.1.5 Critério de Seleção de Ações

Muitos critérios de seleção de ações podem ser considerados quando se planeja uma arquitetura inteligente. Dentre eles, pode-se citar: relevância, utilidade e fatores internos (que incluem motivações, estados afetivos, emoções, impulsos, entre outros).

A Relevância diz respeito ao quão bem a ação responde à situação apresentada. Esse critério se aplica, principalmente, a sistemas que possuem a capacidade de raciocínio simbólico sobre as pré e pós condições da regra antes de aplicá-la. Alguns exemplos que aplicam esse conceito são: MIDAS (GORE et al., 2009), Soar (LAIRD, 2012b), ACT-R (LEBIERE et al., 2013), entre outros.

A Utilidade da ação é uma medida numérica a ser calculada para estimar a contribuição esperada de uma determinada ação em um dado estado do ambiente. Esse critério se aplica quando é possível determinar a priori o valor da ação no ambiente, seja através de uma função de avaliação ou de exploração do ambiente, como no caso de aprendizagem por reforço. Exemplos de arquiteturas que se utilizam dessa mecânica são: RoboCog (BANDERA e BUSTOS, 2013), MACSi (IVALDI et al., 2014), NARS (WANG, 2013), MLECOG (STARZYK e GRAHAM, 2015) e novamente DSO (NG et al., 2012), MIDCA (PAISNER et al., 2013) e Soar (LAIRD, 2012).

No que tange os fatores internos, ressalta-se que eles não determinam diretamente o próximo comportamento, mas servem de influenciadores na seleção de ações. Esses fatores podem ser modelados como fatores de curto, médio e longo prazo, fazendo uma analogia a emoções, estados afetivos, impulsos e traços de personalidade. Isso pode ser utilizado tanto para identificar a afetividade em um indivíduo que interage com a máquina, e assim responder da melhor forma, ou simular afetividade em máquina. As emoções artificiais nas arquiteturas cognitivas são tipicamente modeladas como estados transitórios (associados à raiva, medo, alegria, entre outros) que influenciam as habilidades cognitivas. Alguns exemplos podem ser encontrados em Hudlicka (2004), Goertzel et al. (2008), Wilson et al. (2013); e Bach (2015). Uma grande discussão acerca de fatores internos que afetam a seleção de ações pode ser encontrada no trabalho de Kotseruba e Tsotsos (2019).

2.3.1.6 Memória

A memória é parte fundamental em qualquer modelo cognitivo, independente se o objetivo é resolver um problema (por exemplo, de engenharia) ou simular a mente humana. Desse modo, não são raros os trabalhos que apresentem algum tipo de modelagem de memória. Na literatura que envolve as arquiteturas cognitivas, a memória é descrita em termos de duração (curto e longo prazo) e tipo (processual, declarativo, semântico, entre outros), embora não seja necessariamente implementada como armazenamento de conhecimento. Esse conceito é influenciado pelo modelo de Atkinson-Shiffrin (1968), posteriormente modificado por Baddeley e Hitch (1974). Essa visão da memória é dominante na psicologia, mas sua utilidade para a engenharia é, algumas vezes, questionada pois não fornece uma descrição funcional de vários mecanismos de memória (PERNER e ZEILINGER, 2011). No entanto, a maioria das arquiteturas faz distinção entre vários tipos de memória, embora as convenções de nomenclatura possam diferir. Aqui, as estruturas de memória serão apresentadas como memória sensorial (curto prazo), memória de trabalho e memória de longo prazo.

A memória sensorial tem o objetivo de armazenar, em *cache* (curto prazo), os dados sensoriais recebidos e realizar um pré-processamento antes de transferi-los para outras

estruturas de memória. Por exemplo, essa memória auxilia na resolução de problemas de continuidade, ou seja, identifica instâncias separadas dos objetos como sendo iguais. Um exemplo de implementação e mais detalhes sobre esse tipo de memória pode ser encontrado no sistema ARCADIA, de Bridewell e Bello (2015).

A memória de trabalho é um mecanismo para armazenamento temporário de informações relacionadas à tarefa atual. Essa característica é fundamental para praticamente qualquer capacidade cognitiva, como atenção, raciocínio e aprendizado. As aplicações particulares da memória de trabalho diferem principalmente em quais informações estão sendo armazenadas, como são representadas, acessadas e mantidas. Além disso, algumas arquiteturas cognitivas contribuem para pesquisas contínuas sobre os processos envolvidos na codificação, manipulação e manutenção de informações na memória de trabalho humana e sua relação com outros processos no cérebro humano. Apesar da importância da memória de trabalho para operação, relativamente poucas arquiteturas cognitivas fornecem detalhes sobre sua organização interna e conexões com outros módulos. Em muitos casos, a memória de trabalho é descrita como "estado atual do ambiente" ou "dados dos sensores". Alguns exemplos dessas representações podem ser encontrados nos sistemas MIDCA (COX et al., 2012) e PRODIGY (VELOSO e BLYTHE, 1994).

Por exemplo, as arquiteturas projetadas para o planejamento e a solução de problemas possuem sistemas de armazenamento de memória de curto e longo prazo, mas não costumam utilizar a terminologia da psicologia cognitiva. O conhecimento de longo prazo, por exemplo, é geralmente referido como uma base de conhecimento para fatos e regras de solução de problemas, que correspondem à memória de longo prazo semântica e processual, por exemplo, Disciple (TECUCI et al., 2000), MACSi (IVALDI et al., 2014), PRS (GEORGEFF e LANSKY, 1986), ARDIS (MARTIN et al., 2011), ATLANTIS (GAT, 1992), IMPRINT (BRETT et al,. 2002). Outras representações mais plausíveis (se comparadas com as teorias de funcionamento da memória humana) utilizam um mecanismo de ativação. Alguns dos primeiros modelos de ativação do conteúdo da memória de trabalho foram implementados no modelo ACT-R. Como antes, a memória de trabalho possui o conhecimento mais relevante, que é recuperado da memória de longo prazo, conforme determinado por um viés. Esse viés pode ser entendido como uma função de ativação. Quanto maior a ativação de um dado elemento, maior a probabilidade de ele entrar na memória de trabalho e afetar diretamente o comportamento do sistema (ANDERSON et al., 1996). Alguns exemplos e mais detalhes sobre essas implementações podem ser encontrados nos trabalhos: MAMID (HUDLICKA, 2010), CogPrime (IKLE e GOERTZEL, 2011) e Soar (LAIRD e MOHAN, 2014).

A memória de longo prazo é responsável por preservar uma grande quantidade de informações por um grande período. De forma geral, ela é dividida em memória processual do conhecimento implícito (por exemplo, habilidades motoras e comportamentos de rotina) e memória declarativa, que contém o conhecimento explícito. Esta última ainda pode ser subdividida em memória semântica e episódica. Conforme exposto por Kotseruba e Tsotsos (2019), as dicotomias entre as dimensões explícita/implícita e declarativa/processual das memórias de longo prazo raramente são implementadas diretamente nas arquiteturas cognitivas. Uma das poucas exceções é o CLARION (SUN et al., 1999), onde as memórias processuais e declarativas são separadas e ambas subdivididas em componentes implícitos e explícitos. Essa distinção é preservada no nível de representação do conhecimento: o conhecimento implícito é capturado por estruturas sub-simbólicas distribuídas, como redes neurais, enquanto o conhecimento explícito tem uma representação simbólica transparente (SUN, 2012). Em outras palavras, a memória de longo prazo armazena o conhecimento inato que permite a operação do sistema, portanto, quase todas as arquiteturas implementam memória procedural e/ou semântica. A memória procedural contém o conhecimento sobre como fazer as coisas no

domínio da tarefa. Em sistemas de produção simbólicos, o conhecimento processual é representado por um conjunto de regras "se-então" pré-programadas ou aprendidas para um domínio específico. Alguns exemplos dessas representações podem ser encontrados em 4CAPS (VARMA, 2006), ACT-R (LEBIERE et al., 2013) e Soar (LINDES e LAIRD, 2016).

Por fim, no contexto de memórias, apesar das evidentes diferenças no conceito, algumas arquiteturas não apresentam representações separadas para diferentes tipos de conhecimento ou memória de curto e longo prazo e, em vez disso, usam uma estrutura unificada para armazenar todas as informações no sistema. Nesse sentido, as arquiteturas tratam a memória de forma global. As arquiteturas CORTEX e RoboCog, por exemplo, utilizam um objeto de múltiplos gráficos dinâmico e integrado que pode representar dados sensoriais e símbolos de alto nível que descrevem o estado do robô e o ambiente (ROMERO-GARCÉS et al., 2015; BUSTOS et al., 2013).

2.3.1.7 Aprendizagem

A aprendizagem é a capacidade do sistema melhorar seu desempenho ao longo do tempo. Segundo Kotseruba e Tsotsos (2019), qualquer tipo de aprendizado é baseado na experiência. Por exemplo, um sistema pode ser capaz de inferir fatos e comportamentos a partir dos eventos observados e/ou dos resultados de suas próprias ações. O tipo de aprendizado e sua realização dependem de muitos fatores, como paradigma de design (por exemplo, biológico, psicológico, simulado), área de aplicação, estruturas de dados e algoritmos utilizados para implementar a arquitetura, entre outros. Squire (1992) apresenta uma taxonomia que divide a aquisição declarativa (ou explícita) e não declarativo, que inclui tipos de aprendizado perceptivo, processual, associativo e não-associativo, a qual será adotava para descrever as diferenças entre as arquiteturas.

A aprendizagem perceptiva se aplica às arquiteturas que mudam ativamente a maneira como as informações sensoriais são tratadas ou como os padrões são aprendidos em tempo real (*on the fly*). Esse tipo de aprendizagem é frequentemente realizado para obter conhecimento implícito sobre o ambiente, como mapas espaciais, agrupamento de características visuais ou encontrar associações. Alguns exemplos e mais informações podem ser encontrados nos sistemas MicroPsi (BACH et al., 2007), BECCA (ROHRER, 2013) e Leabra (O'REILLY et al., 2013).

O conhecimento declarativo pode ser entendido como uma coleção de fatos sobre o ambiente e relacionamentos definidos dentro dele. Em muitos sistemas, como o ACT-R, por exemplo, novos conhecimentos declarativos são aprendidos quando um novo dado é adicionado à memória declarativa (por exemplo, quando um objetivo é concluído). Nos sistemas de inspiração biológica, aprender novos conceitos geralmente consiste em formar correspondência entre as características visuais do objeto e seu nome. Exemplos dessas implementações podem ser encontrados em CLARION (SUN et al., 1999), Disciple (BOICU et al., 2005) e Leabra (O'REILLY et al., 2013).

A aprendizagem procedural refere-se aos mecanismos de aprendizagem que ocorrem através da repetição de ações até que a habilidade se torne automática. O aprendizado de um caminho mínimo, por exemplo, é uma tarefa clássica que pode ser realizada através desse método de aprendizagem. Alguns exemplos de arquiteturas que implementam esse conceito e apresentam aplicações, bem como mais detalhes na explicação do mecanismo, são AIS (HAYES-ROTH et al., 1993), RCAST (FAN et al., 2010) e RoboCog (MANSO et al., 2014).

De acordo com Kotseruba e Tsotsos (2019), a aprendizagem associativa é um termo amplo para processos de tomada de decisão influenciados por recompensa e punição. Na psicologia comportamental, ele é estudado dentro de dois principais paradigmas: condicionamento clássico (de Pavlov) e operante (de Skinner). O aprendizado por reforço e suas variantes, como aprendizado por diferença temporal, aprendizado da tabela Q (*Q-Learning*), aprendizado Hebbian, entre outros, são comumente usados em modelos computacionais de aprendizado associativo. Além disso, há evidências substanciais de que a aprendizagem baseada em erros é fundamental para a tomada de decisões e a aquisição de habilidades motoras. Exemplos podem ser analisados utilizando as arquiteturas: CoJACK (RITTER, 2009), ACT -R (CAO et al., 2015) e NARS (SLAM et al. 2015).

Por fim, a aprendizagem não-associativa não requer associações para vincular estímulos e respostas. Habituação e sensibilização são comumente identificadas como dois tipos de aprendizagem não-associativa. A habituação descreve uma redução gradual na intensidade da resposta a estímulos repetidos. O processo oposto ocorre durante a sensibilização, ou seja, quando a exposição repetida a estímulos causa maior resposta. Devido à sua simplicidade, esses tipos de aprendizado são considerados um pré-requisito para outras formas de aprendizado. Por exemplo, a habituação filtra estímulos irrelevantes e ajuda a se concentrar em estímulos importantes (RANKIN et al., 2009), especialmente nas situações em que recompensas positivas ou negativas estão ausentes (WENG e HWANG, 2006). A maior parte do trabalho até o momento nesta área foi dedicada à habituação no contexto da robótica social e da interação humano-computador para obter um comportamento adaptável e realista. Por exemplo, a arquitetura ASMO permite que o robô ignore estímulos irrelevantes (mas salientes) durante a tarefa de rastreamento, quando o robô pode ser facilmente distraído por movimentos rápidos em segundo plano. O aprendizado da habituação (implementado como um valor de impulso associado ao módulo de movimento) permite que ele se concentre nos estímulos relevantes para a tarefa (NOVIANTO et al., 2013).

2.3.1.8 Raciocínio

O raciocínio, originalmente um importante tópico de pesquisa em filosofia e epistemologia, nas últimas décadas também se tornou um dos pontos focais da psicologia e das ciências cognitivas (Kotseruba e Tsotsos, 2019). O raciocínio pode afetar ou estruturar praticamente qualquer forma de atividade humana, uma vez que conta com a capacidade de processar lógica e sistematicamente o conhecimento, Como resultado, além da tríade clássica de inferência lógica (dedução, indução e abdução), outros tipos de raciocínio estão sendo considerados, como heurístico, exequível, analógico, narrativo, moral, entre outros.

De forma geral, a maioria das arquiteturas cognitivas estão preocupadas com raciocínio prático, cujo objetivo final é encontrar a próxima melhor ação e executá-la, em oposição ao raciocínio teórico que visa estabelecer ou avaliar crenças. Há também uma terceira opção, exemplificada pela arquitetura Subsumption, que vê o raciocínio sobre ações como uma etapa desnecessária (BROOKS, 1987) e, em vez disso, busca ação fisicamente fundamentada (BROOKS, 1990). Um dos principais desafios que um ser humano, e consequentemente qualquer inteligência em nível humano, enfrenta regularmente é agir com base em conhecimento insuficiente ou, em outras palavras, tomar decisões racionais em um contexto de ignorância generalizada, conforme salientado por Pollock (2008). Além do conhecimento escasso do domínio, existem limitações em termos de recursos internos disponíveis, como capacidade de processamento de informações. Além disso, restrições externas, como execução em tempo real (*on the fly*), também podem ser exigidas pela tarefa a ser realizada. Alguns

exemplos que abordam o raciocínio através dos conceitos de crença-desejo-intenção (BDI) podem ser encontrados em NARS (WANG, 2013) e OSCAR (POLLOCK, 1993). Contudo, essas arquiteturas visam criar agentes racionais com inteligência no nível humano, mas não necessariamente tentam modelar processos de raciocínio humano.

2.3.1.9 Metacognição

A metacognição, de acordo com Flavell (1979) é definida intuitivamente como "pensar em pensar", é um conjunto de habilidades que monitoram introspectivamente os processos internos e raciocinam sobre eles. Há interesse em desenvolver a metacognição para agentes artificiais, devido ao seu papel essencial na experiência humana e na necessidade prática de identificar, explicar e corrigir decisões errôneas. Desse modo, é possível coletar dados referentes a operações internas e ao status do sistema. Os dados geralmente incluem a disponibilidade de recursos internos e conhecimento atual de tarefas com valores de confiança associados. Além disso, algumas arquiteturas suportam uma representação temporal atual e/ou passado das soluções. Alguns exemplos dessas implementações podem ser encontrados em COGNET (ZACHARY et al., 2000), Soar (LAIRD, 2012a), MIDCA (DANNENHAUER et al., 2014) e CLARION (SUN, 2016).

2.4 Computação Evolutiva

Na Ciência da Computação, a computação evolutiva é uma família de algoritmos para otimização inspirada na evolução biológica, comumente estudada nas áreas de Inteligência Artificial, Pesquisa Operacional e *Soft Computing*. Esses algoritmos consistem em metaheurísticas determinísticas ou estocástica que buscam a resolução de um problema simulando um processo de evolução da solução. Esses conceitos têm sido empregados em uma vasta gama de disciplinas, desde ciências naturais, engenharias, biológicas e da computação (Bäck et al., 2000).

Ainda, de acordo com Bäck et al. (2000), a computação evolutiva surgiu com uma grande vantagem frente a outros métodos pois permite resolver problemas através da simples descrição matemática do que se quer ver presente na solução, não havendo necessidade de se indicar explicitamente os passos até o resultado. Com isso, a computação evolutiva pode ser entendida como um conjunto de técnicas e procedimentos genéricos e adaptáveis, a serem aplicados na solução de problemas complexos, para os quais outras técnicas conhecidas são ineficazes. É importante ressaltar que, atualmente, com a evolução das técnicas e do hardware disponível, muitos problemas que não conseguem ser expressos através de fórmulas matemáticas, podem ser abordados utilizando computação evolutiva. Nesse caso, no lugar de uma função matemática de aptidão, pode-se realizar uma simulação da solução obtida em um ambiente virtual que apresente as principais características do problema real e assim estimar sua qualidade.

Em seu surgimento, a computação evolutiva, juntamente com as metaheurísticas representavam um novo paradigma acerca da solução de problemas, uma vez que ela abre mão da garantia de uma solução ótima para permitir a tratabilidade via uma ferramenta de propósito geral. Em termos históricos, três algoritmos para a computação evolutiva foram desenvolvidos independentes entre si:

- Algoritmos Genéticos: Holland (1975), Bremermann (1962) e Fraser (1957);
- Programação Genética: Fogel (1962);

- Estratégias Evolutivas: Rechenberg (1965) e Schwefel(1965).

A computação evolutiva engloba uma família de algoritmos inspirados na teoria evolutiva de Darwin. Os primeiros livros e teses sobre o tema foram escritos por alguns dos próprios pioneiros da área, onde pode se ter uma ideia acerca da capacidade dos algoritmos evolutivos, apesar das limitações de hardware existentes na época (FOGEL et al., 1966; RECHENBERG, 1973; HOLLAND, 1975; DE JONG, 1975; SCHWEFEL, 1975). Contudo, assim como ocorreu com outros métodos (redes neurais e sistemas *Fuzzy*), somente na década de 90 foi possível confirmar a capacidade dos algoritmos evolutivos na solução de problemas complexos, assim como evidenciar suas limitações. Ainda nessa década, mais especificamente em 1997, surgiu uma referência que tendia a ser fonte de inspiração para outros métodos evolutivos, o livro *Handbook of Evolutionary Computation* (BÄCK et al., 2000).

As motivações por trás do uso de processos evolutivos podem ser apontadas, de modo objetivo, em termos da flexibilidade quanto a solução:

- Enquanto em métodos como o Simplex ou Branch-and-Bound (para resolvermos problemas modelados com programação inteira mista), admite-se apenas soluções ótimas, enquanto que nos métodos de computação evolutiva é possível determinar quão boa espera-se que a solução seja e encerrar o algoritmo quando esse critério for atingido. Isso evidencia algumas vantagens:
    - Menor demanda de hardware;
    - Menor tempo computacional;
    - Qualidade aceitável;
    - Abordagem viável para problemas complexos.
- Permite a aplicação em problemas não-estacionários, onde o problema está sujeito a pequenas interferências do ambiente sem que todo o algoritmo tenha de ser reiniciado;
- Possibilidade de agregar conhecimento ao computador (assim como outros algoritmos de aprendizagem de máquina) sem que seja necessário programar esse conhecimento de forma explícita (por exemplo, através de uma base de regras).

Dentre todos os algoritmos classificados como evolutivos alguns são citados abaixo. Mais informações sobre os métodos podem ser analisadas no trabalho de Bäck et al. (2000):

- Otimização por Colônia de Formigas;
- Programação Evolutiva;
- Algoritmos Genéticos;
- Estratégias Evolutivas;
- Algoritmos Meméticos;
- Otimização por Enxame de Partículas;
- Neuroevolução;
- Inteligência de Enxames.

Neste estudo optou-se por um especialista do domínio de ensino baseado em um algoritmo genético. O objetivo do especialista não é apenas resolver o problema, mas evidenciar características da solução que evoluíram ao longo do tempo. Nesse sentido, os algoritmos genéticos, através da geração populacional, permitem que tais características possam ser

evidenciadas em pós-processamento. Na seção a seguir (seção 2.4.1) serão apresentadas as características do funcionamento dos Algoritmos Genéticos.

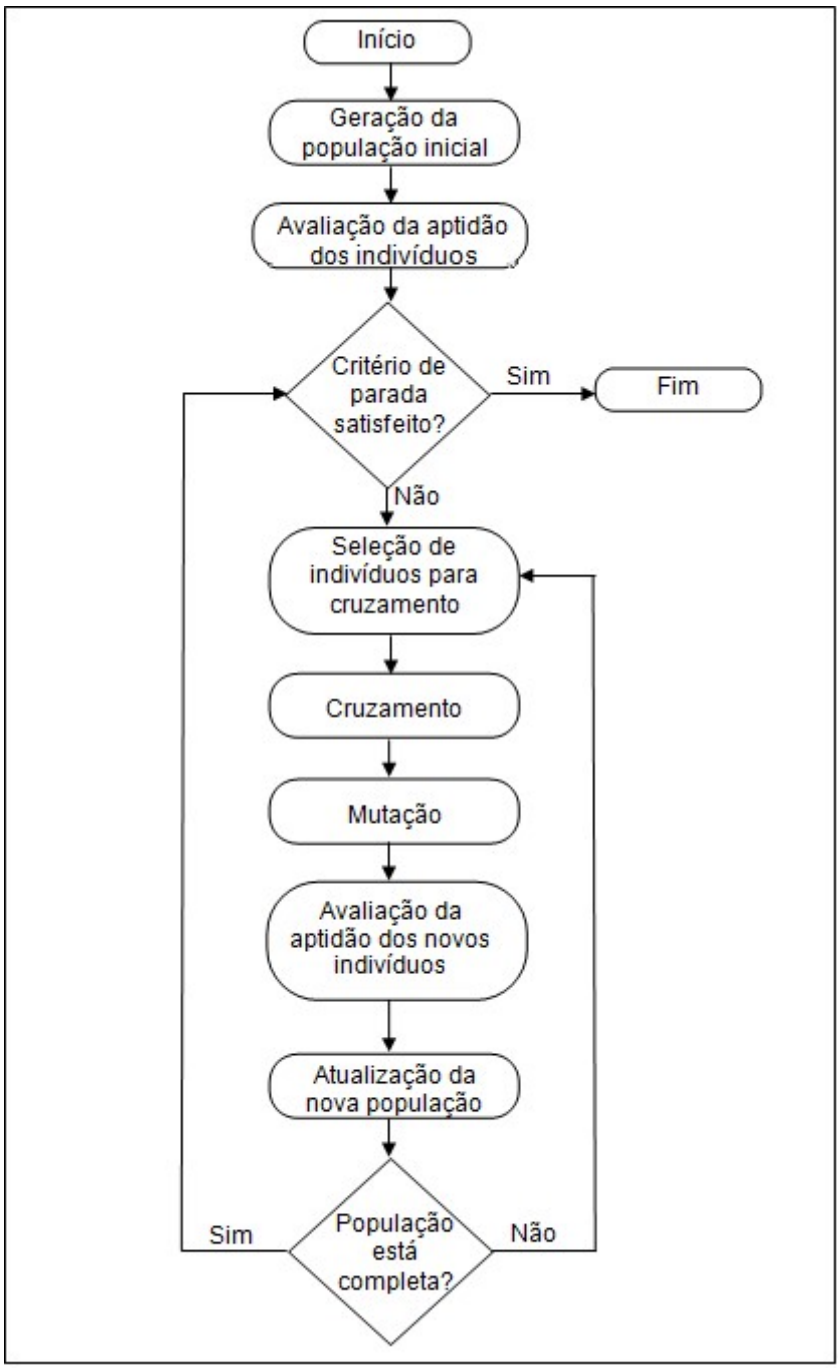
2.4.1 Algoritmos Genéticos

O termo Algoritmo Genético foi inicialmente apresentado por Holland (1975) no livro *Adaptation in Natural and Artificial Systems*. A motivação original do AG foi uma análise biológica. Na reprodução seletiva de animais e plantas, por exemplo, a prole possui certas características desejáveis, dependendo do nível genético proveniente da combinação resultante dos cromossomos de seus pais. No caso do AG, uma população de indivíduos é utilizada, esses indivíduos são referenciados na literatura como cromossomos (REEVES, 2003). A recombinação é realizada fazendo uma analogia com a genética, utilizando operadores de cruzamento e mutação, e a busca é guiada pela avaliação da função objetivo de cada indivíduo da população. Com base nessa sistemática, os indivíduos mais bem avaliados, com melhor *fitness*, podem ser identificados e, assim, terem mais chances de se reproduzir. O funcionamento de um Algoritmo Genético está representado por um fluxograma na Figura 2.4.

O processo do Algoritmo Genético apresentado na Figura 3.3 consiste na geração da população inicial e avaliação da aptidão dos seus indivíduos. O Critério de Parada irá garantir que o algoritmo pare de executar, dada alguma regra. Depois da avaliação da aptidão dos indivíduos da população atual, é realizada a seleção de indivíduos para o cruzamento. Esses indivíduos serão cruzados e mutados, caso atenderem aos critérios de cruzamento e mutação e de uma forma definida *a priori*. O novo indivíduo gerado é avaliado e a nova população que está sendo criada é atualizada. Ainda assim, é necessário garantir que a nova população esteja completa antes de voltar a avaliar o critério de parada do algoritmo.

Um AG irá compor a arquitetura do modelo a ser gerado através desse estudo, sendo responsável por um dos três módulos aplicados à resolução do problema (seção 4.3.4). Com isso, seus componentes serão apresentados a seguir, bem como possíveis apontamentos da forma de utilização no modelo a ser desenvolvido.

Figura 2.4 - Dinâmica do processo do Algoritmo Genético.



2.4.1.1 Codificação

Os genes podem ser representados de diversas formas. Reeves (2003) e Hu e Di Paolo (2007) apresentam diferentes possibilidades de representação. Uma das possíveis representações pode ser binária, onde os indivíduos podem ser compostos de 0s e 1s. Em outros casos, podem ser necessários valores não binários, inteiros ou reais. Nesse caso, pode-se, por exemplo, transformar, *a priori*, os números em binários para após aplicar os operadores. Ainda assim, outra possibilidade de representação consiste na utilização de múltiplos vetores. Por fim,

Reeves (2003) e Hu e Di Paolo (2007) abordam o fato de existirem problemas em que a escolha óbvia de representação seria uma sequência de valores e não um conjunto de 0s e 1s. Esse é o caso de diversos problemas como o Problema do Caixeiro Viajante e suas especializações, assim como uma série de outros problemas, incluindo os problemas que serão citados nesta proposta.

2.4.1.2 População Inicial

O Algoritmo Genético parte da geração da população inicial, onde, normalmente, assume-se ser gerada de modo aleatório. Apesar disso, a geração de uma população aleatória não garante necessariamente uma cobertura uniforme do espaço de busca. Para tal, segundo Reeves (2003), podem ser utilizadas técnicas mais sofisticadas que garantem que as soluções promovam maior variabilidade genética e, por conseguinte, possibilitem melhor exploração do espaço de busca. Além disso, soluções conhecidas de boa qualidade podem ser inseridas na população inicial, possibilitando que o Algoritmo Genético encontre melhores soluções mais rapidamente do que simplesmente utilizando uma geração aleatória.

2.4.1.3 Critério de Parada

O critério de parada é necessário para definir quando o processamento deve encerrar. De acordo com Reeves (2003), inúmeros métodos já foram empregados para garantir que o AG não fique executando indefinidamente, dentre eles, pode-se citar o número total de população geradas, atingir um valor objetivo para a função aptidão e avaliar a diversidade da população. Mais de um critério pode ser empregado ao mesmo tempo de modo a garantir qualidade e/ou tempo de processamento.

2.4.1.4 Métodos de Seleção

Os métodos de seleção definem quais indivíduos serão utilizados no cruzamento. Dentre todos os métodos de seleção, os mais utilizados são a roleta, a classificação e o torneio (REEVES, 2003). No método de seleção por roleta, divide-se a população uma roleta, com valores entre 0 e 1, pelo número total de indivíduos e o tamanho ocupado na roleta é definido pela sua aptidão em relação aos demais. Em seguida, sorteia-se um número aleatório entre 0 e 1 e o indivíduo correspondente é selecionado. Nesse caso, os indivíduos mais aptos possuem maior probabilidade de serem selecionados. Contudo, para esse processo, faz-se necessário que todos os indivíduos sejam avaliados para então serem classificados. Em alguns tipos de problemas, a avaliação completa da população é necessária, tornando esse critério de seleção inviável.

O método de classificação consiste em ordenar os indivíduos de uma população, possibilitando a escolha dos mesmos da forma que for mais conveniente. Para tal, supõe-se que a probabilidade de seleção de um indivíduo classificado na posição k é dada por $P(k)$, conforme equação (2.1).

$$P(k) = \alpha + \beta k \tag{2.1}$$

Na equação (3.10), α e β são pesos. A premissa de que $P(k)$ é uma função distribuição de probabilidade. Nesse caso, não é necessário calcular a função de aptidão dos indivíduos da população, sendo mais adequado a simulações populacionais.

Por fim, o método de seleção por torneio consiste em selecionar e comparar um conjunto τ de indivíduos onde o mais apto é selecionado. Uma vantagem desse método de seleção é que ele necessita avaliar apenas um par ou um conjunto de indivíduos e ainda pode atuar em situações onde não existe uma função objetivo formal. Contudo, é importante destacar que essa técnica não garante que todos os indivíduos serão selecionados para o torneio.

2.4.1.5 Condição de Cruzamento e Mutação

Após a seleção dos indivíduos, deve-se verificar se o critério de cruzamento é atendido, para que então os operadores de cruzamento sejam aplicados. De modo geral, implementa-se uma regra de aleatoriedade (REEVES, 2003). Define-se um valor, também conhecido como taxa de cruzamento, e sorteia-se um número aleatório que irá definir se o cruzamento será ou não realizado. Não obstante, além do operador de cruzamento, pode ser utilizado um operador de mutação, o qual também necessita de um parâmetro para definir a taxa de mutação. Davis (1991) sugerem que, inicialmente, as taxas de cruzamento sejam altas e as taxas de mutação baixas, porém, quando a população começar a convergir, diminuindo a diversidade da população, a taxa de mutação seja aumentada, atualizada em tempo de execução.

2.4.1.6 Cruzamento

A técnica de cruzamento é um dos operadores que permite que as soluções para o problema evoluam. Segundo Reeves (2003), o cruzamento consiste, simplesmente, em substituir uma parte dos genes de um pai, pela parte correspondente de outro pai, através da escolha de 1 ou mais pontos de corte. Por exemplo, para um caso com dois indivíduos a e b, cada um consistindo de 6 variáveis, representando duas possíveis soluções para o problema, Figura 2.5.

Figura 2.5 - Representação dos cromossomos pais.

a_1	a_2	a_3	a_4	a_5	a_6
b_1	b_2	b_3	b_4	b_5	b_6

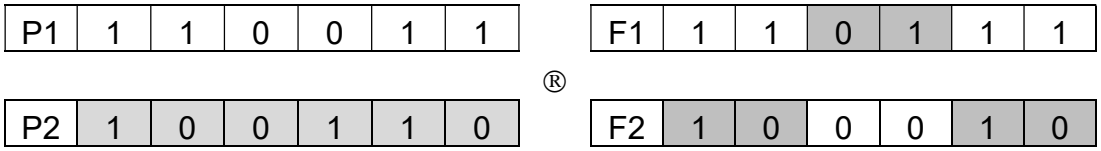
Dois pontos de corte são escolhidos aleatoriamente. No caso de terem sido selecionados, por exemplo, os pontos de corte 2 e 4, a prole gerada (novos indivíduos) ficará como na Figura 2.6.

Figura 2.6 - Indivíduos gerados.

a_1	a_2	b_3	b_4	a_5	a_6
b_1	b_2	a_3	a_4	b_5	b_6

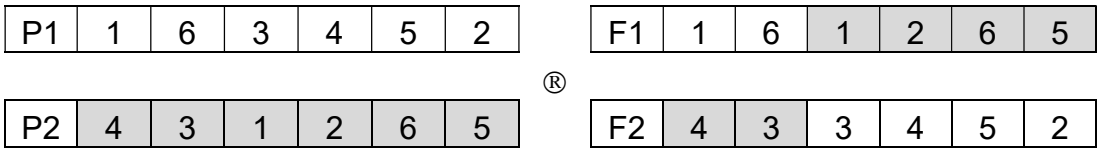
No caso de um cruzamento de cromossomos pais, P1 e P2, compostos por genes binários com os mesmos pontos de corte (2 e 4), pode-se obter dois filhos, F1 e F2, como é mostrado na Figura 2.7.

Figura 2.7 - Cruzamento com um ponto de corte.



Apesar de simples, esse operador de cruzamento pode gerar soluções inviáveis para muitos problemas. Nesse caso, para exemplificar o caso de uma solução inviável, bem como apresentar outra forma de representar um indivíduo (solução), utiliza-se, como exemplo, o Problema do Caixeiro Viajante (DAVIS, 1991), onde um viajante deve visitar todos as cidades uma única vez e retornar para a primeira cidade. Nesse caso, um problema comum a diversos a problemas ocorre na representação de cromossomos não binários, como no caso de o espaço de soluções baseado em um espaço de permutações de $1, \ldots, n$, onde n é o tamanho máximo do cromossomo. A Figura 2.8 representa dois pais (P1 e P2) e dois filhos (F1 e F2, gerados a partir dos pais P1 e P2) gerados a partir de 1 ponto de corte (posição 2).

Figura 2.8 - Cruzamento de cromossomos com representação não binária.



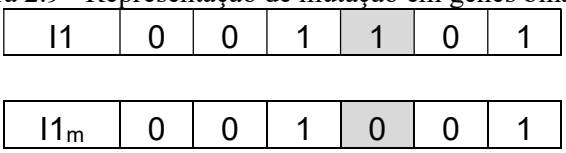
Se no caso da Figura 3.8 com os pais P1 e P2 e um ponto de corte, na segunda posição, fosse a representação do Problema do Caixeiro Viajante, o indivíduo F1 visitaria as cidades 1 e 6 duas vezes e nunca chegaria às cidades 3 e 4. Assim como, o indivíduo F2, visitaria as cidades 3 e 4 duas vezes e nunca visitaria as cidades 1 e 6. Esse tipo de cruzamento tende a gerar uma grande quantidade de soluções inviáveis (DAVIS, 1991). Para lidar com esse problema, inúmeras técnicas foram propostas, dentre as quais cita-se algumas: *partially mapped crossover* (GOLDBERG e LINGLE, 1985), *order based crossover* (DAVIS, 1991), *uniform order based crossover* (DAVIS, 1991), entre outros.

2.4.1.7 Mutação

A mutação pode ser aplicada de diversas formas, desde a seleção de um gene aleatório até mesmo utilizando uma distribuição de Bernoulli associando-se um valor p muito baixo para cada gene, para definir se irá ocorrer ou não uma mutação. Contudo, no caso de utilização de uma distribuição probabilística, pode-se gerar um problema de desempenho quando o cromossomo for muito grande. Devido a isso, em 1964 (BREMERMANN et al., 1964) surgiu uma nova forma da aplicação da mutação. Utilizando uma distribuição de Poisson com parâmetro λ, onde λ é o número médio de mutações por cromossomo e geralmente com valor 1. Com isso, se $\lambda = 1$ e n é o número total de genes de um cromossomo, tem-se que a taxa de mutação é definida por $\mu = 1/n$.

No caso de os genes serem binários, pode ser utilizada uma simples troca de valor, de 0 para 1 ou de 1 para 0, como na Figura 2.9.

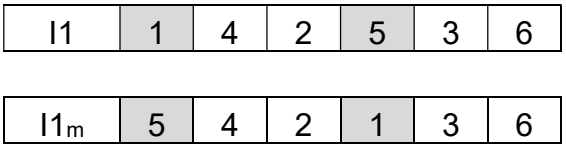
Figura 2.9 - Representação de mutação em genes binários.



A Figura 2.9, mostra o indivíduo I1 sofrendo uma mutação no quarto gene, tendo o valor de seu alelo alterado de 1 para 0, resultando no indivíduo $I1_m$.

Apesar de ser uma troca simples, quando existem diversas possibilidades de valores para os alelos dos genes, pode-se decidir o novo valor de modo aleatório. Ainda assim, em alguns casos, os alelos são sensíveis às variações desses valores, como no caso do Problema do Caixeiro Viajante. Desse modo, Geiger (2008) sugere a utilização de técnica de mutação chamada de *swap mutation* que consiste na troca de valores entre dois genes selecionados aleatoriamente, como na Figura 2.10.

Figura 2.10 - Representação de mutação utilizando *swap mutation*.



A Figura 2.10, acima, mostra um exemplo onde as posições 1 e 4 foram selecionadas no indivíduo I1. Aplicando a técnica de *swap mutation*, alteram-se os valores das posições entre si, gerando um novo indivíduo $I1_m$.

2.4.1.8 Nova População

O AG original de Holland, segundo Reeves (2003), assumia que os passos de seleção, cruzamento e mutação eram aplicados em uma população de M indivíduos até que um conjunto de novos M indivíduos fosse gerado, formando uma nova população. Do ponto de vista da otimização combinatória, isso pode parecer errôneo, pois existe a possibilidade de combinações inapropriadas ocorrerem, descartando boas soluções obtidas até o momento. Por essa razão, De Jong (1975) introduziu o conceito de elitismo e sobreposição de população. A ideia do elitismo consistia em manter vivo o melhor indivíduo da população atual, e gerar os outros $(M-1)$ indivíduos, enquanto sobreposição tinha por objetivo substituir uma fração da população por novos indivíduos.

Geiger (2008) ressalta que é importante que seja mantida a diversidade da população o maior tempo possível e que uma forma muito utilizada se chama política de não duplicidades (DAVIS, 1991). Essa política consiste em não permitir que clones sejam inseridos na nova população. Isso acaba gerando um problema de desempenho, uma vez que os novos indivíduos precisavam ser comparados. Devido a isso, Davis (1991) sugeriu que antes da aplicação do cruzamento, os pais fossem avaliados para identificar possíveis pontos de corte.

2.5 Trabalhos relacionados

Ao longo do detalhamento das estruturas que compõem as arquiteturas cognitivas, inúmeros trabalhos foram citados, correspondendo a modelos de arquiteturas propostas ou a aplicação

dessas arquiteturas em um determinado cenário. Contudo, nenhum desses trabalhos são aplicados na área de educação. Devido a isso, essa seção se destina a apresentação de estudos realizados no contexto de Sistemas Tutores Cognitivos.

Conforme mencionado na seção 2.3, o primeiro tutor cognitivo surgiu na década de 80 para testar o modelo cognitivo proposto por Anderson (1983) intitulado *"The Architecture of Cognition"*. Neste trabalho, o autor apresenta a teoria cognitiva de aprendizagem e solução de problemas ACT (*Adaptive Control of Thought*), a qual seria complementada posteriormente, em 1993, passando a se chamar ACT-R (Anderson, 1993). De acordo com Anderson et al. (1989), o primeiro tutor cognitivo a ser desenvolvido, foi o LISP Tutor, baseado na arquitetura cognitiva ACT-R. A principal motivação para o desenvolvimento do tutor foi abordar questões relativas à natureza da cognição. No centro do tutor havia um modelo cognitivo do conhecimento de um aluno ideal ao realizar os exercícios de codificação. Esse conhecimento é representado na forma de regras de produção (se-então) para geração de código. Quando fornecidas descrições de problemas análogas às fornecidas ao aluno, o modelo gera uma solução passo a passo para os exercícios. O modelo também contém regras de codificação incorretas que geram erros que os alunos provavelmente cometem em vários contextos. O tutor auxilia os alunos essencialmente executando o modelo em sincronia com o aluno, comparando a resposta do aluno em cada etapa com as regras corretas e incorretas relevantes e respondendo de acordo. Analisando a descrição de Anderson et al. (1989), é passando a se chamar ACT-R (Anderson, 1993). De acordo com Anderson et al. (1989), o primeiro tutor cognitivo executa, e *knowledge tracing*, avaliando os conceitos que são esperados que o aluno aprenda.

O LISP Tutor foi testado, primeiramente, em um minicurso, em 1984, para ensino da linguagem LISP onde os alunos assistiam a aulas e realizavam uma série de exercícios de programação determinados *a priori*. Neste primeiro experimento, os alunos que utilizaram o tutor realizaram as tarefas em 30% menos tempo e obtiveram 43% mais sucesso em um pós-teste. Um segundo experimento foi realizado nos mesmos moldes do primeiro (CORBETT e ANDERSON, 1991) e foi evidenciado que os alunos que utilizaram o tutor realizaram os exercícios 64% mais rápido e obtiveram notas 30% maiores em um pós-teste. Apesar dos resultados nos pós-testes terem sido favoráveis aos alunos que utilizaram o tutor, os autores ressaltam que parte disso pode ser em função de que os alunos que realizaram os exercícios na interface padrão, sem o apoio do tutor, não conseguiram finalizar todos os exercícios propostos. Os autores complementam que acreditam que, se esses alunos tivessem mais tempo, também conseguiriam atingir os mesmos resultados nos pós-testes, mas que um tutor bem projetado pode alavancar o desenvolvimento do aluno para um patamar mais elevado em um terço do tempo requerido em ambientes de aprendizagem tradicionais.

De acordo com Anderson et al. (1995), juntamente com o LISP Tutor, surgiu o Geometry Tutor. O tutor foi desenvolvido para realizar provas matemáticas em geometria, para o ensino médio, consiste em um conjunto de regras ideais e confusas, um tutor e uma interface. O conjunto de regras é responsável pelo cálculo eficiente das correspondências com todas as regras corretas e incorretas. A interface é responsável por interagir com o aluno e representar graficamente a prova. O tutor é responsável por direcionar o conjunto de regras e a interface para alcançar uma estratégia de tutoria. A estratégia, por sua vez, consiste em rastrear o comportamento do aluno em termos de quais regras (dentre o conjunto de regras) ele instancia, corrigindo o aluno quando necessário e o ajudando a superar obstáculos.

O tutor teve seu estudo piloto realizado no ano escolar de 1985-1986. No ano escolar seguinte, foi realizado um experimento com turmas de controle, onde foi verificado uma performance melhor nas turmas que utilizaram o tutor. Inúmeras pesquisas foram realizadas por outros pesquisadores, obtendo resultados positivos e negativos (SCHOEFIELD e EVAN-

RHODES, 1989; WERTHEIMER, 1990; KOEDINGER e ANDERSO, 1990; KOEDINGER e ANDERSON, 1993).

Poucos anos depois, Singley et al., (1989), propuseram um tutor de álgebra para problemas que envolvessem a interpretação de um enunciado para elaboração e resolução do problema. O tutor possui um *knowledge tracing* que avalia, através de uma rede bayesiana, a probabilidade de o aluno ter aprendido os conceitos que se referem a resolução de determinado problema. São eles: 1) definir o problema visualmente, utilizando figuras e representações espaciais; 2) identificar e mapear os valores; 3) gerar restrições; 4) combinar as restrições geradas; e 5) calcular o resultado final para o problema. Contudo, esse tutor não dispõe do knowl*edge tracing* utilizado no LISP Tutor.

Os primeiros resultados do tutor cognitivo Algebra I foram obtidos no ano letivo de 1987-1988, onde identificou-se que não houve diferenças entre as turmas experimentais, que tiveram acesso aos tutores, e as classes de controle que não tiveram. Os autores argumentam que o principal motivo da falta de efeito foi que havia uma grande diferença entre a interface do tutor e a interface usada nas aulas (papel e lápis). Além disso, outra razão estaria relacionada ao perfil dos alunos, onde, para alguns, a manipulação de símbolos poderia ser considerada fácil, não havendo influências do tutor, enquanto que para outros, que faltam regularmente, o tutor de álgebra se tornou uma parte muito periférica de sua experiência para ajudar a mudar seu padrão geral de comportamento em relação à escola.

Após sua primeira versão, o tutor passou inúmeras melhorias e adaptações, sendo que, de acordo com Corbett (2002), uma das principais funções do tutor seria "saber o que os alunos sabem" através de três pilares: um modelo cognitivo do aluno, observar o comportamento do aluno ao realizar determinadas tarefas e um método para extrair inferências sobre o conhecimento do aluno a partir do seu comportamento.

Nesse sentido, o tutor conta com um *model tracing* que é empregado para interpretar cada ação do aluno, seguindo o seu caminho para a solução do problema, fornecendo apenas o apoio necessário para que o aluno resolva o problema com sucesso, acompanhando o aluno passo a passo. Assim como os tutores humanos, o feedback do tutor cognitivo é focado no contexto de resolução de problemas do aluno, ou seja, tendo como base seus passos até o momento. Se a ação do aluno estiver correta, ela é simplesmente aceita. Se o aluno cometer um erro, ele é rejeitado e sinalizado, contudo, o tutor não fornece explicações detalhadas dos erros, mas permite que o aluno reflita sobre erros. Por fim o aluno ainda pode solicitar ajuda ao tutor que pode fornecer três níveis de conselhos: 1) aconselha sobre uma meta a ser alcançada; 2) conselhos gerais sobre o cumprimento do objetivo; e 3) conselhos concretos sobre como resolver o objetivo no contexto atual.

O Algebra I também monitora o conhecimento do aluno na resolução de problemas, em um processo chamado de *knowledge tracing*. Cada vez que o aluno empregar uma regra cognitiva na resolução de um problema, calcula-se a probabilidade de o aluno conhecer o determinado conceito (Corbett & Anderson, 1995), sendo que essas estimativas de probabilidade são exibidas para o aluno na interface do tutor. Um dos usos para o *knowledge tracing* é identificar quais conceitos o aluno ainda não domina e sugerir exercícios em que ele tenha de aplicá-los.

Saindo da área de exatas, o Genetics Cognitive Tutor é um Sistema Tutor Cognitivo, também desenvolvido na Carnegie Mellon University por Corbett et al. (2010), e como o seu nome sugere, o objetivo é abordar a resolução de problemas na área de genética. O tutor foi construído em torno de um modelo cognitivo que é usado para acompanhar os passos do aluno durante a resolução de um problema (*model tracing*), possibilitando fornecer *feedback* e dicas conforme necessário, além de manter um modelo do conhecimento do aluno com base em seu desempenho (*knowledge tracing*).

O sistema consiste em 16 módulos, abrangendo cinco tópicos gerais: genética mendeliana, análise de linhagem, recombinação e mapeamento genético, regulação de genes e genética de populações. Há um total de cerca de 125 problemas no currículo, sendo que cada problema requer, em média, 25 passos para ser resolvido. Para avaliar os benefícios do uso do tutor, foram realizadas 36 avaliações (pré e pós-teste), em 12 instituições de ensino diferentes, e foi verificado que os alunos que utilizaram o tutor obtiveram, em média, notas 18% maiores que os demais.

Recentemente, Weitekamp et al. (2020) apresenta, o Cognitive Tutor Authoring Tools, um STI que possui um método de *example-tracing*, substituindo as regras de produção existentes nos demais tutores cognitivos, reduzindo o tempo necessário para incluir novos exercícios no tutor. Os autores citam que um designer instrucional pode criar um único problema aritmético de várias colunas (soma de grandes números) com *example-tracing*, simplesmente e resolvendo um problema de todas as maneiras possíveis para gerar um modelo de comportamento, um modelo direcionado especificando todos os caminhos possíveis da solução. Apesar disso, eles argumentam que o *example-tracing* geralmente não é a melhor ferramenta quando se trata de problemas com comportamento complexo ou espaços de solução altamente variáveis, como em um problema de álgebra complexo.

2.5.1 Ensino de Lógica

Das ferramentas encontradas na literatura que dão apoio ao ensino de Lógica, vale o destaque de alguns STIs: Logic-ITA (Lesta e Yacef, 2002; Yacef 2005), P-Logic Tutor (Lukins et al. 2007) e AProS (Sieg, 2007), além do ambiente Heráclito (Galafassi et al., 2013, 2019a, 2019b).

O Logic-ITA é um assistente de ensino/aprendizagem de lógica proposicional baseado na Web. Seu domínio de aplicação é a construção de provas formais em lógica. O sistema atua como um intermediário entre o professor e os alunos: por um lado, ele fornece aos alunos um ambiente para a prática de provas formais com feedback e, por outro lado, permite aos professores monitorar o progresso e os erros da classe. O sistema é adaptado para dois tipos diferentes de usos: para os alunos, é um STI autônomo, enquanto que para os professores, inclui a funcionalidade para configurar níveis de aprendizagem, ajustar os parâmetros para progredir através destes níveis, monitorar o progresso de classe "e recolher dados. P-Logic Tutor é ensinar aos alunos conceitos fundamentais da lógica proposicional e técnicas de prova de teoremas. O P-Logic Tutor desempenha um duplo papel como um instrumento educativo e um ambiente de pesquisa. Ele apresenta aos alunos os conceitos fundamentais da lógica proposicional e também disponibiliza a prática na resolução de teoremas. O programa também fornece um ambiente no qual é possível acompanhar o aprendizado dos alunos, explorar as questões cognitivas de resolução de problema, e investigar as possibilidades de aprendizado. O projeto AProS iniciou em 2006 buscando abranger os conteúdos de lógica. Contudo, o projeto se expandiu e atualmente conta com diversas ferramentas que se interconectam possibilitando buscar provas, resolver teoremas e acompanhar os alunos no processo de prova. Dentre elas destacam-se o Proof Tutor que faz a ponte entre as ferramentas Proof Lab e Truth Lab de prova. Ele permite que os estudantes que estão com dificuldades de avançar em uma prova recebam sugestões, obtidos dinamicamente a partir de outras provas que já foram geradas.

O Ambiente Heráclito é um STI voltado para o ensino de Lógica, possibilitando que os alunos resolvam exercícios desde tabela-verdade até em prova de argumentos por meio das regras da Dedução Natural. Para tanto disponibiliza o um Caderno Eletrônico de Exercícios de

Lógica – LOGOS que permite criar e editar fórmulas, tabelas-verdade e provas da Lógica Proposicional (Galafassi et al., 2019b).

Além de STIs, existem os provadores automáticos, tais como: Coq e HOL, que fornecem linguagens de especificação baseados em lógicas avançadas (Lógicas de Alta Ordem), capazes de oferecer um sofisticado apoio para a construção de provas formais nestas Lógicas. Já Prover9/Mace4, EProver e SPASS são provadores automáticos de Lógica de Primeira Ordem; além dos editores/verificadores de provas JAPE, Pandora e Isabelle.

2.5.2 Considerações

Ao analisar os tutores apresentados, em especial, os tutores nas áreas de matemática e genética, pode-se verificar que há uma limitação quanto aos exercícios disponíveis. No caso do Genetics Cognitive Tutor, há um conjunto estático de 125 problemas a serem resolvidos pelos alunos e não há uma interface para inclusão de novos exercícios. Nos tutores Geometry Tutor e Algebra I também há um conjunto pré-determinado de exercícios. Contudo, há a possibilidade de um professor incluir um novo exercício na base de dados, desde que ele modele o processo de resolução do problema. Por outro lado, o trabalho de Weitekamp et al. (2020) apresenta uma alternativa para lidar com o problema da dificuldade e demora em criar novos exercícios no STI, apontando que recentemente estão sendo feitos esforços nessa direção (assim como esta proposta de trabalho).

Do ponto de vista de DNPL, dentre os STI apresentados (Logic-ITA, P-Logic Tutor, AproS e Heráclito), todos fazem uso de uma demonstração formal que oferece uma estrutura simbólica apropriada para acompanhar o processo de ensino-aprendizagem de dedução na Lógica. Contudo, apenas o AproS e o Heráclito se utilizam de demonstrações similares às utilizadas pelos professores e alunos e alunos em sala de aula. Ainda no que tange a demonstração, apenas o Heráclito apresenta a possibilidade de continuar a prova do teorema de onde o aluno parou, bem como pode fornecer o próximo passo (com base na prova atual). Nesse contexto, o EvoLogic se assemelha ao Heráclito, por possuir todas essas características. Contudo, a diferença se dá no processo em como a solução é gerada. Enquanto o Heráclito continua a prova do ponto que lhe foi apresentada (exercício novo ou parcialmente resolvido pelo aluno), gerando uma solução possível, dentre as diversas existentes, diferentemente do EvoLogic. Por possuir um AG como especialista, o EvoLogic obtém inúmeras soluções para o mesmo exercício no momento em que o aluno o inicia. Essas diversas soluções podem levar a diferentes linhas de raciocínio possibilitando que o *model tracing* forneça feedback (sugerindo novos passos ao aluno) de acordo com o comportamento que ele vem apresentando ao longo da resolução do exercício.

A necessidade de modelar todo o processo cognitivo (*model tracing*) torna inviável, em muitos casos, a inclusão de novos exercícios na ferramenta, sendo esse o maior limitador dos sistemas tutores cognitivos (Blessing et al., 2009). Alguns trabalhos, como o de Galafassi (2019b) e Weitekamp et al. (2020), bem como esta proposta, buscam abordar uma ou mais dessas características, utilizando-se de técnicas de IA para reduzir a carga de trabalho do professor. Nesse sentido, esse estudo propõe um *model tracing* automatizado (apresentado na seção 3.4.1) que acompanhe os passos do aluno ao longo da resolução de qualquer exercício, dentro do domínio de ensino aplicado.

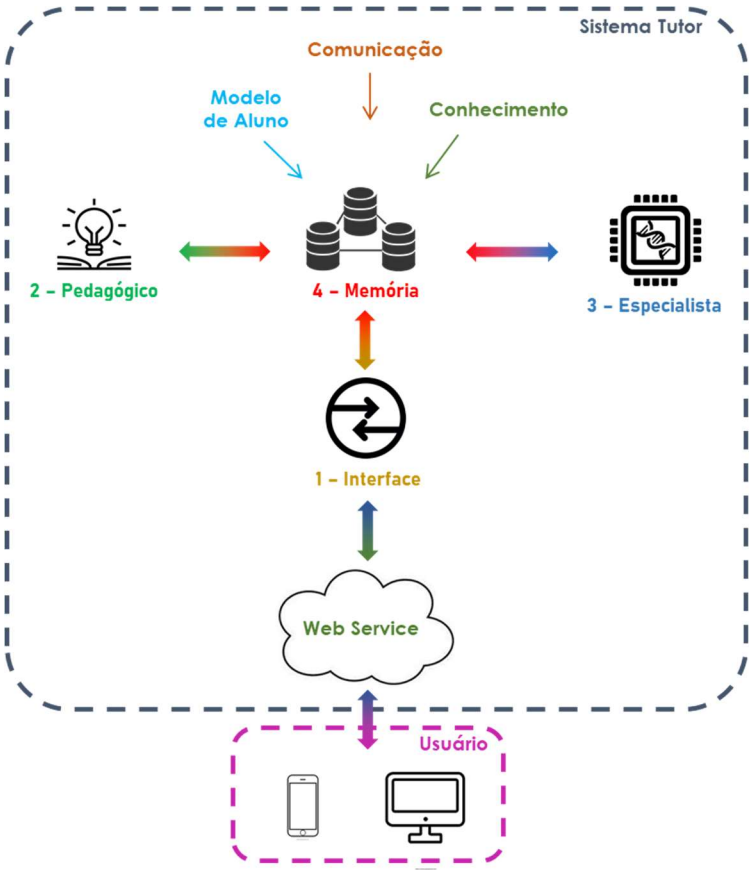
3. EVOLOGIC

O Sistema Tutor Inteligente, aqui proposto, consiste em um sistema multiagente que busca acompanhar o aluno no processo de resolução de um problema. Mais especificamente, o foco desse sistema se dá na resolução de problemas complexos, onde faz-se necessário a realização de múltiplos passos para obter uma solução viável ou recaiam em problemas de análise combinatória. Mais detalhes sobre esses problemas serão apresentados na Seção 3.2.

Inicialmente, o sistema é apresentado como uma arquitetura multiagente composta por 3 agentes: Interface, Pedagógico e Especialista. Um Sistema Multiagente é um sistema composto por diversos agentes onde pode-se decompor o problema em questão em tarefas menores e de menor complexidade, pode-se abstrair o problema, dando foco nos itens relevantes do sistema e permite-se que as interações entre os agentes sejam flexíveis, permitindo que apenas as relações necessárias sejam projetadas e desenvolvidas. Para mais informações, sugere-se o trabalho apresentado por Girardi (2004).

Neste trabalho, cada agente possui uma função específica, sendo detalhadas ao longo desta seção. Além disso, cabe salientar que o modelo pedagógico tradicional em um STI compreende: estratégias de ensino aprendizagem, táticas pedagógicas e modelo do aluno. A comunicação entre os agentes se dá através da troca de simples mensagens, suportadas pelas estruturas de memória, a qual também armazena o Modelo de Aluno e o Conhecimento acerca dos exercícios resolvidos pelo agente Especialista. A organização do ponto de vista de um sistema multiagente pode ser visto na Figura 3.1, onde são destacados os elementos que compõem o sistema e Web Service, utilizado para receber as mensagens provenientes da interação do aluno com seu ambiente de trabalho.

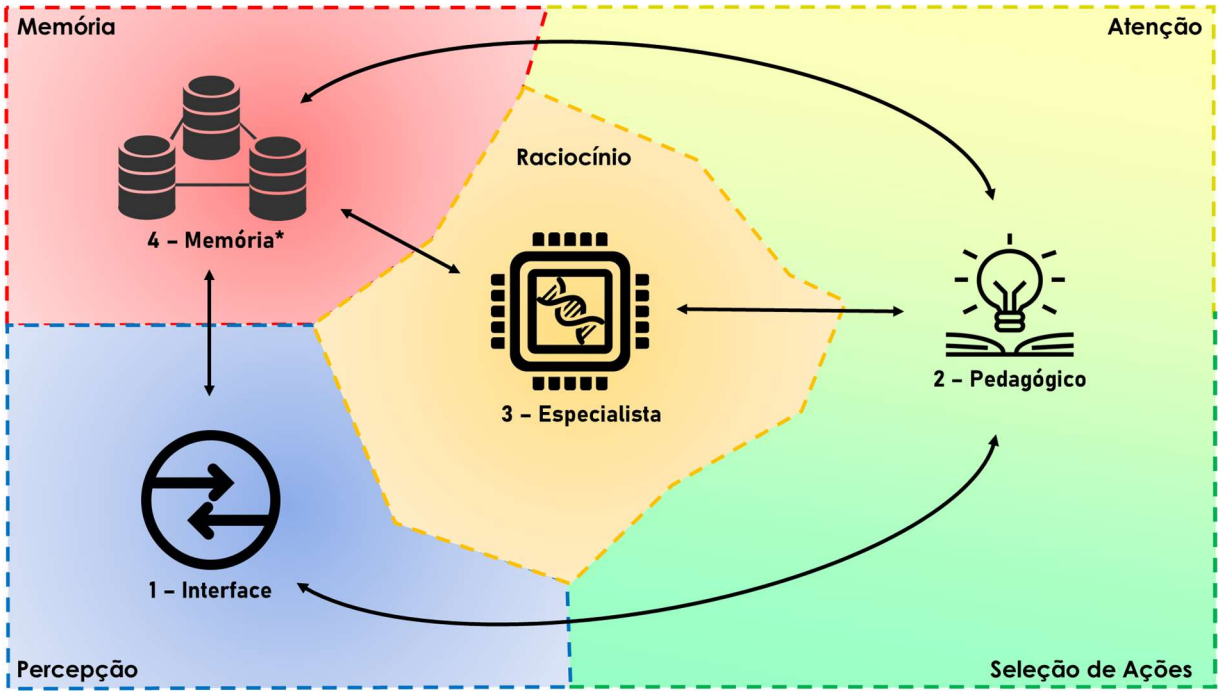
Figura 3.1 - Estrutura multiagente do sistema tutor.



Para suportar a concepção deste STI, utiliza-se uma arquitetura cognitiva que foi inspirada no referencial teórico apresentado na seção 2.3.1, onde incorporou-se apenas as estruturas consideradas mais importantes para o estágio atual do sistema. Dentre as estruturas existentes, optou-se pela implementação de cinco delas: Memória, Percepção, Atenção, Seleção de Ações e Raciocínio (detalhadas na seção 3.2, 3.3, 3.4 e 3.5, respectivamente). Além dessas, outras duas estruturas que são apresentadas no referencial teórico, Aprendizagem e Metacognição, não serão contempladas no escopo deste estudo. A estrutura de Aprendizagem refere-se a aprendizagem do sistema em relação às suas ações, enquanto a Metacognição corresponde ao processo de raciocínio sobre si mesmo, os quais não se adequam ao estágio atual de desenvolvimento deste sistema. Além disso, como propõem o objetivo desse estudo, o tutor conta um model tracing inteligente, responsável por acompanhar as ações do aluno ao longo da resolução de um exercício, fornecendo *feedbacks* quando necessário.

A organização do EvoLogic é apresentada na Figura 3.2, onde detalha-se as estruturas cognitivas mencionadas anteriormente, bem como os 3 agentes que irão compor o STI, as estruturas de memória e as interrelações existentes entre cada componente do sistema.

Figura 3.2 - Arquitetura do sistema.

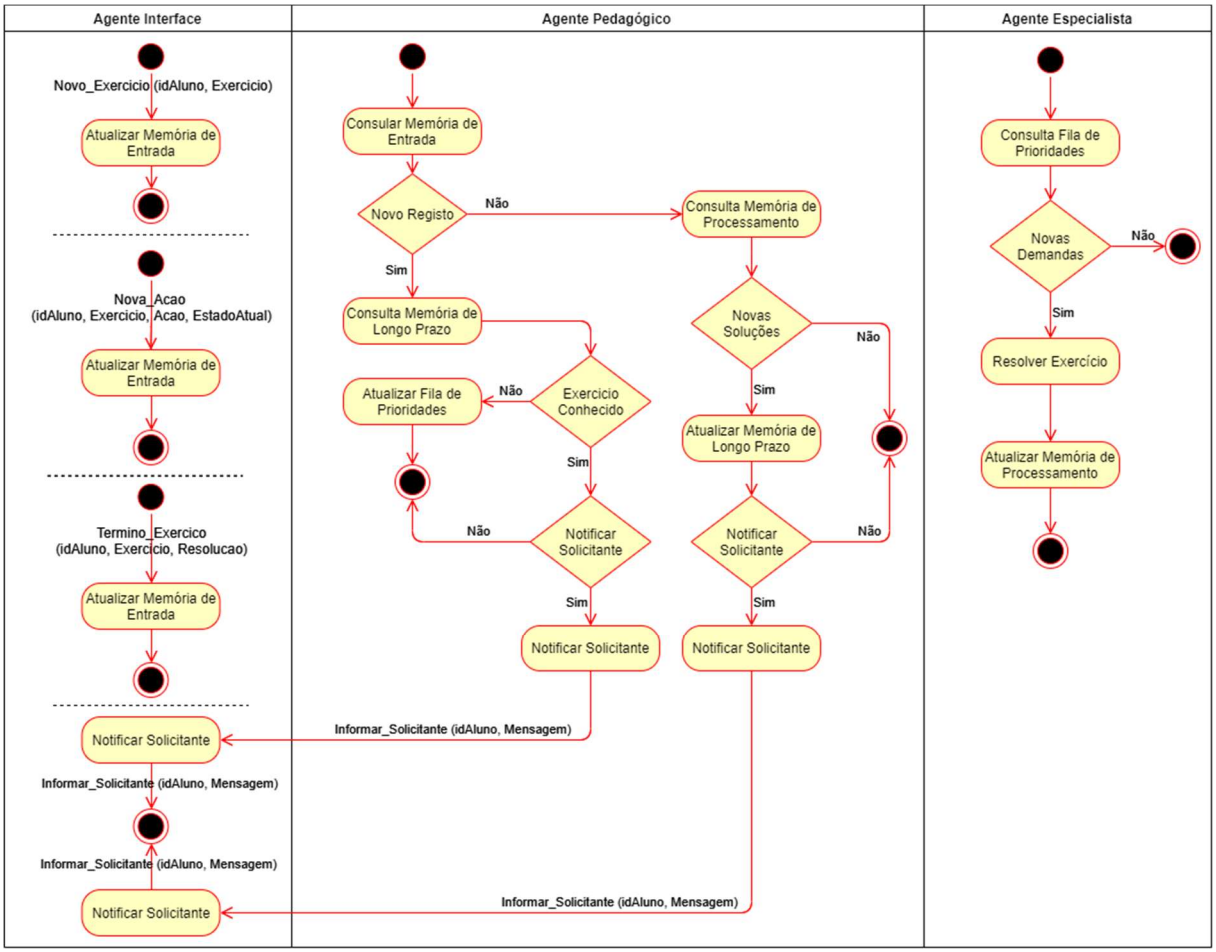


Analisando a Figura 3.2, pode-se verificar a separação entre as 5 estruturas cognitivas: Percepção, Atenção, Seleção de Ações, Raciocínio e Memória. A Percepção é implementada através de um agente Interface (item 1), que se comunicará de forma simbólica, através da troca de mensagens pré-determinadas. Além disso, esse agente é responsável por decodificar as mensagens, armazenando as informações em uma memória de curto prazo, enquanto informa o agente responsável pelo gerenciamento dos processos acerca da nova mensagem. O agente Pedagógico (item 2) caracteriza as estruturas de Atenção e Seleção de Ações, sendo responsáveis por gerir os processos internos dos demais agentes e determinar onde deve-se direcionar o foco do trabalho. Nesse contexto, o agente Pedagógico é acionado para classificar a nova mensagem recebida. Ele avalia se a mensagem necessita de uma resposta e se ela é conhecida. Em caso afirmativo, o agente Interface pode ser acionado para informar algo ao aluno. Contudo, quando não há informações de como responder a uma determinada situação, o agente Especialista (item 3) é acionado. O Especialista consiste em um mecanismo evolutivo que utiliza as informações conhecidas para buscar respostas para esse problema ou para

acompanhar o processo de resolução por parte do aluno. Sempre que novas informações são obtidas, o agente Especialista armazena as informações em memória para que o agente Pedagógico possa verificar se a resposta é satisfatória. Por fim, a Memória (item 4) consiste em um componente, e não em um agente, por isso de sua marcação *, e é subdividida em três partes: Memória de Entrada, Memória de Processamento (memórias de curto, médio prazo, respectivamente) e Memória de Longo Prazo.

Para entender melhor o processo interno de funcionamento do sistema, a Figura 3.3 apresenta o diagrama de atividade, seguindo o padrão UML (*Unified Modeling Language*) (RUMBAUGH et al., 1998), onde são detalhados os processos decisórios, as interações entre os agentes, sua autonomia e as mensagens trocadas.

Figura 3.3 - Diagrama de atividade do sistema tutor.



Na Figura 3.3 pode-se verificar que os 3 agentes (categorizados na parte superior da figura) possuem atividades que são independentes dos demais. Ao ser inicializado, o agente Interface fica aguardando uma mensagem do aluno, para então decodificá-la e inseri-la na memória de entrada.

O agente Pedagógico é responsável por responder uma situação originada pelo aluno, resultante da inclusão de uma nova demanda na Memória de Entrada por parte do agente Interface. Essa demanda por ser de três tipos:

- Novo Exercício: o aluno iniciou a resolução de um novo exercício. O agente Pedagógico irá avaliar se há alguma informação acerca desse exercício e, caso não haja, irá demandar que o agente Especialista o resolva, com o objetivo de adiantar o processo de geração de soluções.

- Nova Ação: o realiza alguma ação para tentar resolver o problema. O agente Pedagógico avaliar a qualidade da ação do aluno e decide se ele deve ser notificado. Caso não haja nenhuma informação acerca do passo realizado pelo aluno, o agente Especialista é notificado para avaliar a situação atual da resolução. A prioridade de inclusão na fila de processamento é alta;
- Término de um Exercício: o aluno concluiu com sucesso um exercício. O agente Pedagógico armazena a solução obtida pelo aluno, servindo como subsídio para atualizar a Memória de Longo Prazo.

O agente Especialista, por sua vez, acompanha a Fila de Prioridades e tem a função de resolver os exercícios solicitados pelo agente Pedagógico. Durante o processo evolutivo, diversas soluções são geradas, sendo que as melhores soluções são armazenadas na Memória de Processamento para posterior analise do agente Pedagógico.

Conforme supracitado, algumas mensagens são trocadas entre os agentes, sendo utilizado o padrão FIPA-ACL[10] (FIPA, 2018) para troca de mensagens. A Tabela 3.1 apresenta o formato de cada mensagem, detalhando o emissor, o receptor e o conteúdo da mensagem.

Tabela 3-1 - Lista de mensagens do sistema.

Mensagem	Emissor	Receptor	Conteúdo
Novo_Exercicio	Solicitante	Interface	idAluno, Exercicio
Nova_Acao	Solicitante	Interface	idAluno, Exercicio, Acao, EstadoAtual
Termino_Exercicio	Solicitante	Interface	idAluno, Exercicio, Resolucao
Informar_Solicitante	Pedagógico	Solicitante	idAluno, Mensagem
Informar_Solicitante	Interface	Solicitante	idAluno, Mensagem

Pode-se verificar, na Tabela 3.1, que todas as mensagens possuem a informação referente ao aluno que realizou a solicitação. Essa informação é imprescindível para o funcionamento adequado do tutor, uma vez que o acompanhamento é realizado de forma independente para cada aluno. Além disso, características do processo de resolução podem ser buscadas com base nas ações de outros alunos, não ficando exclusivamente dependente do agente Especialista. A seguir, apresenta-se os detalhes do sistema proposto, iniciando pelo tipo de problema abordado, pelas estruturas de memórias e finalizando com os agentes.

3.1 Problemas Abordados

Conforme mencionado no início deste capítulo, esse sistema cognitivo não é indicado a qualquer modelo de problema. Ressalta-se que a estrutura de raciocínio consiste em Algoritmos Genéticos, o qual se caracteriza por ser robusto para resolver problemas complexos, em especial, problemas de otimização combinatória. Nesse sentido, apesar de ser possível a

[10] As especificações FIPA definem um modelo de referência para uma plataforma de agentes e também um conjunto de serviços fornecidos ao se conceber sistemas multiagentes interoperáveis. A FIPA (*Foundation for Intelligent Physical Agents*) é uma associação internacional composta por companhias e organizações que têm como objetivo compartilhar esforços com o intuito de produzir especificações para tecnologias genéricas de agentes. Foi criada em 1996 sem fins lucrativos para desenvolver uma coleção de normas relativas à tecnologia de agentes de software (FIPA, 2018).

aplicação de AGs em problemas simples, o custo computacional poderia ser grandemente reduzido ao utilizar um algoritmo exato (por exemplo, algoritmo guloso).

Do ponto de vista deste trabalho, o modelo cognitivo é utilizado para se lidar com problemas que envolvam otimização combinatória ou a resolução de um problema através de uma sequência de passos. A seguir, será exemplificado um possível problema onde o sistema pode ser aplicado, na área de Computação (Dedução Natural em Lógica Proposicional). Para fins de exemplificação do funcionamento do sistema cognitivo nas seções futuras, será utilizado, como exemplo, o problema de Dedução Natural em Lógica Proposicional.

O APÊNDICE A apresenta uma definição formal sobre DNPL, apresentado no trabalho de Galafassi 2019. Além disso, o Ambiente Heráclito[11] disponibiliza gratuitamente um livro INTRODUÇÃO À LÓGICA PROPOSICIONAL, escrito por João Carlos Gluz e Mônica Xavier Py (Gluz e Py, 2005).

3.1.1 Dedução Natural em Lógica Proposicional

Dedução natural é uma técnica utilizada para construir demonstrações formais na Lógica Proposicional onde inicia-se com um conjunto de hipóteses e um argumento que se deseja provar. Através do uso de regras de dedução aplicadas às hipóteses, busca-se encontrar um conjunto de passos que resultem no argumento.

Esse tipo de problema possui uma peculiaridade relacionada às ações que podem ser realizadas a fim de provar o teorema. Ao introduzir as hipóteses, há apenas um conjunto limitado de possíveis ações. Sempre que uma ação é aplicada (ou seja, uma regra de dedução é aplicada), um novo conjunto de ações possíveis é gerado. Esse conjunto pode ser mais abrangente ou mais restritivo que o anterior, resultando, em alguns casos, na impossibilidade de provar o teorema.

Durante o processo de prova de um teorema, o aluno pode seguir por diferentes caminhos, dependendo de sua linha de raciocínio. Uma regra, ao ser aplicada, pode direcionar diretamente a solução do problema (sendo parte do conjunto de regras que compõem a prova do teorema) ou pode ajudar o aluno a identificar qual deveria ser o caminho da prova. Ao identificar que o aluno aplicou uma regra que não faz parte da solução, o STI pode informar ao aluno sobre a situação da aplicação dessa regra ou, ao identificar que mais alunos também realizaram esse mesmo passo, pode questionar o aluno acerca do próximo passo, direcionando o aluno para a conclusão da prova. Vale salientar que o STI busca características acerca da resolução dos problemas através de sua Memória de Longo Prazo, a qual pode ser alimentada pelo agente especialista e por características obtidas através de outros alunos.

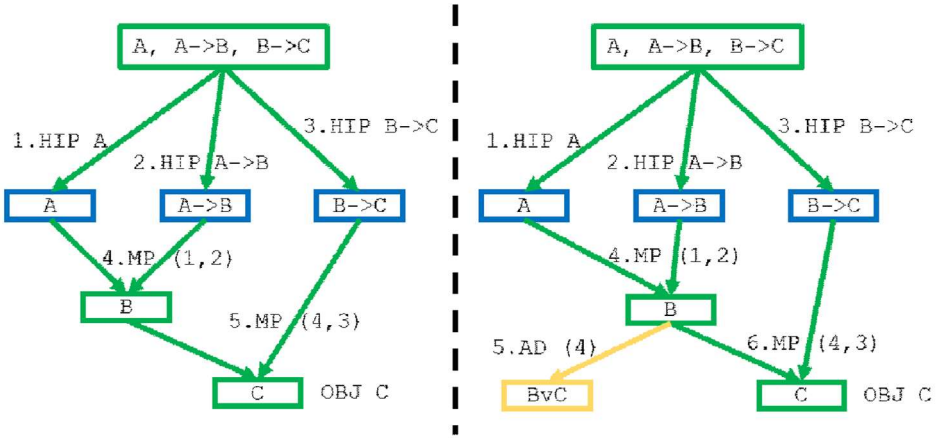
Considerando um exemplo hipotético onde um aluno está a dois passos da prova de um teorema e esse aluno aplica uma regra que não condiz com a solução. Nesse momento, o agente especialista identifica que essa regra poderia ser substituída por outra, mais objetiva. Junto a isso, o agente pedagógico busca em sua base de dados que outros alunos também apresentaram dificuldades em aplicar a próxima regra e, para tal, acabaram aplicando uma regra intermediária. Esse processo permitiu que esses outros alunos visualizassem a necessidade da aplicação da regra correta. De posse dessas informações, o agente pedagógico pode intervir no processo de aprendizagem do aluno questionando sobre regra atual ou sobre a próxima regra, a

[11] Disponível em: http://labsim.unipampa.edu.br:8080/heraclito/index.jsp

qual espera-se que faça com que o aluno perceba qual regra ele deveria ter aplicado desde o início.

Dado a característica dos problemas e DNLP, uma solução para o problema pode ser obtida realizando apenas os passos necessários para provar o teorema ou podem incorporar outros passos. Nesse contexto, a interpretação da solução consiste em identificar se a solução para o problema é correta, em outras palavras, se o teorema pode ser provado, bem como avaliar os passos que foram realizados para obtenção da solução. Considerando o problema dado por: A, A -> B, B -> C :-- C, onde as hipóteses do teorema são A, A -> B e B -> C e deseja-se provar C. A Figura 3.4 mostra duas soluções corretas para um exercício simples, identificando os passos que foram realizados.

Figura 3.4 - Exemplo de duas soluções para um exercício de DNLP.



Ao lado esquerdo, vemos uma solução composta com 5 passos, todos levando diretamente ao objetivo C. Em contrapartida, do lado direito, também temos uma solução correta, atingindo o objetivo C, contudo pode ser visto que foi realizado um passo extra (5.AD (4)). Vale ressaltar que esse passo não inviabiliza a solução, uma vez que a regra de Adição pode ser aplicada durante o processo. Entretanto, esses passos extras, no contexto deste trabalho, podem indicar que o aluno está tendo alguma dificuldade e pode ser conveniente uma intervenção por parte do tutor. Contudo, em alguns exercícios, o aluno pode optar por utilizar apenas regras básicas, evitando o uso de regras derivadas. Nesses casos, é importante que o tutor identifique claramente a diferença entre as linhas de raciocínio e os passos extras.

3.2 Memória – Estruturas de Memória

A memória do sistema é dividida em três estruturas diferentes entre si. A primeira consiste em uma memória de curto prazo, chamada de Memória de Entrada, representada através de uma fila, armazenando as mensagens recebidas, seguindo a regra FIFO (*First In, First Out,* ou seja, a primeira mensagem que chegou será a primeira a ser analisada e, por conseguinte, removida da fila). Essa memória é considerada de curto prazo pois apenas irá armazenar as mensagens de entrada até que o agente Pedagógico possa processá-la. Uma representação gráfica dessa memória pode ser vista na Figura 3.5.

Figura 3.5 - Representação gráfica da Memória de Entrada.

A segunda memória é chamada de Memória de Processamento. Considera-se que essa memória seja de médio prazo, uma vez que ela armazena os principais indivíduos de cada população gerada pelo agente Especialista. Essa memória consiste em uma estrutura simples, onde armazena-se a origem da demanda repassada ao Especialista (ou seja, a demanda identificada pelo agente Pedagógico) e os principais indivíduos de cada população. Esses indivíduos serão utilizados pelo agente Pedagógico, de modo a obter respostas às situações que surgiram com as mensagens de entrada. Uma representação gráfica dessa estrutura de memória é mostrada na Figura 3.6.

Figura 3.6 - Representação gráfica da Memória de Processamento.

Teorema #1

População Inicial

Indivíduo 1					
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B
Dependência	-	-	-	-	1, 2
Indivíduo 2					
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	CJ A ^ (A -> B)
Dependência	-	-	-	-	1, 2
Indivíduo 3					
...					

População 100

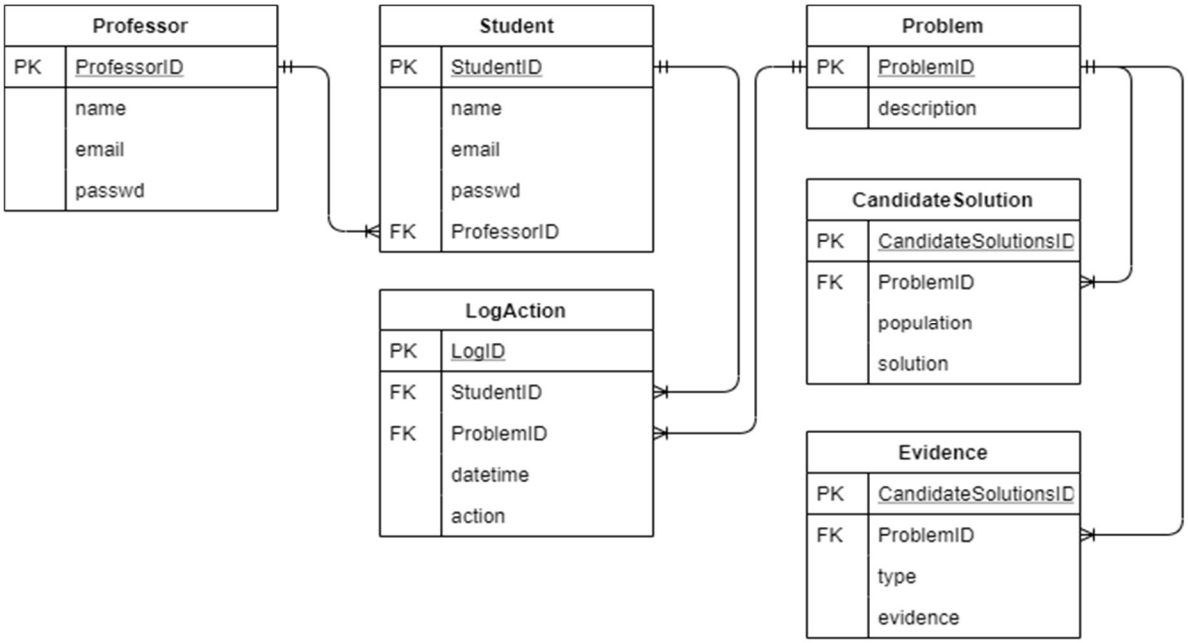
Indivíduo 1						
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B	CJ A ^ (A -> B)
Dependência	-	-	-	-	1, 2	1, 2
Indivíduo 2						
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B	MP C
Dependência	-	-	-	-	1, 2	3, 4
Indivíduo 3						
...						

População 200

...

Por fim, quando o agente Pedagógico identifica que novas respostas foram geradas para as demandas apresentadas, o processo de identificação de soluções analisa a Memória de Processamento e armazena as novas respostas na Memória de Longo Prazo do sistema. Essa memória consiste em uma estrutura de entidade-relacionamento, que armazena todas as mensagens recebidas, bem como as situações problema, suas respostas, e todos os exercícios que foram iniciados (mesmo que não tenham sido resolvidos). Na Memória de Longo Prazo, Figura 3.7, armazena-se todo o histórico do aluno (modelo do aluno), além das soluções obtidas pelo agente Especialista.

Figura 3.7 - Modelo de Entidade-Relacionamento da Memória de Longo Prazo.



Na Figura 3.7 é possível identificar que, além das soluções candidatas geradas pelo agente Especialista, um registro de ações do aluno também será armazenado. Esse registro é essencial para que informações mais detalhadas, como quais passos o aluno voltou atrás, em caso de ter realizado mais de uma vez o algoritmo, quais as diferenças podem ser evidenciadas, entre outras situações.

3.3 Percepção – Agente Interface

O mecanismo de percepção se baseia na captura de informações do ambiente e, conforme visto na seção 2.3.1.1, pode consistir em diversos mecanismos. Neste trabalho, o sistema percebe o ambiente através de um mecanismo simbólico. Em outras palavras, o sistema não possuirá estruturas visuais ou auditivas, sendo que toda informação será enviada a ele através de mensagens predefinidas. O recebimento dessas informações se dá através de um agente chamado de Interface, sendo ele responsável pela comunicação do sistema cognitivo com o aluno.

O agente Interface tem a função de receber as mensagens, decodificá-las e armazenar as informações na memória de entrada. O processo de decodificação consiste em transformar a mensagem textual recebida em uma estrutura que se adeque ao formato da memória. Esse processo é importante para simplificar o processo interno de interpretação, processamento de demandas e análise dos resultados.

Além do recebimento de mensagens, o agente também é responsável por comunicar qualquer informação relevante ao aluno. Conforme mencionado no início deste capítulo, o agente Pedagógico pode julgar necessário notificar o aluno acerca de alguma ocorrência ou de alguma informação relevante. Essa troca de mensagens se dá através de estruturas de mensagens predefinidas conforme mostrado anteriormente, na Tabela 3.1.

3.4 Atenção e Seleção de Ações – Agente Pedagógico

Os mecanismos de Atenção e de Seleção de Ações serão mostrados de forma conjunta, uma vez que ambos são caracterizados pelo agente Pedagógico. Nos termos desta proposta, a Atenção é entendida como um processo interno, referindo-se à atenção dada às tarefas cognitivas do sistema e consiste em um processo similar ao descrito na seção 2.3.1.2. De forma objetiva, a Atenção determina qual deve ser a prioridade de processamento a ser realizada pelo agente Especialista, uma vez que algoritmos evolutivos podem demandar grandes recursos computacionais.

A Seleção de Ações, conforme mostrado na seção 2.3.1.3, busca responder a uma situação em que o sistema se encontra. No caso do tutor, o objetivo é responder às ações do aluno, quando necessário.

Com base no fluxo de processos, apresentado no Diagrama de Atividades, Figura 3.3, algumas atividades são essenciais para o funcionamento do sistema cognitivo. A primeira função consiste em analisar o processamento evolutivo, realizado pelo agente Especialista, sobre uma determinada demanda. Esse processo consiste em avaliar as gerações populacionais de um exercício (armazenadas na Memória de Processamento) e identificar as diferenças que ocorreram ao longo do processo evolutivo como, por exemplo, o surgimento ou desaparecimento de uma característica da solução. Essas mudanças são armazenadas na Memória de Longo prazo, servindo como fonte de consulta para avaliar as ações do aluno no ambiente e informá-lo sobre seu progresso.

A segunda função é buscar, na Memória de Longo Prazo, se há informações sobre as ações dos alunos. Essas informações surgem com base na comparação da resolução do mesmo exercício por outros alunos ou através da avaliação do processo evolutivo do agente Especialista. Em caso afirmativo, uma mensagem pode ser elaborada e encaminhada para o ambiente, de modo que o aluno seja informado.

Além dessas características obtidas através do agente Especialista, uma análise do processo de resolução do problema por parte do aluno também é realizada. Essa análise consiste em avaliar os passos realizados, os passos refeitos no mesmo exercício (caso haja uma opção de retroceder a ação) e soluções prévias. Ao resolver o mesmo problema mais de uma vez, o aluno pode identificar que alguns passos foram desnecessários ou uma nova lógica pode emergir. Essas ocorrências podem ser analisadas pelo agente Pedagógico e armazenadas como informação para auxiliar outros alunos quando tentarem esse mesmo problema.

3.4.1 Estratégias de Ensino-Aprendizagem: Aprendizagem Baseada em Exemplos

No contexto desta proposta, a identificação da linha de raciocínio do aluno é importante para que um feedback adequado seja fornecido. Contudo, dadas as características operacionais do agente Especialista, pode-se utilizar os conceitos de Ensino Baseado em Exemplos (EBL – *Example-Based Learning*) (GOG e RUMMEL, 2010; RENKL, 2011l; RENKL, 2014) como suporte pedagógico para suportar as estratégias de mediação.

A aprendizagem baseada em exemplos integra os princípios teóricos de três áreas de pesquisa em educação: raciocínio analógico, aprendizagem observacional e aprendizagem dos exemplos trabalhados. De acordo com Kolodner (1997), a teoria do raciocínio por analogia define que fornecer exemplos relevantes de casos mostrando o processo de solução de problemas permite que os alunos façam inferências análogas úteis quando confrontadas com

novos problemas. Kochen (1983) aponta que um exemplo disso são os médicos que geralmente confiam na experiência anterior em solução de problemas para fazer essas inferências analógicas. Alguns princípios da aprendizagem baseada em exemplos também são baseados na aprendizagem observacional (KOCHEN, 1983). Esse tipo de aprendizado ocorre quando uma pessoa obtém conhecimento observando um exemplo de modelagem no qual outra pessoa (isto é, um modelo) executa uma tarefa (VAN GOG e RUMMEL, 2010). Mais especificamente, quando os educadores precisam lidar com a aquisição de habilidades cognitivas com base em princípios abstratos subjacentes, a modelagem abstrata pode ser usada para permitir que os alunos estudem um exemplo de modelagem no qual o modelo explica seu raciocínio (RENKL, 2014). Na aprendizagem baseada em exemplos trabalhados, os alunos estudam uma solução didática de um dado problema apresentado passo a passo, demonstrando como ele deve ser resolvido (BANDURA, 1977; ATKINSON et al., 2000).

Inicialmente, os alunos são apresentados a conhecimentos fundamentais específicos do domínio de ensino. Nesta etapa os alunos adquirem algum conhecimento básico para entender o problema ou a terminologia desconhecida. De acordo com Große e Renkl (2007), o conhecimento específico do domínio, para os alunos, é ainda mais importante quando eles estão envolvidos em atividades adicionais durante o processo de aprendizagem. Por exemplo, se os alunos necessitam gerar uma explicação para um diagrama gráfico, eles precisam ter um certo nível de conhecimento do domínio para tornar essa explicação significativa.

Em seguida, os alunos são apresentados a uma série de problemas resolvidos, que contêm tanto a declaração do problema quanto as soluções. De acordo com Sern' et al. (2015), esta etapa é o que torna o EBL único, pois apresentar aos alunos um problema resolvido pode dar a eles uma ideia de como um problema deve ser efetivamente resolvido. Além disso, espera-se que os alunos consigam projetar uma estratégia de abordagem aos problemas com base nos exemplos analisados. Segundo Houghton (2004), não há regras para definir o formato das apresentações de problemas elaborados e alguns instrutores praticam uma maneira mais convencional - mostrando o problema elaborado através do quadro branco ou dos livros - enquanto alguns usam tecnologia de computador e elementos multimídia para problema mais apresentável, autêntico e acessível. Outro ponto que vale ressaltar é que a instrução do problema elaborado pode ser combinada com sucesso com outras atividades de aprendizagem, como instruções de autoexplicação, apoios de tutores ou o fornecimento de feedback durante o processo de aprendizagem. Por fim, o aluno pode iniciar a resolução de exercícios que se baseiem nos exemplos estudados.

Esses processos, aliado com a capacidade do EvoLogic em obter diferentes soluções, a priori, para o mesmo exercício, torna possível se inspirar nos conceitos de EBL para apoiar o aluno no seu processo de ensino-aprendizagem. Nesse sentido, com base em Renkl (2014), propõe-se 4 estratégias de ensino-aprendizagem:

- ELB1: aprender com exemplos de um caso particular;
- ELB2: fornecer vários exemplos;
- ELB3: solicitar aos alunos que façam conexões entre exemplos e princípios;
- ELB4: vincular exemplos a princípios subjacentes.

3.4.1.1 Táticas Pedagógicas – Planos de Ação

Tendo como inspiração as estratégias citadas anteriormente, aqui serão detalhados os planos de ação do agente Pedagógico para intervir no processo de ensino-aprendizagem do aluno.

Considerando a primeira estratégia pedagógica (**EBL1 - aprender com exemplos de um caso particular**), pode-se utilizar, como exemplo, o exercício `B<->A, (BvC)->(EvQ), (BvA)->C, ~E, C->Q, B|--Q` onde `B<->A, (BvC)->(EvQ), (BvA)->C, ~E, C->Q` e `B` são as hipóteses e deseja-se provar `Q`. Esse problema, assim como a maioria dos problemas de DNLP, possui inúmeras soluções possíveis, mas ao se considerar apenas as soluções diretas (sem passos extras), duas linhas possíveis podem ser analisadas, mostradas na Tabela 3.2.

Tabela 3-2 - Diferentes soluções para o mesmo exercício.

Solução 1		Solução 2	
0	OBJ Q	0	OBJ Q
1	HIP A <-> B	1	HIP A <-> B
2	HIP (B v C) -> (E v Q)	2	HIP (B v C) -> (E v Q)
3	HIP (B v A) -> C	3	HIP (B v A) -> C
4	HIP ~E	4	HIP ~E
5	HIP C -> Q	5	HIP C -> Q
6	HIP B	6	HIP B
7	ADD B v A (6)	7	ADD B v A (6)
8	MP C (7, 3)	8	MP C (7, 3)
9	**MP Q (8, 5)**	**9**	**SD Q (4, 8)**

Um aluno que está iniciando seu estudo em DNLP, por exemplo, pode conhecer apenas as regras de dedução básicas, provando o teorema apresentando na Solução 1 da Tabela 3.2. Pode-se verificar a realização de 3 passos (7, 8 e 9) aplicando as regras Adição e Modus Ponens. Ao resolver o problema, o EvoLogic identificou que existem duas soluções que seguem linhas de raciocínio e regras diferentes, envolvendo uma regra de dedução derivada (Silogismo Disjuntivo). Nesse momento, após a conclusão da prova do teorema, o STI apresenta uma nova solução ao aluno (Solução 2), de forma que ambas soluções podem ser comparadas e analisadas. Nesse momento, o aluno tem a opção de analisar, individualmente, cada regra e analisar sua teoria. Além disso, pode-se sugerir a realização de um novo exercício onde uma (ou mais) dessas regras podem ser aplicadas para resolver o problema.

Ao analisar o exemplo da Tabela 3.2, pode-se verificar que os passos para provar o teorema são diretos e não permitem uma mudança na ordem de execução, ou seja, o passo 8 da Solução 1 não pode ser realizado antes do passo 7 e assim por diante. Contudo, em muitos exercícios, a ordem dos passos pode ser alterada, atingindo o mesmo objetivo. Um desses exemplos é o exercício `A<->Q, F<->R, A∧R |- F∧Q` onde as hipóteses são `A<->Q, F<->R` e `A∧R` e deseja-se provar `F∧Q`. Esse exercício já foi estudado na literatura por Galafassi (2019a) e, apesar de utilizar apenas regras básicas de dedução (Simplificação, Modus Ponens, Eliminação da Equivalência e Conjunção) possui inúmeras soluções. A Tabela 3.3 apresenta três exemplos dessas possíveis soluções.

Tabela 3-3 - Soluções similares para o mesmo exercício.

	Solução 1		Solução 2		Solução 3
0	OBJ F∧Q	0	OBJ F∧Q	0	OBJ F∧Q
1	HIP A<->Q	1	HIP A<->Q	1	HIP A<->Q
2	HIP F<->R	2	HIP F<->R	2	HIP F<->R
3	HIP A∧R	3	HIP A∧R	3	HIP A∧R
4	-EQ R->F (2)	4	SP R (3)	4	SP R (3)
5	-EQ A->Q (1)	5	-EQ R->F (2)	5	SP A (3)
6	SP A (3)	6	MP F (4, 5)	6	-EQ A->Q (1)
7	SP R (3)	7	SP A (3)	7	MP Q (6, 5)
8	MP Q (6, 5)	8	-EQ A->Q (1)	8	-EQ R->F (2)
9	MP F (7, 4)	9	MP Q (7, 8)	9	MP F (8, 4)
10	**CJ F∧Q (9, 8)**	**10**	**CJ F∧Q (6, 9)**	**10**	**CJ F∧Q(9, 7)**

No exemplo apresentado na Tabela 3.3, pode-se verificar três formas de resolver o problema. A Solução 1 mostra um cenário onde as Hipóteses foram manipuladas a priori para então combinar os resultados e chegar na solução. Pode-se observar que os primeiros passos (4 ao 7) aplicam regras diretamente nas Hipóteses e, somente nos passos seguintes (8 e 9) buscam derivar as proposições para atingir a solução (passo 10). No caso da solução 2, pode-se verificar um comportamento diferente, as regras foram aplicadas em duas Hipóteses (passos 4 e 5) e já manipuladas (passo 6). Em seguida, retorna para a hipóteses (passos 7 e 8) e novamente manipuladas (passo 9), culminando na obtenção da solução. Por fim, a Solução 3 apresenta um misto desses dois cenários. Vale ressaltar que ambas as soluções são corretas e que várias combinações de passos podem gerar diferentes soluções válidas para o problema em questão. Nesse sentido, considerando a segunda estratégia (**EBL2 - fornecer vários exemplos**), o EvoLogic pode atuar mostrando ao aluno algumas das inúmeras possíveis soluções para o mesmo problema. É importante considerar que apesar das regras serem as mesmas, as estratégias de abordagem são diferentes (Solução 1 e 2), podendo apresentar diferentes formas raciocinar sobre os problemas de DNLP.

O modelo cognitivo do EvoLogic, que acompanha os passos e a linha de raciocínio do aluno, permite que a terceira estratégia de ensino-aprendizagem (**ELB3: solicitar aos alunos que façam conexões entre exemplos e princípios**) seja implementada. Durante a realização dos passos, o aluno pode aplicar regras de forma correta ou incorreta. Uma regra aplicada erroneamente ocorre quando o aluno seleciona premissas inválidas para determinada regra, por exemplo, a regra Modus Ponens (demanda `P` e `P->Q`, resultando em `Q`) é considerada errada quando o aluno seleciona as premissas `R` e `P->Q`. Nesse caso, após um determinado número de tentativas de aplicação da mesma regra de forma errônea, o STI questionará o aluno se ele deseja visualizar um exemplo da aplicação da regra em questão. Em caso positivo, o EvoLogic procura em sua Memória de Longo Prazo e apresenta um exemplo ressaltando o uso da regra. Retomando ao exercício `B<->A, (BvC)->(EvQ), (BvA)->C, ~E, C->Q, B|--Q` onde `B<->A, (BvC)->(EvQ), (BvA)->C, ~E, C->Q` e `B` são as hipóteses e deseja-se provar `Q`, pode-se verificar, na Tabela 3.4, uma tentativa de aplicação da regra MP (Passo 8 ao lado esquerdo) e a aplicação da regra MP realçada no passo 4 (ao lado direito), junto de suas premissas em itálico (passos 1 e 3), em outro exercício. Este exemplo é mostrado por completo pois, além de demonstrar o uso da regra, ele pode servir de inspiração para continuação do exercício inicial.

Tabela 3-4 - Exemplo sugerido ao identificar a aplicação errônea da regra Modus Ponens.

Tentativa de Aplicação da Regra MP		Exemplo de Aplicação da Regra MP	
0	OBJ Q	0	OBJ C
1	HIP A<->B	*1*	*HIP A->B*
2	HIP (B v C) -> (E v Q)	2	HIP B->C
3	HIP (B v A) -> C	*3*	*HIP A*
4	HIP ~E	**4**	**MP B (3, 1)**
5	HIP C-> Q	5	MP C (4, 2)
6	HIP B		
7	ADD B v A (6)		
~~8~~	**MP ? (7, 2)**		

Além de aplicações incorretas, é importante identificar se as regras que o aluno está aplicando direcionam para a prova do teorema. Ao identificar que o aluno está realizando diversos passos extras (conforme definido na seção 3.1.1) o EvoLogic pode questionar o aluno se ele está tendo dificuldades e se ele gostaria de analisar um exemplo. Em caso positivo, pode-se apresentar um exemplo onde ressalta-se a regra que envolve um próximo passo possível, esperando direcioná-lo para a conclusão da prova do teorema. Considerando o mesmo exercício `B<->A, (BvC)->(EvQ), (BvA)->C, ~E, C->Q, B|--Q` onde `B<->A, (BvC)->(EvQ), (BvA)->C, ~E, C->Q` e `B` são as hipóteses e deseja-se provar `Q`, pode-se verificar que a primeira hipótese (`B<->A`) não é utilizada em nenhuma solução conhecida. Caso o aluno se atenha a ela e insista na realização de diversos passos que não estão direcionando-o para a solução do problema, um exemplo pode ser apresentado a ele, conforme a Tabela 3.5.

Tabela 3-5 - Exemplo sugerido ao identificar que o aluno apresenta confusão.

Possível cenário de confusão		Exemplo de sugestão de regra	
0	OBJ Q	0	OBJ Q
1	HIP A <-> B	1	HIP PvQ->(Q^R)
2	HIP (B v C) -> (E v Q)	*2*	*HIP P*
3	HIP (B v A) -> C	**3**	**ADD PvQ (2)**
4	HIP ~E	4	MP Q^R (3,1)
5	HIP C -> Q	5	SP Q (5)
6	HIP B		
7	**-EQ A->B (1)**		
8	**-EQ B->A (1)**		

Na Tabela 3.5 pode-se verificar que o aluno realizou dois passos tendo como base a Hipótese `A<->B` (passos 7 e 8). Nesse caso, ao verificar que uma das possíveis continuações envolvem a aplicação da regra Adição, o EvoLogic identificou um exemplo onde essa regra é aplicada e sugere ao aluno (passo 3).

A quarta estratégia (**ELB4: vincular exemplos a princípios subjacentes**) encontra-se em fase de definição. Uma possibilidade seria, ao apresentar exemplos equivalentes, solicitar que o aluno demonstre os princípios subjacentes, como por exemplo, apresentar uma solução que utilize apenas regras básicas e sua contrapartida utilizando regras derivadas. Pode-se solicitar que o aluno destaque graficamente quais regras básicas foram aplicadas que podem ser substituídas por uma regra derivada.

Por fim, os exemplos apresentados nas Tabelas 3.2 a 3.5 foram obtidos automaticamente pelo EvoLogic, através de uma simulação de uso. Contudo, vale salientar que em ambos os processos cabem refinamentos. Por exemplo, ao identificar que existe mais de uma possível solução para o problema, pode não ser conveniente exibir essa solução ao aluno se ela utilizar várias regras complexas. Ao aplicar regras corretamente, pode ser interessante avaliar se essa regra o impedirá de concluir a prova e, em caso positivo, uma mensagem de alerta pode ser apresentada ao aluno. Além disso, um mecanismo robusto de identificação dos exemplos pode ser interessante, evitando que situações muito discrepantes sejam apresentadas como exemplo de aplicação de regra (caso o aluno esteja apresentando dificuldades em continuar a prova).

3.4.2 Model Tracing

Conforme já mencionado na seção 2.5.1, os trabalhos que se consideram tutores cognitivos recaem na mesma problemática referente ao *model tracing*, exigindo que todo o processo cognitivo por trás da resolução do exercício seja modelado *a priori*. Em sistemas onde o *model tracing* não é automatizado, os alunos não possuem autonomia de buscar e realizar outros exercícios com o apoio cognitivo da ferramenta. Nesse sentido, esse estudo propõe um mecanismo automatizado que tem o objetivo de acompanhar as ações do aluno acerca de qualquer exercício que ele deseje realizar (no escopo deste domínio de ensino), permitindo que o aluno tenha mais autonomia no uso do EvoLogic.

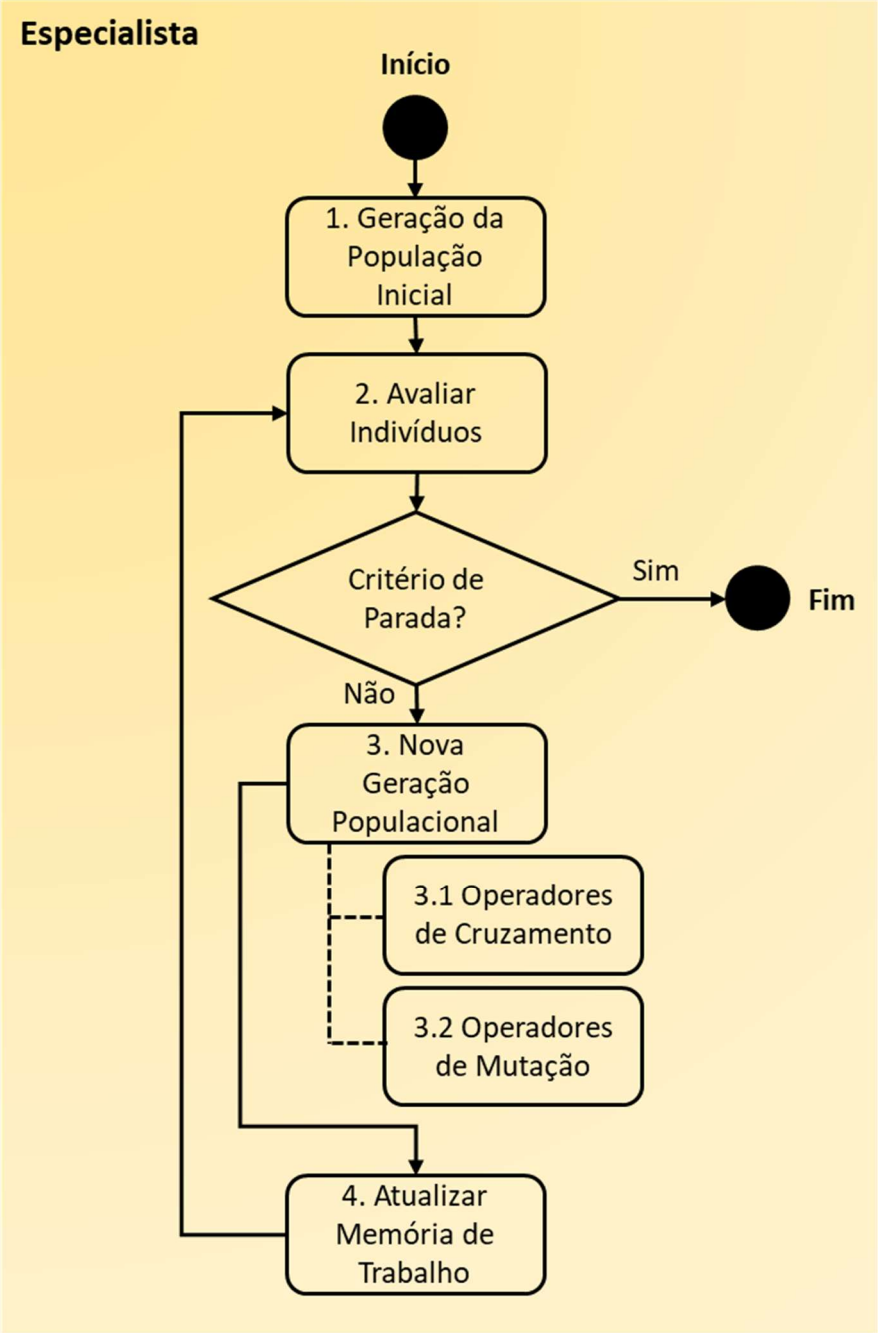
O *model tracing*, nos termos deste estudo, se caracteriza por um conjunto de processos, sendo o agente Pedagógico, o principal responsável. Sempre que um novo exercício é iniciado pelo aluno, o agente Pedagógico consulta a Memória de Longo Prazo a fim de identificar se o exercício que o aluno tenta resolver é conhecido (ou seja, já foi realizado por ele ou por outros alunos) ou se ele é completamente novo. Caso o exercício seja conhecido, o agente Pedagógico já possuirá informações acerca do processo cognitivo e aguardará as ações do aluno. Por outro lado, caso o exercício seja desconhecido, o agente Especialista é acionado para que o problema seja resolvido e as diferentes soluções sejam armazenadas na Memória de Longo Prazo.

O processo de identificação das características das soluções se concentra em duas etapas. A primeira consiste em avaliar as respostas obtidas pelo agente Especialista, ao longo das gerações populacionais, comparando quais elementos da solução foram incluídos e quais deixaram de existir ao longo do tempo. Esse processo permite identificar diferentes formas de resolver o mesmo exercício, bem como identificar simplificações e quando elas ocorreram. A segunda etapa consiste em avaliar as soluções já existentes, assim como avaliar quais passos foram refeitos e quais não fazem parte do caminho mais curto da solução. Quando um exercício é resolvido com passos extras (passos que não compõem a solução mais objetiva), pode indicar que esses passos tenham sido necessários para que o aluno identificasse qual seria o passo correto. O mesmo vale para quando o aluno retroceder um passo, indicando que ele pode ter percebido que esse passo não o levaria diretamente para a solução do exercício.

3.5 Raciocínio – Agente Especialista

O agente Especialista é constituído de um mecanismo evolutivo, baseado em Algoritmos Genéticos, que busca resolver os exercícios iniciados pelos alunos, seja desde o início, ou continuando o exercício a partir de onde o aluno parou. O funcionamento do AG proposto (Figura 3.8) segue a estrutura geral de funcionamento dos AG, apresentada na seção 2.4.1, com uma adaptação para atualizar a memória de trabalho.

Figura 3.8 - Fluxograma do processo evolutivo do agente Especialista.



3.5.1 Representação dos indivíduos

Uma vez que o AG pode ser aplicado a diversos problemas, a representação dos indivíduos necessita atender essa especificidade (REEVES, 2003). Nesse sentido, faz-se necessário especificar as duas principais características dos indivíduos: Genes e Alelos.

O tamanho do indivíduo é determinado pela quantidade de genes, impactando diretamente no tempo de processamento, o que determinará se o sistema será eficiente no acompanhamento do aluno em seu processo de resolução de exercícios (HU e DI PAOLO, 2007). Vale ressaltar que, um indivíduo de tamanho grande, tende a representar e atender a qualquer problema, apesar do tempo de processamento, enquanto um indivíduo de tamanho pequeno resultará em um baixo

tempo de processamento, mas pode não representar adequadamente a solução, tornando-se ineficiente. Sendo assim, opta-se por uma representação de indivíduo de tamanho variável. Essa escolha impacta diretamente nos operadores de Cruzamento e Mutação, que devem se adequar a essa característica.

A segunda característica dos indivíduos são os alelos. De forma objetiva, os alelos correspondem a que tipo de dados (do ponto de vista computacional) que estarão dentro de cada gene. Pode-se utilizar alelos binários, sendo que cada gene poderá ter valores 0 e 1, ou até mesmo estruturas mais complexas, com a utilização de vetores ou matrizes. Para representar esta proposta, opta-se pela utilização de alelos baseados em vetores de duas posições, podendo assumir valores alfanuméricos (números, letras ou símbolos) de tamanho variável. Em outras palavras, cada gene será composto por um vetor de duas posições, sendo que a primeira posição do vetor pode, por exemplo, assumir uma característica da solução enquanto a segunda pode assumir uma quantidade e/ou restrição.

Para ilustrar a representação do indivíduo, utilizaremos um exemplo simples (mencionado na seção 3.1.1) de DNLP dado por: A, A -> B, B -> C :-- C, onde as hipóteses do teorema são A, A -> B e B -> C e deseja-se provar C. A Figura 3.9 mostra a representação gráfica de uma possível solução para o problema.

Figura 3.9 - Representação de um indivíduo.

Índice	0	1	2	3	4	5
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B	MP C
Dependência	-	-	-	-	1, 2	4, 3

Na Figura 3.9 pode-se observar que a solução possui 6 genes, sendo que o primeiro (índice 0) sempre representará o objetivo a ser atingido. Os demais Passos que possuírem o prefixo HIP definem as hipóteses e, dada a característica do problema, sempre devem estar presentes na solução (índices 1, 2 e 3). Os últimos dois genes possuem o prefixo MP que representam regras aplicadas no contexto de DNLP. Nas dependências desses genes pode-se verificar o índice ao qual eles estão vinculados, ou seja, se eles são copiados para formar um novo indivíduo, as dependências também devem ser copiadas (conforme será mostrado na seção 3.5.4).

3.5.2 Critério de Seleção

O critério de seleção determina como os dois pais serão selecionados para gerar indivíduos da nova população (REEVES, 2003). Nesse sentido, foi escolhido o critério de seleção baseado em probabilidade. Ao final da geração populacional, os indivíduos são classificados de acordo com sua função de aptidão, sendo que o indivíduo mais apto possui, proporcionalmente, mais chances de ser escolhido.

3.5.3 Identificação de Possíveis Alelos

Esse processo não é comum aos AG, uma vez que na maioria dos problemas o tamanho da solução é estático e/ou o problema de inviabilidade da solução pode ser tratado através do tipo de dado permitido no alelo ou através do operador de cruzamento. Contudo, em problemas onde é necessário realizar passos diferentes para se obter a solução e, principalmente, esses passos dependem diretamente de seus antecessores, faz-se necessário identificar os possíveis passos antes de realizar os processos de cruzamento e mutação.

Para exemplificar essa ocorrência, pode-se citar o problema de Dedução Natural em Lógica Proposicional, onde o conjunto de hipóteses pode permitir que apenas um conjunto restrito de regras possam ser aplicadas (de forma correta). Contudo, ao longo da prova (ou seja, quando regras vão sendo aplicadas), o conjunto de possíveis regras aumenta, mas ainda assim pode conter restrições. Desse modo, cada indivíduo possui o seu conjunto de possíveis regras a serem aplicadas que precisam ser identificadas a priori.

A Figura 3.10 mostra uma lista de possíveis passos, considerando apenas duas regras: Modus Ponens e Conjunção. Novamente, tomamos como base o exercício: A, A -> B, B -> C :-- C onde temos as três hipóteses A, A -> B e B -> C e deve-se provar C.

Figura 3.10 - Exemplo hipotético de possíveis passos.

Índice	0	1	2	3
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C
Modus Ponens: MP B		(1, 2)		
Conjunção: CJ A^ (A -> B) CJ A^ (B -> C) CJ (A -> B) ^ (B -> C)		(1, 2) (1, 3) (2, 3)		

Observando a Figura 3.10, pode-se verificar que existem 4 passos possíveis, considerando apenas 2 regras e 3 hipóteses. À medida que o número de regras e de passos aumentar, as combinações tendem a acrescer exponencialmente, justificando a necessidade da utilização de um algoritmo de busca robusto.

3.5.4 Operador de Cruzamento

Os operadores de Cruzamento combinam partes da solução de dois pais para gerar um novo indivíduo (REEVES, 2003). Novamente, devido às características dos problemas e da representação da solução (tamanho variável), o operador de cruzamento clássico, onde é determinado 1 ou 2 pontos de corte, conforme mostrado na seção 2.4.1.6, não pode ser aplicado. Desse modo, um novo operador de Cruzamento foi desenvolvido para lidar com esse cenário. A Figura 3.11 mostra o funcionamento do operador de Cruzamento tendo como base o exercício mostrado anteriormente: A, A -> B, B -> C :-- C onde temos as três hipóteses A, A -> B e B -> C e deve-se provar C.

Figura 3.11 - Representação do processo de cruzamento.

Indivíduo Pai 1						
Índice	0	1	2	3	4	
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B	
Dependência	-	-	-	-	1, 2	
Indivíduo Pai 2						
Índice	0	1	2	3	4	
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	CJ A ^ (A -> B)	
Dependência	-	-	-	-	1, 2	
Novo Indivíduo						
Índice	0	1	2	3	4	5
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	CJ A ^ (A -> B)	MP B
Dependência	-	-	-	-	1, 2	1, 2

Na Figura 3.11 temos 2 indivíduos, sendo que eles se diferenciam pelo 5º gene (índice 4) onde o primeiro indivíduo possui o passo MP B (1, 2) enquanto o segundo possui CJ A ^ (A -> B) (1, 2). O novo indivíduo terá, no mínimo, o tamanho suficiente para comportar 1 passo de cada indivíduo pai. No caso do exemplo acima, o indivíduo pai 1 possui tamanho 5, enquanto que o novo indivíduo possui tamanho 6 (para que possa ser incluído um elemento de cada pai). Em seguida, após determinar o tamanho do novo indivíduo, as hipóteses (comum a todas as soluções do problema) são copiadas e seleciona-se, aleatoriamente, um passo de cada indivíduo pai. No exemplo, temos apenas 1 passo (índice 4), o qual é copiado para o novo indivíduo, juntamente com suas dependências. Caso o passo a ser copiado para o novo indivíduo dependesse de outros passos (que não sejam hipóteses) esses passos também são copiados junto com suas respectivas dependências e o tamanho do novo indivíduo é ajustado (caso necessário).

3.5.5 Operador de Mutação

Os operadores de Mutação têm a principal função de manter a variabilidade genética da população (REEVES, 2003). Com base nisso, o operador de Mutação escolhido consiste em substituir os valores de um gene ou incluir um novo gene. O processo de substituição do gene depende de uma série de validações, uma vez que este gene pode ser a dependência de outros. Nesse sentido, uma vez que o critério de Mutação é atendido e o gene a ser substituído é escolhido, todos os genes que são dependentes dele são removidos e a solução é reconstruída aleatoriamente, mantendo-se o mesmo tamanho da solução. No caso da inclusão de um novo gene, a solução é aumentada e um alelo é escolhido aleatoriamente para o novo gene. Vale ressaltar que, em ambos os casos, os alelos são escolhidos em uma lista de valores possíveis, uma vez que é importante manter a viabilidade da solução, conforme mostrado na seção 3.5.3.

A Figura 3.12 mostra a representação do processo de mutação para o problema já mostrado anteriormente (A, A -> B, B -> C :-- C onde temos as três hipóteses A, A -> B e B -> C e deve-se provar C), onde 1 gene é substituído.

Figura 3.12 - Representação do processo de mutação.

Indivíduo Original						
Índice	0	1	2	3	4	5
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B	CJ A ^ (A -> B)
Dependência	-	-	-	-	1, 2	1, 2
Indivíduo Mutado						
Índice	0	1	2	3	4	5
Passos	OBJ C	HIP A	HIP A -> B	HIP B -> C	MP B	MP C
Dependência	-	-	-	-	1, 2	3, 4

Na representação da Figura 3.12, o 6º gene (índice 5) é selecionado. Verifica-se que suas dependências são apenas hipóteses, de modo que apenas seu valor é removido. Escolhe-se aleatoriamente o novo valor para o gene, dentre uma lista de possíveis elementos (seção 3.5.3), e recompõem-se o novo indivíduo.

3.5.6 Critério de Parada

Conforme mencionado na seção 2.4.1.3, existem inúmeros tipos de critérios de parada. Neste trabalho, opta-se pela utilização do número máximo de gerações populacionais como critério de parada para o AG. Além desse, também foi verificado a possibilidade de uso de um critério de parada que considere a variabilidade genética. Em testes empíricos, verificou-se que o AG apresentava dificuldades em convergir, sendo que em alguns casos o processo teve de ser encerrado manualmente. Uma conjectura possível acerca do que estava acarretando esse comportamento remete às características do problema, onde pode haver diversas soluções ótimas (ótimos locais e ótimos globais) para o mesmo problema.

3.5.7 Função de Aptidão

A função de Aptidão determina a qualidade de uma solução através de um valor numérico calculado através de algum critério. Inúmeras possibilidades podem ser desenvolvidas, considerando, por exemplo, o tamanho da solução, se o indivíduo soluciona ou não o problema ou quão eficiente é a solução (em casos onde múltiplos passos podem ser realizados e nem todos comporem a solução). A função de Aptidão está diretamente atrelada ao problema e deve ser modelada conforme o cenário de aplicação. Neste trabalho calcula-se a Aptidão do indivíduo através de dois critérios, o tamanho e a eficiência da solução, conforme mostrado na Equação 3.1.

$$f = (1 - x) \times t_i + (x \times (E_i \times c)) \quad (3.1)$$

Onde:

- x é uma variável inteira assumindo valor 1 se o indivíduo i resolve o problema e 0 caso contrário;
- t_i é o tamanho do indivíduo i (ou seja, o número de genes)
- E_i é a eficiência do indivíduo i, calculada apenas quando $x = 1$ através da Equação 4.2;
- c é uma constante que determina o peso dado a uma solução que resolve o problema. Nesta proposta será utilizado $c = 25$, sendo 25 o tamanho máximo de um indivíduo.

Através da variável x, utilizando $(1 - x)$ e x, é possível zerar uma parte da equação, priorizando os indivíduos que possuem mais características genéticas (maiores indivíduos) ou os indivíduos mais eficientes. Isso permite que os indivíduos que possuem maior carga genética passem adiante parte de seus genes, além de permitir que os indivíduos que apresentem soluções também tenham um lugar de destaque na população através do cálculo da eficiência da solução (Equação 3.2).

$$E_i = t'_i / t_i \tag{3.2}$$

Onde:

- t_i é o tamanho do indivíduo i;
- t'_i é o tamanho do indivíduo considerando apenas as características que compõem o a solução do problema.

A Equação 3.2 apresenta a relação entre o tamanho da solução e a parte que compõem a solução do problema. Em diversas situações, um indivíduo pode representar a solução para um problema que demanda diversos passos. Alguns desses passos podem ser simplificados, não sendo necessário que eles apareçam no indivíduo. Nesse contexto, apenas os passos que estão diretamente ligados com a solução são considerados na variável t'_i.

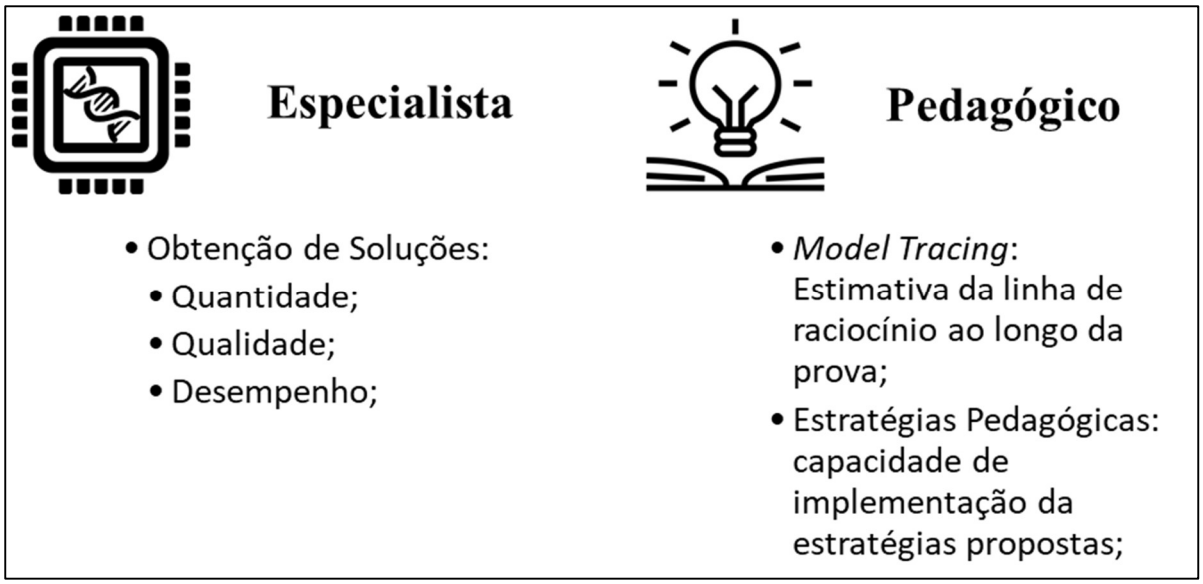
Ainda na Equação 3.1, a constante c, determinado a priori, serve como um viés para priorizar soluções mais eficientes em detrimento as menos eficientes ou a soluções que não resolvem o problema. Vale relembrar que soluções que não resolvem o problema são aceitáveis durante o processo evolutivo, uma vez que os problemas abordados neste trabalho não são triviais e demandariam algoritmos complexos e específicos para que uma solução pudesse ser obtida.

Por fim, destaca-se que nas aplicações de AGs em problemas de otimização, o objetivo está centrado em obter a melhor solução no menor tempo possível. Devido a característica do problema abordado neste trabalho e a finalidade do uso das soluções, o AG proposto no agente Especialista tem o objetivo de obter o maior número possível de soluções ótimas. Essas soluções permitem que seja possível acompanhar o aluno e aplicar estratégias de ensino-aprendizagem específicas para cada linha de raciocínio seguida.

4. PLANEJAMENTO DE EXPERIMETNOS E VALIDAÇÃO

O objetivo do EvoLogic é acompanhar, eficientemente, o aluno durante o processo de resolução de um exercício de DNLP. Para isso, faz-se necessário identificar as diferentes soluções existentes para um mesmo teorema a fim de acompanhar os passos realizados pelos alunos e categorizá-los de acordo com uma das soluções encontradas, provendo feedbacks de acordo com a necessidade. Nesse sentido, o processo de validação e experimentação divide-se em duas etapas. Na primeira etapa avalia-se a capacidade do EvoLogic em obter soluções, considerando a qualidade, quantidade e tempo de processamento demandado para resolver um determinado problema. Na etapa seguinte analisa-se como se dá o processo de acompanhamento dos alunos em um experimento conhecido, permitindo uma comparação direta com o Heráclito. A Figura 4.1 elenca as características do processo de validação do agente Especialista e do *model tracing*.

Figura 4.1 – Característica do processo de validação.



A primeira etapa tem o objetivo de avaliar as soluções obtidas pelo agente Especialista de acordo com quantidade, qualidade e desempenho. O agente Especialista consiste em um Algoritmo Genético que tem o objetivo de encontrar todas as soluções existentes para o problema em questão. Vale ressaltar que o interesse do Especialista é de obter as soluções para acompanhar os passos dos alunos e não somente resolver o problema, diferentemente do que é encontrado, em geral, na literatura sobre AG, onde o objetivo é obter a melhor solução no menor tempo possível.

Os experimentos apresentados a seguir utilizam uma base de dados conhecida, servindo como base de comparação, consistindo em 10 problemas apresentados a 54 alunos como pós-teste no trabalho de Galafassi (2019a). Desse modo, inicia-se os resultados apresentando as características dos dados, como:

- Teoremas: qual o teorema que se deseja provar.
- Total de soluções: quantos alunos conseguiram concluir a prova de determinado teorema;
- Total de soluções com passos extras: quantos alunos apresentaram soluções que continham passos extras;

- Total de soluções diferentes: como o objetivo é acompanhar os passos dos alunos de acordo com sua linha de raciocínio, busca-se identificar quantas soluções diferentes (eliminando-se os passos extras) foram apresentadas pelos alunos a fim de categorizá-las no futuro;
- Total de soluções diferentes existentes: estimativa do número de soluções diferentes (sem passos extras) que existem para cada exercício;
- Total de passos: quantidade de passos necessários para provar o teorema de acordo com as diferentes soluções.

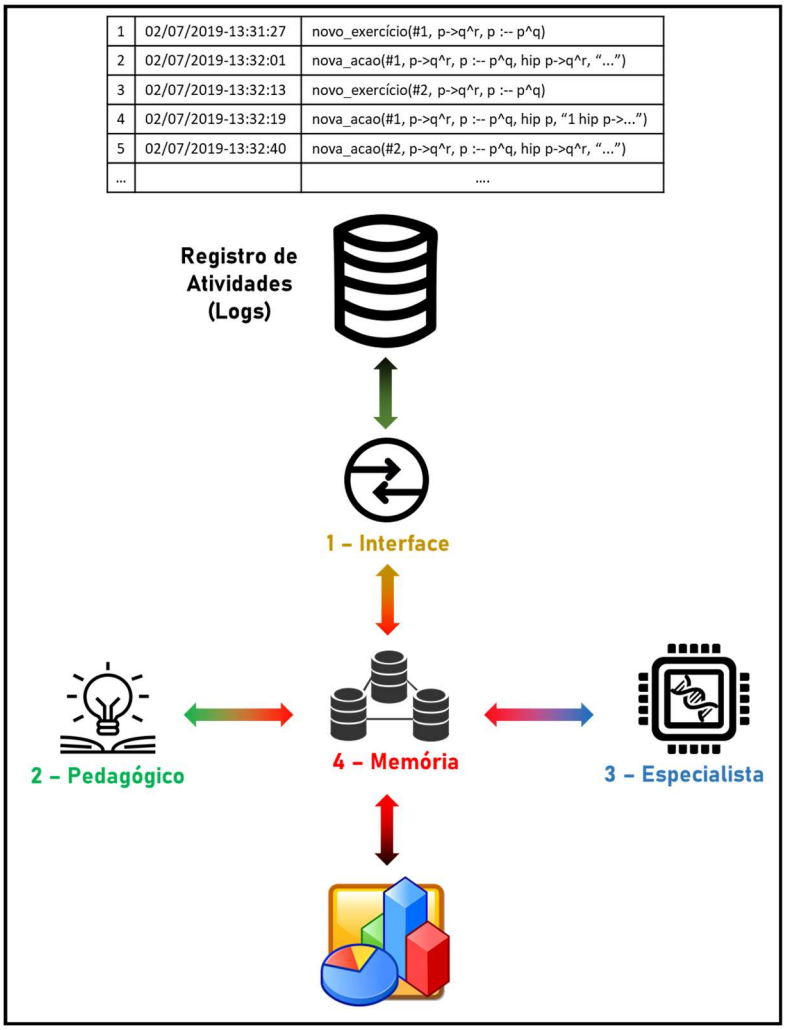
Após a discussão acerca dos exercícios e dos alunos, inicia-se a avaliação do agente Especialista. O processo começa através do estudo do tempo de processamento. Tendo como base diferentes critérios de parada, sendo eles o número de gerações populacionais (500, 1000, 1500 e 2000), é importante verificar se o tempo necessário para resolver o exercício é viável. Caso o agente Especialista necessite de mais tempo que o aluno para resolver o problema, o seu uso seria viável apenas se as atividades fossem planejadas e testada no ambiente a priori (por um professor ou tutor). Contudo, isso acaba tirando a autonomia do aluno de resolver diferentes exercícios, dado que não teria o suporte do sistema tutor. Além do tempo de processamento, também será avaliada a capacidade do agente em obter as diferentes soluções para o problema. Em outras palavras, não basta que o Especialista seja rápido o suficiente, ele precisa obter o maior número de soluções possíveis para possibilitar que o *model tracing* acompanhe o aluno e forneça feedbacks condizentes com o raciocínio que está sendo desenvolvido.

Uma vez que o agente Especialista consegue obter as soluções de forma satisfatórias, é preciso avaliar a capacidade do *model tracing* em identificar as diferentes linhas de raciocínio e fornecer feedbacks de acordo com a necessidade de cada aluno. Para fins de validação e experimentação, desenvolveu-se um ambiente simulado, onde todas as variáveis podem ser controladas e utiliza-se os registros de atividade dos alunos coletados por Galafassi (2019a). Cada ação individual dos alunos foi registrada, sendo elas:

- Novo exercício: o aluno iniciou a prova de um novo teorema;
- Nova ação: o aluno aplicou uma regra, desistiu da prova, ficou parado por muito tempo, solicitou ajuda entre outros;
- Fim do exercício: o aluno concluiu a prova do teorema;

Todos os registos são acompanhados do ID do aluno e da data/hora de ocorrência. Desse modo, cada ocorrência pode ser reproduzida simulando um cenário real, inclusive no que tange o tempo em que cada exercício foi iniciado e que cada ação é executada. Vale ressaltar que devido aos experimentos que abrangiam o agente Especialista, todos os exercícios já haviam sido resolvidos. Sendo assim, antes do início da simulação, a Memória de Longo Prazo foi limpa e todos os dados foram apagados, de modo a reproduzir um cenário mais próximo possível do real. A Figura 4.2 apresenta o esquema de funcionamento do processo de simulação do ambiente com os alunos interagindo com o EvoLogic. O processo de simulação consiste em enviar uma mensagem para o agente Interface contendo a ação realizada pelo aluno no tempo em que ela foi executada. Nesse contexto, os agentes devem armazenar as mensagens de entrada, gerenciar a resolução dos exercícios e acompanhar os passos dos alunos. Cada passo deve ser categorizado quanto a sua acurácia, a linha de raciocínio do aluno identificada e os dados armazenados na base de dados do EvoLogic para futura análise. Com isso, é possível verificar se os alunos apresentaram alguma solução que o agente Especialista não encontrou e se os passos foram identificados como extra assim que o aluno os aplicou ou se o Especialista ainda estava resolvendo o exercício.

Figura 4.2 – Representação gráfica do processo de validação do *model tracing*.



Além disso, foi desenvolvida uma interface simples, por linha de comando, com o objetivo de analisar alguns cenários onde as estratégias pedagógicas propostas são aplicadas. Esse processo consiste em avaliar se o EvoLogic apresenta os subsídios para que elas sejam desenvolvidas e como se comportariam em um ambiente real.

Por fim, uma vez que os dados dos experimentos com o ambiente Heráclito estão disponíveis, realiza-se uma comparação direta entre os dois STIs: EvoLogic e Heráclito. Essa comparação visa identificar as diferenças e os pontos fortes e fracos de cada abordagem no que tange a quantidade e qualidade das soluções obtidas, desempenho computacional, necessidade de espaço de armazenamento e capacidade em acompanhar os passos do aluno.

5. RESULTADOS

Os experimentos e validações foram conduzidos com base nos exercícios de pós-teste apresentados por Galafassi (2019a), os quais foram desenvolvidos no ambiente Heráclito. Vale lembrar que todos os dados acerca das ações dos alunos (início de uma nova prova, regra aplicada, ajuda solicitada, entre outros) foram coletados e permitem que o mesmo comportamento seja reproduzido. Desse modo, antes de apresentar os resultados obtidos pelo EvoLogic, é importante discutir algumas características dos exercícios analisados.

5.1 Agente Especialista

A Tabela 5.1 apresenta os teoremas propostos (de A a J), o total de alunos que concluíra satisfatoriamente a prova do teorema (Teoremas Provados), quantos desses alunos apresentaram soluções que continham passos extras (Passos Extras), o total de soluções diferentes desconsiderando os passos extras (Soluções Diferentes), uma estimativa de quantas soluções diferentes sem passos extras existem para o problema (Soluções Possíveis) e a quantidade mínima de passos necessários para provar o teorema.

Tabela 5-1 – Exercícios analisados.

Ex.	Teorema	Teoremas Provados	Passos Extras	Soluções Diferentes	Soluções Possíveis	Passos
A	`avb->c, c^a :-- avb^(a^c)`	43	15	4	4	7
B	`p->q^r, p :-- p^q`	47	14	1	1	5
C	`p^q, p->r, r^s->~t, q->s :-- ~t`	43	19	7	7	**10/20**
D	`b<->a, bvc->evq, b^a->c, ~e, b :-- q`	40	**26**	2	2	**8/13**
E	`(~b->~a)<->cvd, d, ~b, av(tv~~q) :-- tv~~qvr`	41	**32**	3	3	**10/15**
F	`a<->q, f<->r, a^r :-- f^q`	42	18	21	**80**	10
G	`f<->svd, s :-- f`	41	**23**	2	2	5
H	`a->c, a->b :-- a->c^b`	41	11	1	**2**	7
I	`s<->(v)^p :-- (v)->p->s`	41	**25**	**4**	**5**	**3/8**
J	`s->v, ~v :-- ~s`	**0**	0	0	1	6

Analisando a Tabela 5.1, pode-se observar que o exercício J não foi resolvido por nenhum aluno. Ao verificar, os registros de atividades dos alunos, constatou-se que 7 alunos iniciaram a sua resolução, mas não foram capazes de concluir a prova do teorema em tempo, sendo que 6 alunos já apresentavam passos extras nas provas parciais. Nos exercícios D, E, G e I pode-se constatar que mais de 20 alunos apresentaram soluções com passos extras. Verificou-se que nesses casos, a maioria dos alunos aplicaram a regra de Eliminação de Equivalência (para

ambos os lados)[12] mesmo quando não haveria a necessidade. Em contrapartida, no Exercício F (que possui menos soluções com passos extras), faz-se necessário a aplicação da regra Eliminação da Equivalência em ambos os lados. Ainda no exercício F, vale salientar que existe um grande número de soluções possíveis devido a quantidade de permutações entre alguns passos, ou seja, passos que podem ser realizados em ordem alternada. Continuando a análise da Tabela 5.1, os exercícios A, C, D, E e I possuem soluções que se utilizam de um conjunto de regras diferentes. Para esses teoremas, verificou-se que a quantidade de passos para provar o teorema varia de um caminho para outro (com exceção do exercício A). Nos demais exercícios, estima-se que exista apenas soluções que se utilizem das mesmas regras, variando apenas a ordem em que elas são apresentadas (por isso da mesma quantidade de passos). Como informação adicional, o tempo médio que os alunos levaram para resolver os exercícios foi de aproximadamente 12 minutos.

Com base no exposto, é importante salientar que o agente Especialista deve ser capaz de obter as diferentes soluções existentes para um mesmo teorema no menor tempo possível. Em outras palavras, o agente Especialista deve obter o maior número possível de diferentes soluções em tempo hábil para que o *model tracing* consiga acompanhar satisfatoriamente o aluno e prover os *feedbacks* necessários.

Sendo assim, inicia-se a análise do comportamento do agente Especialista avaliando o tempo médio de processamento de acordo com o critério de parada (número máximo de gerações populacionais). A Tabela 5.2 mostra o tempo médio necessário para resolver os exercícios (em segundos) e o seu desvio padrão para 500, 1000, 1500 e 2000 gerações populacionais. Cada exercício foi resolvido 30 vezes com cada critério de parada.

Tabela 5-2 – Comparação de tempos de processamento.

Ex.	500		1000		1500		2000	
	Média	Desv. Pad.	Média	Desv. Pad.	Média	Desv. Pad.	Média	Desv. Pad.
A	107,37	17,60	205,38	37,28	481,66	83,13	1393,81	272,49
B	110,53	16,63	211,41	33,66	495,80	96,48	1434,75	307,32
C	106,04	19,05	202,83	37,50	475,66	89,33	1376,47	272,82
D	108,07	18,70	206,71	31,96	484,77	85,61	1402,83	285,62
E	106,41	17,14	203,54	33,46	477,34	79,76	1381,33	251,12
F	106,89	17,69	204,46	35,04	479,50	95,61	1387,57	277,24
G	106,57	16,26	203,85	32,82	478,06	79,17	1383,41	258,14
H	105,08	18,03	200,99	34,17	471,35	81,17	1363,98	291,35
I	105,61	16,94	202,01	31,41	473,74	80,96	1370,92	255,13
J	104,22	16,73	199,35	30,52	467,51	76,58	1352,88	247,31

Pode-se verificar, analisando a Tabela 5.2, que o critério de parada (número máximo de gerações populacionais) possui grande influência no tempo total de processamento. Transformando genericamente os tempos de segundos para minutos, tem-se que o Especialista necessitou de menos de 2 minutos para concluir seu processamento com 500 gerações

[12] Considerando o exercício I, Hipótese: `s<->(v)^p`, pode-se aplicar a regra de Eliminação da Equivalência resultando em `s->(v)^p` ou `(v)^p->s`, contudo, apenas o segundo caso faz parte do caminho direto para a solução:

```
1 - hip       s<->(v)^p
2 - -eq                                          (v)^p->s   (1)
3 - exp      (v)->p->s (2)
```

populacionais, aproximadamente 3:30 para 1000 gerações, em torno de 8 minutos para 1500 e mais de 23 minutos para 2000 gerações populacionais. Pode-se verificar que apesar do critério dobrar crescer de forma linear, o tempo de processamento aumenta exponencialmente. Esse fenômeno pode ser explicado pelo aumento no tamanho médio da população ao longo do tempo. Uma vez que o agente Especialista possui uma etapa de identificação dos passos possíveis (antes das etapas de cruzamento e mutação), o aumento da população faz com que o número de possíveis passos também aumente, exigindo mais tempo de processamento a cada geração populacional. Considerando que os alunos necessitaram em torno de 12 minutos para resolver cada exercício, o critério de 2000 gerações populacionais pode ser descartado, dado que, além de necessitar mais do que o dobro do tempo dos alunos, em um cenário prático, mais de um exercício diferente pode ser iniciado ao mesmo tempo. Nesse mesmo sentido, se 2 exercícios fossem iniciados ao mesmo tempo, com o critério de 1500 gerações, seriam necessários aproximadamente 16 minutos para resolver os exercícios, mais do que o tempo médio que um aluno levaria para resolver um dos exercícios.

Além do tempo de processamento, é preciso que o agente Especialista encontre todas as soluções existentes para um problema (ou o maior número possível), de modo que o *model tracing* possa acompanhar as diferentes linhas de raciocínio. Sendo assim, a segunda parte da análise do Especialista consiste em avaliar, para as 30 execuções com cada critério de parada, quantas soluções foram obtidas. A Tabela 5.3 apresenta a quantidade média de soluções sem (Diretas) e com passos extras (Extras) para cada critério de parada.

Tabela 5-3 – Comparação de quantidade de soluções em relação ao critério de parada.

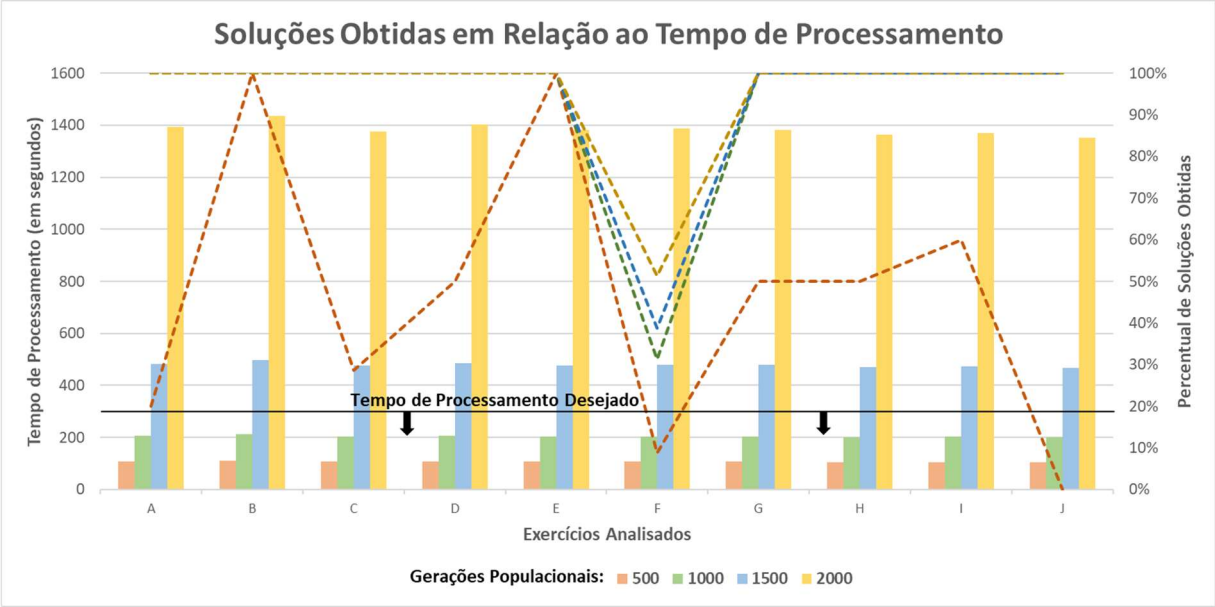
Ex.	Est.	Soluções							
		500		1000		1500		2000	
		Diretas	Extras	Diretas	Extras	Diretas	Extras	Diretas	Extras
A	4	1,0	3,1	4	0	4	0	4	0
B	1	1	0	1	0	1	0	1	0
C	7	1,9	3,3	7	0	7	0	7	0
D	2	1	0	2	0	2	0	2	0
E	3	1,8	0,5	3	0	3	0	3	0
F	80	7,5	14,9	25,1	27,7	31,8	28,0	41,4	31,1
G	2	1,1	0,6	2	0	2	0	2	0
H	2	1,4	0,3	2	0	2	0	2	0
I	5	3,1	1,1	5	0	5	0	5	0
J	1	0	0,4	1	0	1	0	1	0

Observando a Tabela 5.3, no que tange o critério de parada de 500 gerações populacionais, apenas no exercício B foi possível obter, de forma consistente, a solução direta para o problema. Contudo, ao analisarmos os critérios de 1000 gerações ou mais, obteve-se todas as soluções nas 30 execuções para a maioria dos problemas (exceto o exercício F). Conforme comentado anteriormente, no exercício F tem-se um conjunto de regras que podem ser permutadas, gerando uma grande combinação de soluções, mas que consistem nas mesmas regras. No tocante aos demais exercícios, vale salientar que, mesmo com o critério de 1000 gerações populacionais, foi possível obter todas as soluções estimadas, incluindo as soluções que não foram apresentadas pelos alunos no experimento.

Para resumir os dados das Tabelas 5.2 e 5.3 e permitir uma análise visual, a Figura 5.1 apresenta um gráfico de colunas, mostrando o tempo de processamento em segundos, e um gráfico de linhas que mostra o percentual de soluções obtidas (relação entre a quantidade

estimada e a quantidade obtida), separando os dados de acordo com o critério de parada (número máximo de gerações populacionais).

Figura 5.1 – Representação gráfica do processo de validação do *model tracing*.



Pode-se observar, na Figura 5.1, que apenas as gerações populacionais 500 e 1000 obtiveram um tempo de processamento inferior a 5 minutos (300 segundos), sendo que esse é a metade do tempo médio que os alunos necessitaram por exercício. Além disso, é possível observar a variação do percentual de soluções obtidas. Nesse sentido, o critério de 500 gerações obteve resultados inferiores que o critério de 1000 gerações, o qual obteve 100% em 9 dos 10 exercícios estudados.

Com base nas características de tempo e soluções analisadas, adota-se o critério de parada de 1000 gerações populacionais para os próximos experimentos. Nesse sentido, o Tabela 5.4 sintetiza os resultados obtidos pelos alunos e pelo Especialista, considerando o critério de parada adotado. A tabela mostra a quantidade de soluções diferentes diretas apresentadas pelos alunos (Alunos) e pelo agente Especialista (Especialista).

Tabela 5-4 – Comparação de soluções.

Exercício	Alunos	Especialista
A	4	**5**
B	1	1
C	7	7
D	2	2
E	3	3
F	21	25
G	2	2
H	1	**2**
I	4	**5**
J	0	1

Na Tabela 5.4 pode-se verificar que o agente Especialista obteve uma quantidade maior de soluções do que as apresentadas pelos alunos em todos os exercícios. No caso dos exercícios A e H, o Especialista identificou que os alunos poderiam ter apresentado uma ordem de passos

diferente em uma das soluções. Além disso, no exercício I, todos os alunos apresentaram soluções que se utilizam de provas condicionais, exigindo no mínimo 6 passos. Contudo, existe uma solução mais curta, utilizando apenas 3 passos, que foi identificada pelo Especialista, apesar dos alunos não terem seguido essa linha de raciocínio.

5.2 Model Tracing

O objetivo do *model tracing* é acompanhar os passos dos alunos e identificar características que indiquem que esse aluno possa estar com alguma dificuldade em continuar a prova do teorema, como por exemplo, os passos extras realizados ao longo do exercício. Quanto um exercício desconhecido é iniciado, o EvoLogic necessita resolvê-lo antes que seja possível fornecer algum feedback ao aluno. Nesse sentido, algumas vezes, os passos iniciais não podem ser categorizados, sendo armazenados para avaliação a posteriori.

Para avaliar a capacidade do *model tracing* proposto, realiza-se uma simulação utilizando dados reais de 54 alunos, onde os exercícios foram iniciados e os passos foram iniciados com o mesmo intervalo de tempo que ocorreu no experimento de Galafassi (2019a). A Tabela 5.5 apresenta a quantidade total de passos extras identificados pelo ambiente Heráclito, servindo como linha base de comparação, a quantidade de passos extras que foram identificados pelo EvoLogic em tempo de execução e a posteriori. Os passos classificados a posteriori consistem em passos que foram armazenados no início da prova do teorema, dado que ainda não haviam sido obtidas soluções para o exercício. Ressalta-se que todos os passos foram corretamente identificados pelo EvoLogic, ou seja, nenhum passo foi classificado erroneamente como extra e nem um passo extra foi ignorado.

Tabela 5-5 – Passos extras realizados e acompanhados.

Exercício	Heráclito	EvoLogic	
		Tempo de Execução	A Posteriori
A	26	18	**8**
B	24	20	**4**
C	25	25	-
D	31	30	**1**
E	33	33	-
F	20	20	-
G	27	27	-
H	23	22	**1**
I	35	35	-
J	4	3	**1**
TOTAL	210	195 93%	15 7%

Vale salientar que mesmo que apenas 1 aluno inicie o exercício, podem ocorrer passos extras antes que o EvoLogic consiga obter todas as soluções. Isso ocorre pois o EvoLogic necessita de aproximadamente 3 minutos para resolver o exercício e obter a maioria das soluções. Devido a isso, analisando a Tabela 5.5, pode-se observar que os exercícios A e B tiveram mais passos extras sendo realizados sem que fossem identificados em tempo de execução. Isso acontece, pois, a maioria dos alunos iniciaram a resolução pelos exercícios A e B, sendo que não havia uma ordem obrigatória a seguir. O mesmo ocorreu com o exercício D, onde 5 alunos iniciaram a sua resolução com menos de 15 segundos de diferença entre si. No exercício H, temos o caso

apontado anteriormente, onde o primeiro aluno que iniciou a prova do teorema realizou um passo extra (no seu segundo passo). Por fim, o exercício J apresenta um cenário diferenciado, onde o primeiro aluno também apresentou um passo extra antes do EvoLogic ter obtido as soluções. Contudo, nenhum aluno conseguiu concluir a prova do teorema do exercício J, devido à complexidade e a falta de tempo.

Ainda considerando a Tabela 5.5, observa-se que 93% dos passos extras foram identificados em tempo de execução (assim que foram realizados), assim como ocorre no Heráclito. Essa capacidade indica que é possível prover feedback aos alunos conforme a necessidade e os planos de ação desenvolvidos para cada estratégia de aprendizagem. Nesse sentido, a diferença entre estratégias pedagógicas presentes no Heráclito e as existentes no EvoLogic se caracteriza pela capacidade de ter diferentes soluções possíveis a priori, identificando a linha de raciocínio do aluno e permitindo que outros conceitos pedagógicos possam ser empregados (como o ensino baseado em exemplos).

5.2.1 Análise de exercícios resolvidos

Para analisar o comportamento dos alunos e as aferições feitas pelo *model tracing* busca-se identificar detalhes do processo de resolução de dois exercícios, evidenciando como o *model tracing* do EvoLogic categoriza cada ação do aluno. Nesse sentido, serão apresentados alguns exemplos de resolução de dois exercícios, destacando como EvoLogic e o Heráclito se comportaram em cada passo. Os exercícios estudados são:

A. (A∨B)->C, C∧A |- (A∨B)∧(A∧C)

I. S<->(V)∧P |- (V)->P->S

Inicia-se a análise pelo primeiro exercício ((A∨B)->C, C∧A |- (A∨B)∧(A∧C)), onde a Figura 5.2 apresenta as soluções possíveis identificadas pelo EvoLogic.

Figura 5.2 – Soluções obtidas pelo EvoLogic para o exercício A.

Solução 1	
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP A (2)
4	ADD A∨B (3)
5	MP C (1, 4)
6	CJ A∧C (3, 5)
7	**CJ(A∨B)∧(A∧C) (4, 6)**

Solução 2	
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP A (2)
4	SP C (2)
5	ADD A∨B (3)
6	CJ A∧C (3, 4)
7	**CJ(A∨B)∧(A∧C) (5, 6)**

Solução 3	
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP C (2)
4	SP A (2)
5	ADD A∨B (4)
6	CJ A∧C (3, 4)
7	**CJ(A∨B)∧(A∧C) (5, 6)**

Solução 4	
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP A (2)
4	ADD A∨B (4)
5	SP C (2)
6	CJ A∧C (3, 4)
7	**CJ(A∨B)∧(A∧C) (5, 6)**

Analisando a Figura 5.2, pode-se verificar que as 4 soluções apresentam 2 conjuntos distintos de regras. Em outras palavras, a solução 1 apresenta as regras Simplificação, Adição, Modus Ponens e Conjunção, enquanto as soluções 2, 3 e 4 apresentam as regras Simplificação, Adição e Conjunção (não utilizam a regra Modus Ponens). Para ilustrar o macro comportamento dos alunos ao resolverem o exercício, a Tabela 5.6 mostra, para cada solução, quantos alunos a obtiveram, quantos passos extras foram realizados e quantos foram categorizados a posteriori.

Tabela 5-6 – Passos extras realizados e acompanhados.

Solução	Alunos	Passos Extras
1	13	3 (1)
2	12	6 (2)
3	8	4 (3)
4	10	5 (2)
Total	43	18 (8)

Analisando a Tabela 5.6, pode-se observar que, de forma geral, para esse exercício, a quantidade de passos extras não variou muito entre as diferentes soluções. No total, 15 alunos realizaram passos extras (Tabela 5.1), sendo que 2 alunos realizaram 3 passos e 7 alunos apresentaram 2 passos extras, enquanto os demais (6 alunos) apresentaram apenas 1. Quanto aos passos que foram identificados a posterior (entre parênteses na tabela), todos foram realizados logo no início do exercício, antes do EvoLogic tê-lo resolvido, e nenhum aluno apresentou mais de 1 passo extra antes deles terem sido categorizados.

Para analisar o funcionamento do *model tracing* e suas conclusões a partir dos passos e das soluções conhecidas, apresenta-se o caso de um aluno (aqui denominado de Aluno #1) de que realizou 3 passos extras ao tentar resolver o exercício A. A Tabela 5.7 apresenta a solução parcial até o passo 5!, onde o aluno apresenta o primeiro passo extra.

Tabela 5-7 – Primeiro passo extra do aluno #1 no exercício A.

	Exercício A – Aluno #1
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP A (2)
4	SP C (2)
5!	*ADD A∨C (3)*

Ao realizar o passo 5, o aluno pode ter indicado que está com dúvidas ou que foi apenas um erro de digitação, dado que, em algum momento, será necessário aplicar a regra Adição à mesma linha (linha 3) resultando a A∨B. Além disso, ao realizar o passo 4 (SP C (2)), o aluno indica que sua linha de raciocínio segue a Solução 2 da Figura 5.2. A seguir, nos passos 6! e 7!, o aluno apresenta outros 2 passos extras, conforme mostrado na Tabela 5.8.

Tabela 5-8 – Passo extra do aluno #1 no exercício A.

	Exercício A – Aluno #1
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP A (2)
4	SP C (2)
5!	*ADD A∨C (3)*
6!	*HIPPC A∨B (A∨B)->C*
7!	*MP C (1, 6)*

Na Tabela 5.8 percebe-se o segundo passo extra (passo 6) quando o aluno tenta realizar uma Prova Condicional. Em seguida, ele aplica a regra Modus Ponens obtendo C novamente. Nesse momento, o aluno identifica qual passo ele deveria ter feito e retrocede os passos, para antes da Prova Condicional, e continua a resolução até o final, conforme mostrado na Tabela 5.9.

Tabela 5-9 – Solução apresentada pelo aluno #1 no exercício A.

	Exercício A – Aluno #1
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP A (2)
4	SP C (2)
5!	*ADD A∨C (3)*
~~*6!*~~	~~*HIPPC A∨B (A∨B)->C*~~
~~*7!*~~	~~*MP C (1, 6)*~~
6	ADD A∨B (3)
7	CJ A∧C (3, 4)
8	**CJ(A∨B)∧(A∧C) (6, 7)**

Ao analisar a Tabela 5.9, pode-se verificar que o aluno não retrocedeu para antes do passo 5 (ADD A∨C (3)), mantendo esse passo extra na solução. Vale lembrar que o fato de existirem passos extras que não impeçam a conclusão da prova, não a tornam errada ou inválida.

Ainda no exercício A, o Aluno #2 inicia a prova do teorema de forma semelhante, cometendo o mesmo passo extra (passo 5), mas em seguida volta a seguir a sua linha de raciocínio e apresenta a mesma solução que o Aluno #1 (Tabela 5.8). No caso do Aluno #2, pode-se considerar que o passo extra serviu para mostrar como ele deveria prosseguir, dado que deveria aplicar a mesma regra (ADD) do passo 5. Essa conjectura é reforçada pois o aluno levou aproximadamente 50 segundos para realizar o próximo passo. Caso ele tivesse cometido um erro de digitação, o próximo passo poderia ter sido refeito em poucos segundos.

Por fim, a Tabela 5.10 apresenta mais um caso acerca do exercício A, onde o aluno apresenta um passo extra. Contudo, esse cenário se difere dos demais pois trata-se de um passo que aparece em outras linhas de raciocínio.

Tabela 5-10 – Mudança de linha de raciocínio do aluno #3 no exercício A.

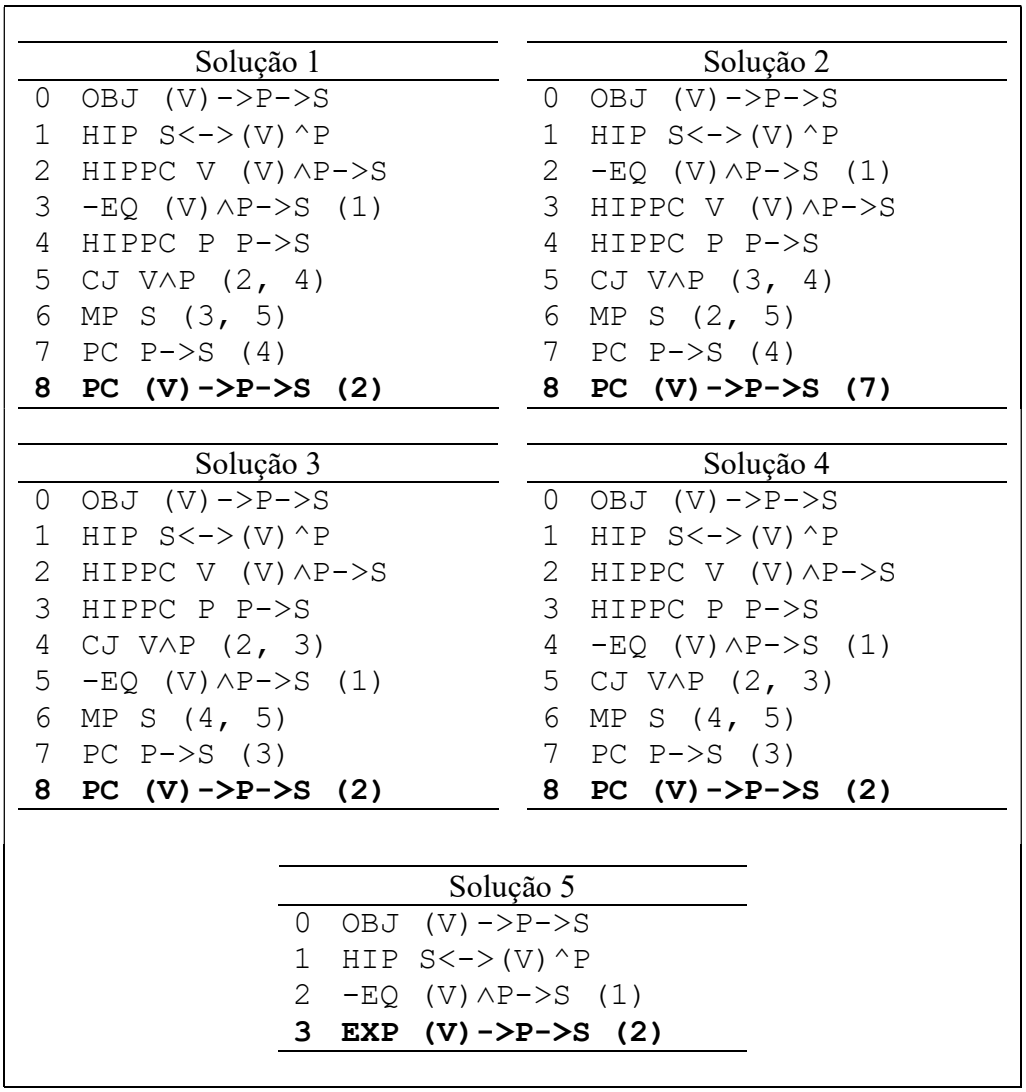
Exercício A – Aluno #3	
0	OBJ (A∨B)∧(A∧C)
1	HIP (A∨B)->C
2	HIP C∧A
3	SP C (2)
4	SP A (2)
5	ADD A∨B (4)
6!	*MP C (1, 5)*
7	CJ A∧C (3, 4)
8	**CJ(A∨B)∧(A∧C) (5, 6)**

Na Tabela 5.10, até o passo 4 (`SP A (2)`) o Aluno #3 indica que está seguindo a linha de raciocínio da Solução 3 (Figura 5.2), contudo, ao aplicar a regra `MP C (1, 5)` (passo 6) ele demonstra uma mudança em sua forma de pensar. Nesse momento, o aluno parece não perceber que já possuía a premissa C, obtido no passo 3 (`SP C (2)`). Contudo, os próximos passos são realizados com alguns segundos de diferença entre si, indicando que nesse momento o Aluno #3 identificou uma possível solução para o exercício. Além disso, ao aplicar o passo 6, o EvoLogic passa a considerar que o aluno possa ter mudado sua linha de raciocínio e passa a considerar o passo 3 como sendo um passo extra, diferentemente do Heráclito que classifica o passo 6 como extra.

O cenário apresentado na Tabela 5.10 (que ocorreu 3 vezes de formas similares) pode apontar uma desatenção do aluno ao não perceber que já possui a premissa `C`, oriunda da regra Simplificação, ou pode indicar que ele mudou sua forma de pensar sobre a forma de resolver o problema. Desse modo, o EvoLogic classificou o passo 3 (`SP C (2)`) como extra e mudou a linha de raciocínio que estava utilizando para acompanhar o aluno. Vale salientar que manter o passo 3 como passo extra pode ser útil caso venha a ser implementada algum plano pedagógico e considere a quantidade de passos extras. Apesar disso, pode-se considerar que nesse caso não seria necessário realizar nenhum processo de intervenção pois ele seguiu uma nova linha de raciocínio.

A seguir, continua-se a análise com o exercício I (`S<->(V)∧P |- (V)->P->S`) que possui 5 soluções diferentes, apresentadas na Figura 5.3. Analisando a figura, pode-se verificar que as 4 primeiras soluções apresentam o mesmo conjunto de regras, variando somente a ordem em que elas aparecem, enquanto que a Solução 5 apresenta o uso de uma regra derivada, chamada de Exportação.

Figura 5.3 – Soluções obtidas pelo EvoLogic para o exercício I.



Solução 1

```
0  OBJ (V)->P->S
1  HIP S<->(V)^P
2  HIPPC V (V)∧P->S
3  -EQ (V)∧P->S (1)
4  HIPPC P P->S
5  CJ V∧P (2, 4)
6  MP S (3, 5)
7  PC P->S (4)
8  PC (V)->P->S (2)
```

Solução 2

```
0  OBJ (V)->P->S
1  HIP S<->(V)^P
2  -EQ (V)∧P->S (1)
3  HIPPC V (V)∧P->S
4  HIPPC P P->S
5  CJ V∧P (3, 4)
6  MP S (2, 5)
7  PC P->S (4)
8  PC (V)->P->S (7)
```

Solução 3

```
0  OBJ (V)->P->S
1  HIP S<->(V)^P
2  HIPPC V (V)∧P->S
3  HIPPC P P->S
4  CJ V∧P (2, 3)
5  -EQ (V)∧P->S (1)
6  MP S (4, 5)
7  PC P->S (3)
8  PC (V)->P->S (2)
```

Solução 4

```
0  OBJ (V)->P->S
1  HIP S<->(V)^P
2  HIPPC V (V)∧P->S
3  HIPPC P P->S
4  -EQ (V)∧P->S (1)
5  CJ V∧P (2, 3)
6  MP S (4, 5)
7  PC P->S (3)
8  PC (V)->P->S (2)
```

Solução 5

```
0  OBJ (V)->P->S
1  HIP S<->(V)^P
2  -EQ (V)∧P->S (1)
3  EXP (V)->P->S (2)
```

Assim como no exercício anterior, para ilustrar o macro comportamento dos alunos ao resolverem o exercício I, a Tabela 5.11 mostra, para cada solução, quantos alunos obtiveram e quantos passos extras foram realizados. Vale ressaltar que, nesse caso, todos os passos extras foram realizados após o EvoLogic resolver o exercício, em outras palavras, esses passos foram categorizados assim que foram aplicados.

Tabela 5-11 – Passos extras realizados e acompanhados.

Solução	Alunos	Passos Extras
2	15	13
3	11	7
4	12	6
5	3	9
Total	41	35

Dos 41 alunos que concluíram a prova corretamente, 15 apresentaram a Solução 2, 11 a Solução 3, 12 a solução 4 e 3 a solução 5. Dentre esses, 25 alunos (Tabela 5.1) apresentaram passos extras em suas soluções, sendo que 1 aluno realizou 5 passos, 1 aluno realizou 4 passos,

3 alunos apresentaram 2 passos extras enquanto que os demais (20 alunos) apresentaram apenas 1.

Analisando a Tabela 5.11, percebe-se que a Solução 1 não foi apresentada. De modo geral, a solução 1 apresenta apenas uma ordem de passos diferente das soluções 2, 3 e 4. Além disso, dois cenários necessitam ser melhor explorados: Solução 2 e Solução 5. No caso da Solução 2, dos 15 alunos, foram verificados 13 passos extras. Enquanto isso, ao observar a solução 5, apenas 3 alunos a apresentaram, sendo que apenas 2 desses alunos realizaram 9 passos extras. A análise dessas ocorrências se inicia pelo aluno #4, que apresentou a solução 5 contendo um total de 4 passos extras.

Tabela 5-12 – Passo extra apresentado pelo aluno #4 no exercício I.

Exercício I – Aluno #4	
0	OBJ (V)->P->S
1	HIP S<->(V)^P
2	HIPPC V (V)∧P->S
3	HIPPC P P->S
4	CJ V∧P (2, 3)
5!	*-EQ S->(V)∧P (1)*

Analisando a Tabela 5.12 pode-se verificar que o aluno segue a linha de raciocínio da Solução 3 até o passo 5, quando apresenta um passo extra. A regra de Eliminação de Equivalência pode ser aplicada a ambos os lados da Hipótese (`HIP S<->(V)^P`). Nesse caso, ele pode estar confuso ou ter apenas selecionado a opção errada. Poucos segundos depois de realizar o passo 5, o Aluno #4 realizar o passo 6 corretamente (`-EQ (V)∧P->S (1)`), visto na Tabela 5.13.

Tabela 5-13 – Solução parcial apresentada pelo aluno #4 no exercício I.

Exercício I – Aluno #4	
0	OBJ (V)->P->S
1	HIP S<->(V)^P
2	HIPPC V (V)∧P->S
3	HIPPC P P->S
4	CJ V∧P (2, 3)
5!	*-EQ S->(V)∧P (1)*
6	-EQ (V)∧P->S (1)

Ao voltar para a sua linha de raciocínio, o aluno percebe que poderia concluir a prova em apenas 1 passo, aplicando a regra Exportação, conforme mostrado na Tabela 5.14.

Tabela 5-14 – Solução final apresentada pelo aluno #4 no exercício I.

Exercício I – Aluno #4	
0	OBJ (V)->P->S
1	HIP S<->(V)^P
2!	*HIPPC V (V)∧P->S*
3!	*HIPPC P P->S*
4!	*CJ V∧P (2, 3)*
5!	*-EQ S->(V)∧P (1)*
6	-EQ (V)∧P->S (1)
7	**EXP (V)->P->S (6)**

Ao resolver o problema utilizando apenas o passo 6 (além da hipótese), o EvoLogic categoriza os demais passos como sendo passos extras. Essa classificação fica armazenada na Memória de Longo Prazo para que, caso seja necessário, possa incorporar alguma estratégia de mediação pedagógica. Vale salientar que, quando o aluno conclui a prova, o Heráclito não apresenta nenhuma informação acerca dos passos extras realizados anteriormente.

O Aluno #5 apresentou 5 passos extras (Tabela 5.15), similar ao que o aluno #4 apresentou na Tabela 5.12, onde ele também identificou que após aplicar erroneamente a regra Eliminação da Equivalência, poderia reaplicar a mesma regra e concluir a prova.

Tabela 5-15 – Solução apresentada pelo aluno #5 no exercício I.

Exercício I – Aluno #4	
0	OBJ (V)->P->S
1	HIP S<->(V)^P
2!	*HIPPC V (V)∧P->S*
3!	*HIPPC P P->S*
4!	*CJ V∧P (2, 3)*
5!	*ADD V∨P (2)*
6!	*-EQ S->(V)∧P (1)*
7	-EQ (V)∧P->S (1)
8	**EXP (V)->P->S (7)**

Assim como no caso do aluno #4, os passos que, inicialmente seguiam a Solução 3, foram classificados como passos extras, uma vez que agora foi apresentada a Solução 5.

Voltando a Tabela 5.11, na solução 2 foi identificado 13 passos extras. Ao analisar o cenário, verificou-se que os alunos iniciaram a resolução do problema através da regra Eliminação da Equivalência (`-EQ (V)∧P->S (1)`), de modo que sempre que um aluno realizou esse passo, o Heráclito emitiu uma mensagem motivacional indicando que faltavam apenas 1 passo para concluir a prova. Nesse sentido, muitos alunos tentam identificar qual regra deveriam aplicar (Exportação), tentando aplicar outras regras sem sucesso e resultando em passos extras. Em contrapartida, o EvoLogic identificou que ao aplicar a regra de Eliminação da Equivalência, haviam duas possíveis linhas de raciocínio, uma com menos passos (Solução 5) e outra mais longa (solução 2). Essa capacidade de identificação permite que um feedback mais especializado seja fornecido, evitando confundir o aluno.

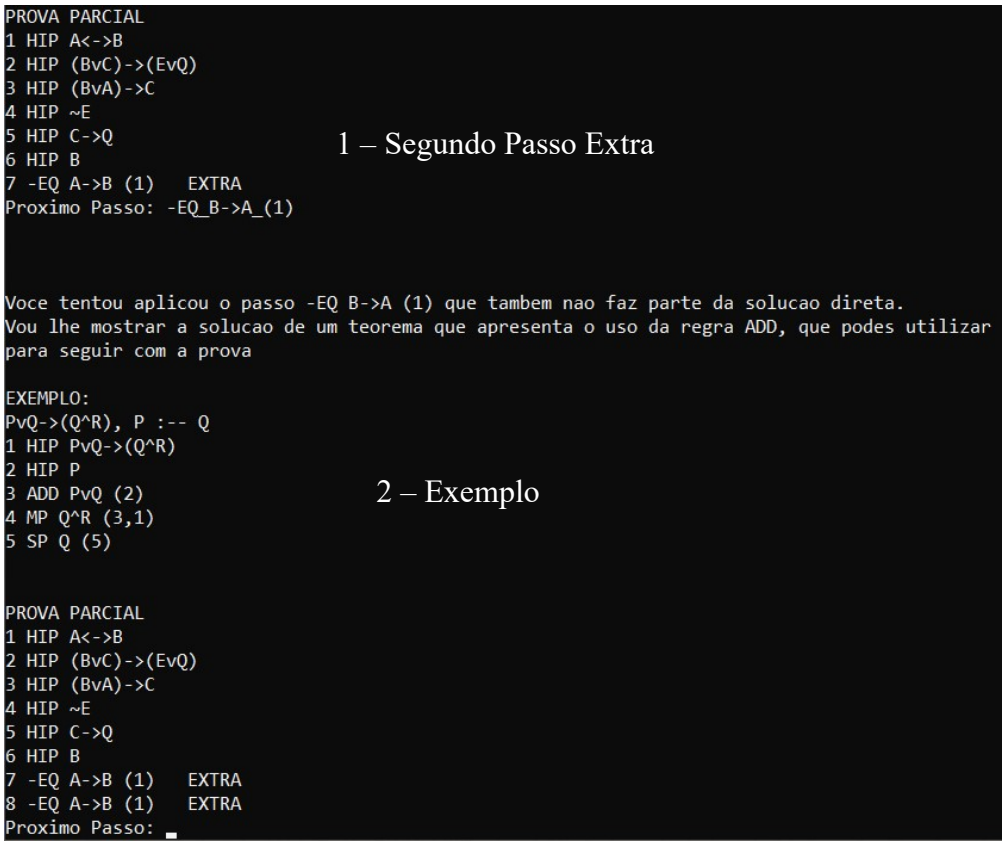
Por fim, os demais passos extras desses exercícios analisados, bem como nos demais exercícios, consistem em passos sem objetivo concreto, onde o aluno está explorando as possibilidades para retomar o caminho da prova, ou cenários similares onde ocorrem mudanças nas linhas de raciocínio.

5.3 Estratégias Pedagógicas

As estratégias pedagógicas são parte fundamental dos STIs, servindo como parte indispensável no auxílio aos alunos quando eles estiverem apresentando dificuldades. Para analisar a viabilidade do uso das estratégias baseadas em exemplos, propostas na seção 3.4.1, foi criada uma interface de linha de comando, onde os casos são simulados. Dentre as quatro estratégias, a EBL4 (vincular exemplos a princípios subjacentes) exige uma interface mais rebuscada, com mais formas de interação e por isso não será considerada neste estudo. A análise das estratégias se inicia pela EBL 3 (solicitar aos alunos que façam conexões entre exemplos e

princípios), onde tenta-se resolver o exercício D (`B<->A, BvC->EvQ, B^A->C, ~E, B :-- Q`) realizando dois passos extras (Figura 5.4).

Figura 5.4 – EBL3 aplicada ao exercício D.

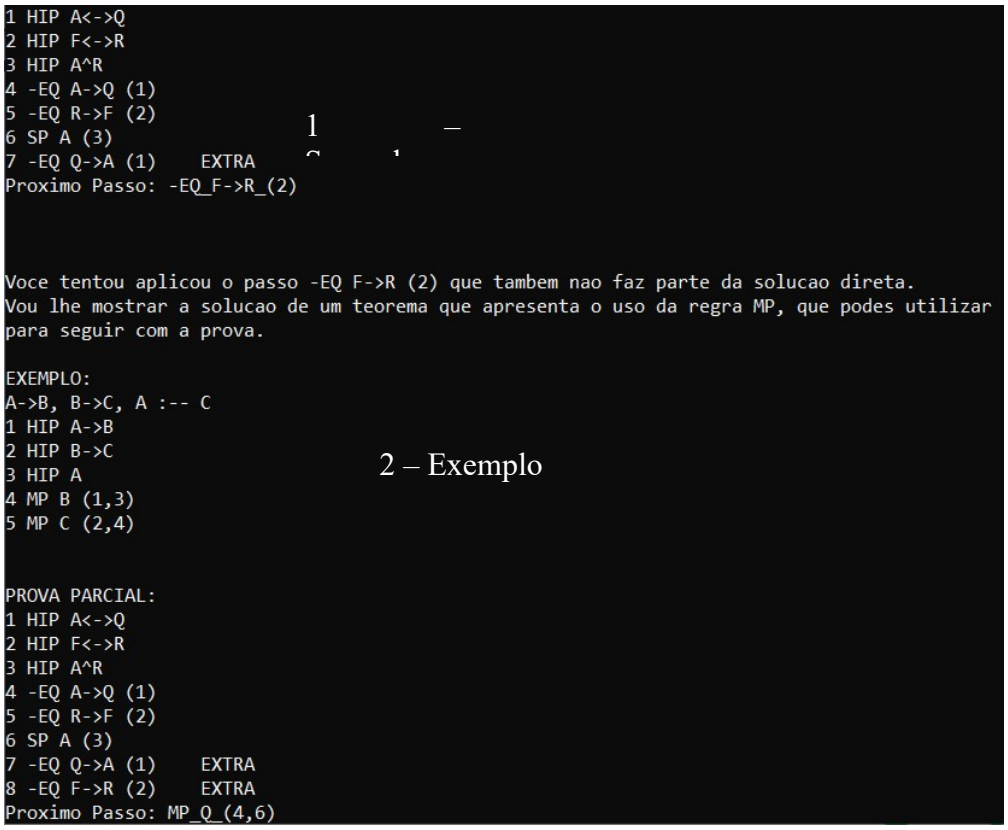


Na Figura 5.4, item 1, mostra-se a tentativa de realização de passo válido, mas que se caracteriza como extra, dado que não faz parte do caminho direto até a solução. Dado que a ocorrência repetida desses passos pode indicar que o aluno está com alguma dificuldade em provar o teorema, utiliza-se a estratégia EBL3 para indicar uma possível regra a ser aplicada, mostrando um exemplo de uso (regra ADD).

Além disso, na Figura 5.4, item 2, mostra-se um exemplo encontrado que contenha a regra que se deseja mostrar ao aluno. Esse exercício já havia sido resolvido previamente pelo EvoLogic e foi o primeiro resultado obtido com a regra ADD.

O segundo exemplo, ainda na estratégia EBL3, segue a resolução do exercício F (`A<->Q, F<->R, A∧R :-- F∧Q`). A Figura 5.5 mostra a tentativa de aplicação do segundo passo extra, seguido de uma sugestão de qual passo realizar para continuar com a prova do teorema.

Figura 5.5 – EBL3 aplicada ao exercício F.

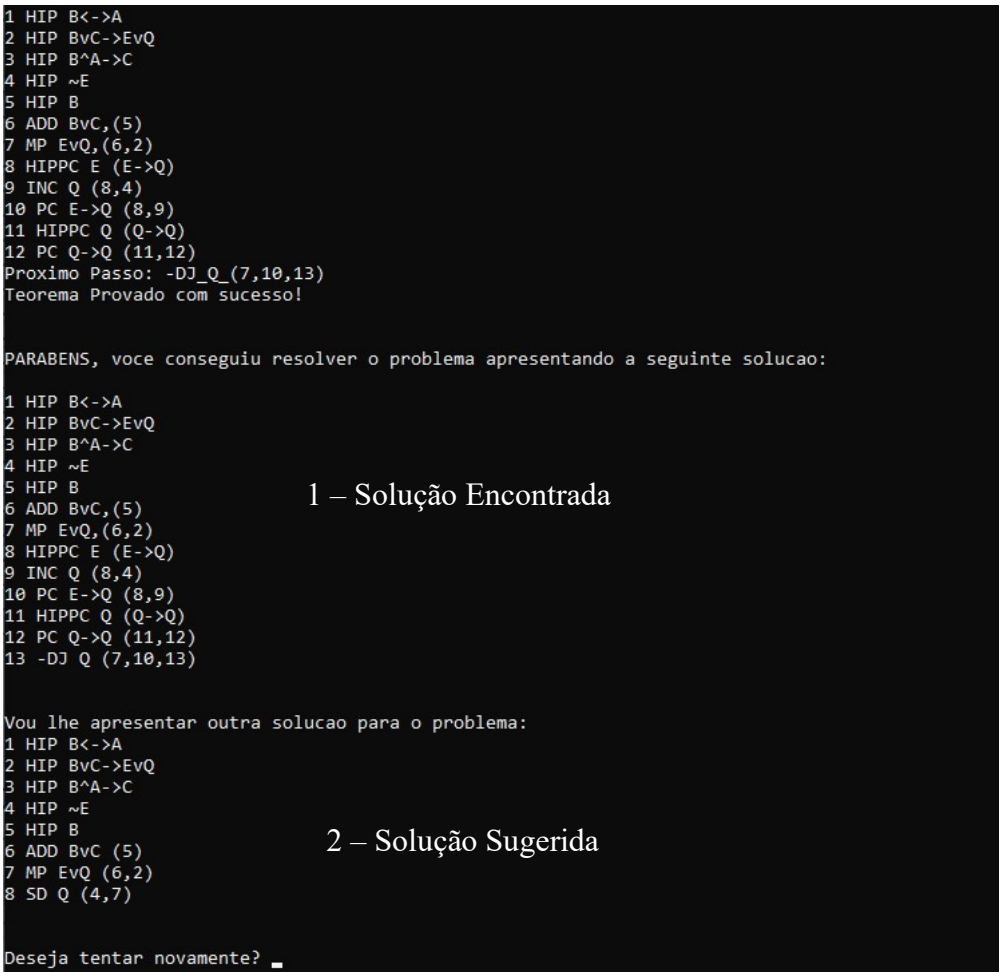


Na Figura 5.5, item 1, o aluno aplica a regra -EQ, sendo classificado como um passo extra. Ao identificar as possíveis linhas de raciocínio do aluno pode seguir, sugere-se a aplicação da regra MP, onde mostra-se um exemplo de uso da regra (passo 4, item 2 da figura).

Analisando os casos das duas Figuras (5.4 e 5.5), pode-se verificar que os exemplos mostram o uso da regra sugerida. Contudo, nenhum desses exemplos levam em consideração o contexto em que a regra é aplicada. No caso da Figura 5.5, ao saber que a regra MP deve ser aplicada para que em seguida seja aplicada a regra CJ, assim finalizando a prova do teorema, poderia ser sugerido um exemplo de uso da regra MP em que seu resultado seja aplicado a regra CJ (caso exista esse exemplo na Memória de Longo Prazo).

A segunda estratégia analisada é a EBL1 (aprender com exemplos de um caso particular), onde apresenta-se soluções que utilizam um conjunto diferente de regras para resolver o mesmo exercício. Nesse sentido, a Figura 5.6 mostra essa estratégia sendo usada após a conclusão do exercício A (A∨B->C, C∧A :-- A∨C∧ (A∧C)).

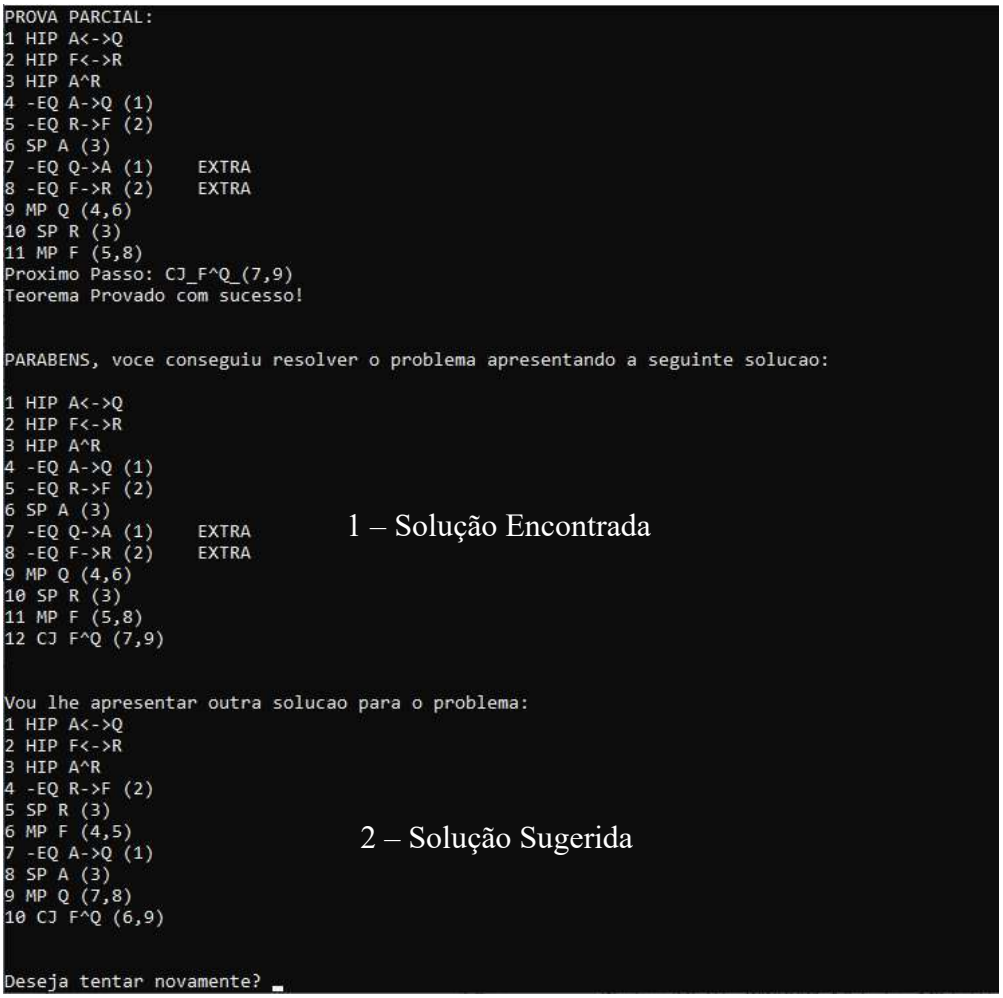
Figura 5.6 – EBL1 aplicada ao exercício A.



Na Figura 5.6 pode-se verificar, no item 1, que o aluno obteve uma solução que contém 13 passos. Assim que o teorema foi provado, o EvoLogic identificou que havia outras soluções para o mesmo exercício, sendo que uma delas utilizava outro conjunto de regras. Quando esse cenário ocorre, utiliza-se do conceito da EBL1 para mostrar uma solução que se utiliza de regras diferentes das apresentadas pelo aluno. A intenção é estimular que o aluno identifique as diferenças e perceba que poderia ter utilizado regras diferentes e, caso ainda não as conheça, passe a ver um exemplo de uso delas. Nesse contexto, a EBL1 ainda pode ser aprimorada de modo a sugerir novos exercícios que utilizem algumas das regras da solução mostrada.

Por fim, a terceira estratégia analisada, EBL2 (fornecer vários exemplos), tem o objetivo de mostrar soluções que utilizem o mesmo conjunto de regras combinadas de forma diferente da apresentada. A Figura 5.7 apresenta um caso para o exercício F (A<->Q, F<->R, A∧R :- F∧Q), onde o aluno uma possível solução para o problema.

Figura 5.7 – EBL2 aplicada ao exercício F.



Na Figura 5.7, pode-se verificar que, no item 1, a solução encontrada durante a realização do exercício apresenta passos extras. Além disso, os passos realizados podem indicar que o aluno não tinha certeza a priori de como resolver o problema, dado que inicialmente aplicou regras nas hipóteses, depois em seus resultados e em seguida retornou às hipóteses. Com o objetivo de mostrar uma organização de passos diferentes, o EvoLogic apresenta outra solução (item 2), similar a solução obtida. Pode-se verificar que na solução sugerida, as partes que compõem o último passo (passos 6 e 9) são obtidas diretamente, sem a realização de passos alternados entre um objetivo e outro. Cabe, ainda, considerar a geração de uma explicação das diferenças das duas soluções, geradas automaticamente, para auxiliar o aluno em seu entendimento.

5.4 Considerações

Além da análise acerca das soluções obtidas (seção 5.1) e do comportamento do *model tracing*, faz-se necessário analisar as vantagens e desvantagens do EvoLogic frente a outro STI que suporta o ensino de Lógica (ambiente Heráclito). Essa comparação considera o Heráclito como linha base, dado suas características e vantagens frente aos demais provadores de teorema existentes e disponíveis ao público. Para realizar essa análise traça-se um comparativo acerca das soluções e da forma de acompanhar o raciocínio do aluno.

Tabela 5-16 – Comparação entre o EvoLogic e o ambiente Heráclito.

Característica	EvoLogic	Heráclito
Soluções para os Problemas Analisados		
1. Obtenção das Soluções	A priori	A cada passo
2. Tempo de processamento por exercício	Aprox. 3m e 30s	Em média 1,1s por passo
3. Tempo total de processamento	**Aprox. 34m**	Aprox. 55m (3019 passos realizados em 10 exercícios)
Acompanhamento automático dos passos dos alunos (*model tracing*)		
4. Identificação de passos extras assim que aplicados	93%*	**100%**
5. Linhas de raciocínio identificadas	Todas obtidas a priori	1 linha seguida pelo aluno
6. Identificação de múltiplas linhas de raciocínio	Sim	Não
Necessidade de armazenamento		
7. Armazenamento	**Aprox. 20MB**	Aprox. 55MB

Considerando a primeira característica comparada entre as duas ferramentas, a obtenção de soluções faz com que o comportamento geral dos sistemas seja completamente diferente. No EvoLogic, quando um novo exercício é iniciado, um processo evolutivo, baseado em AGs, é iniciado e o máximo de soluções possíveis são extraídas. Esse processo leva em torno de 3 minutos e meio onde obteve-se 100% das soluções estimadas para 9 dos 10 exercícios estudados. Considerando que cada exercício foi resolvido apenas uma única vez, necessitou-se de aproximadamente 34 minutos de processamento para se obter as soluções. Nesse quesito, o ambiente Heráclito continua a prova do teorema a partir do estado atual da prova que o aluno está realizando, identificando assim qual a característica do passo do aluno. Esse processo, no mesmo hardware utilizado pelo EvoLogic, necessita de pouco mais de 1 segundo por passo realizado. Ao considerar que foram realizados 3019 passos totais, seguindo as diferentes linhas de raciocínio e soluções com tamanhos diferentes, bem como os passos extras, foram necessários, aproximadamente 50 minutos todos de processamento durante o experimento. Uma informação adicional importante para essa comparação é que, em alguns casos, no EvoLogic, os passos iniciais levaram até 3 minutos para serem categorizados, sendo que o passo extra que mais demorou para ser identificado levou 31 segundos (o EvoLogic ainda estava obtendo a solução para o exercício quando o passo ocorreu). Após as soluções terem sido obtidas, os passos foram categorizados em menos de 1 segundo. No que tange o ambiente Heráclito, alguns passos levaram até 10 segundos para serem categorizados, devido a demanda de processamento em paralelo, uma vez que o teorema precisa ser provado a cada passo de cada aluno. Nesse contexto, vale mencionar que os sistemas possuem vantagens e desvantagens. Enquanto que o EvoLogic consegue categorizar um passo em menos de 1 segundo ele necessita que as soluções sejam obtidas a priori, de modo que fica aguardando enquanto as soluções não forem conhecidas. Por outro lado, o ambiente Heráclito necessita de, em média, 1,1 segundos para categorizar os passos, variando para mais ou para menos conforme a demanda de processamento.

Devido às diferenças na forma de obtenção de soluções, o *model tracing* dos dois sistemas também possuem características distintas. Conforme visto na Tabela 5.16, o EvoLogic não identificou todos os passos extras assim que eles ocorreram, uma vez que ele ainda não possuía as soluções para os exercícios recém iniciados. Nesse contexto, o Heráclito se destaca por conseguir fornecer essa informação logo após o passo ser realizado. Ao considerar o tempo de atualização da tela ao aplicar uma regra na interface do ambiente Heráclito e o tempo médio que os alunos levam para aplicar as regras, mesmo os passos que demoraram 10 segundos para serem identificados, eles parecem terem sido identificados em tempo de execução. Contudo, a grande diferença aparece na forma em que as linhas de raciocínio são identificadas. Enquanto que o EvoLogic categoriza os passos dos alunos conforme uma ou mais linhas de raciocínio, o Heráclito considera apenas 1 linha, obtida a partir da prova atual do aluno (que está sendo realizada). Essa característica pode sugerir que os alunos sigam uma linha de raciocínio diferente da que estão seguindo, como no caso do exercício I. Ao informar que os necessitavam de apenas mais um passo para concluir a prova do teorema, muitos acabaram buscando esse passo em vez de seguir sua linha de raciocínio.

Outro ponto importante a ser considerado é a capacidade de memória de armazenamento requerida nesse processo, uma vez que a forma e as informações armazenadas nos dois sistemas são diferentes. No ambiente Heráclito, a cada passo do aluno, toda sua prova parcial é armazenada, ou seja, tudo o que foi feito até o passo realizado no momento, gerando grandes quantidades de dados redundantes. No EvoLogic, quanto aos passos dos alunos, registra-se apenas as mudanças que ocorreram após a chegada da mensagem, ou seja, passos novos, passos desfeitos, entre outros. Contudo, o EvoLogic também necessita armazenar todas as soluções obtidas pelo agente Especialista para que possam ser utilizadas pelo *model tracing*. Para lidar com essa diferença de informações, o EvoLogic é integrado ao Sistema Gerenciador de Banco de Dados MySQL e utiliza um banco de dados relacional para modelar as informações armazenadas (Figura 3.7). De forma resumida, considerando essas diferenças, ao final do experimento o Heráclito necessitou de 55MB de memória para armazenar os dados gerados, enquanto que o EvoLogic utilizou apenas 20MB.

Por fim, para analisar as estratégias propostas, desenvolveu-se uma forma simples de interação com o EvoLogic, onde foi possível analisar três estratégias na prática. Verificou-se que o EvoLogic possui os subsídios para a utilização dessas estratégias e que elas apresentam o comportamento esperado. Contudo, é possível que elas sejam aprimoradas, apresentando exemplos mais condizentes com o uso esperado da regra, explicações sobre diferentes soluções e novas regras existentes nesses exemplos, bem como sobre diferentes formas de abordar o mesmo exercício.

6. CONCLUSÕES

O presente trabalho apresenta o Sistema Tutor Inteligente para ensino de Lógica, EvoLogic, composto por três agentes, que identifica as diferentes linhas de raciocínio dos alunos ao longo dos exercícios, proporcionando que seja oferecido um *feedback* alinhado com os objetivos do aluno. Para que o EvoLogic atinja seus objetivos, o STI é projetado como um sistema multiagente, sendo suportado por estruturas cognitivas existentes nas principais arquiteturas cognitivas aqui apresentadas.

Para identificar as linhas de raciocínio, o EvoLogic conta com um agente Especialista, baseado em um Algoritmo Genético, que resolve os exercícios iniciados pelos alunos. Essas soluções são utilizadas pelo agente Pedagógico para compor o *model tracing*, mecanismo que acompanha o raciocínio dos alunos. Além desses dois agentes, o EvoLogic conta com um agente Interface, responsável pela comunicação entre o STI e a interface. Vale ressaltar que o EvoLogic pode ser acoplado em diversas interfaces através da troca de mensagens predefinidas.

Para avaliar a capacidade do Especialista em resolver diferentes problemas (previamente desconhecidos) e do *model tracing* em acompanhar os alunos, utilizou-se um ambiente simulado, onde foi reproduzido um experimento conduzido previamente no ambiente Heráclito (Galafassi, 2019b).

Nesse experimento, para avaliar as soluções obtidas, foram analisados diferentes critérios de parada em relação à quantidade de soluções diferentes obtidas e ao tempo de processamento. Ao comparar o número máximo de gerações populacionais, verificou-se que o tempo de processamento aumenta de forma exponencial conforme o critério de parada aumenta. Além disso, também foi identificado que um critério de parada restritivo (500 gerações populacionais) obteve poucas soluções. Desse modo, definiu-se que o critério de parada ideal, considerando o universo de experimentos, seria de 1000 gerações populacionais, obtendo todas as soluções conhecidas para 9 dos 10 exercícios estudados, necessitando de, em média, 3 minutos e 30 segundos para cada exercício.

Além disso, ao avaliar o *model tracing*, pode ser observado que alguns alunos realizaram passos extras antes do agente Especialista ter resolvido o problema. Isso ocorre quando o primeiro aluno inicia a resolução do exercício, acionando o mecanismo que obtém as soluções para o problema. Nesse sentido, verificou-se que os primeiros exercícios, que foram iniciados pela maioria dos alunos, tiveram maior ocorrência desses casos.

Ao detalhar os passos que os alunos realizaram em determinados exercícios, verificou-se um cenário atípico no exercício I. Esse exercício possui dois conjuntos de soluções distintas, sendo que em um caso são necessários 8 passos e em outro apenas 3. Para se obter qualquer uma das soluções se faz necessário realizar um passo em comum, de modo que seria necessário realizar apenas mais um passo para obter a solução mais curta. Ao identificar que o exercício pode ser finalizado com apenas mais um passo, o Heráclito envia uma mensagem motivacional, informando o aluno que o final do exercício está próximo. Essa informação pode gerar confusão no aluno, uma vez que ele pode estar seguindo uma linha de raciocínio mais longa e que demandaria mais passos. Vale ressaltar que ambas as soluções são válidas e estão corretas para o exercício. No exercício em questão, o EvoLogic identificou que existiriam duas linhas de raciocínio diferentes, permitindo que as estratégias pedagógicas possam ser mais precisas.

Outro ponto importante refere-se à necessidade de espaço de armazenamento. Além de armazenar os passos dos alunos, o EvoLogic também armazena diferentes soluções para o mesmo problema. Para lidar com essa questão, além das diferentes soluções para cada

problema, o EvoLogic armazena apenas as mudanças ocorridas entre um passo e outro. Essa mudança sutil permite que cada solução completa seja armazenada apenas uma vez, enquanto que o Heráclito armazena a prova parcial (a cada passo novo que o aluno realiza) e a solução final apresentada de cada aluno. Desse modo, no universo de experimentos realizados, o EvoLogic reduziu a necessidade de armazenamento em aproximadamente 63%.

Além disso, ainda é apresentado um conjunto de estratégias de aprendizagem baseadas em exemplos. Essas estratégias se utilizam da capacidade do agente Especialista em gerar diversas soluções a priori para o mesmo problema. Dentre as três estratégias analisadas, pode-se verificar que o EvoLogic provê o subsídio necessário para que elas sejam aplicadas e ainda possibilita que elas sem aprimoradas. Essas lacunas precisam ser endereçadas para que o processo de mediação pedagógico possa ser realizado adequadamente, em especial, contando com uma interface que permita uma interação amigável com o usuário.

O EvoLogic diferencia de muitos STIs por possuir um Especialista baseado em Algoritmo Genéticos, o qual permite a resolução de problemas baseados em análise combinatória bem como em estrutura de passos (como no caso da DNLP). Além disso, o AG proposto se diferencia dos métodos clássicos ao incluir um passo que caracteriza o problema de DNLP, podendo ser alterado para lidar com outros tipos de problemas. Aliado a isso, o *model tracing* automatizado permite que o professor tenha mais autonomia no planejamento de atividades, sem que seja preciso delinear o processo cognitivo para cada novo exercício proposto. Por fim, devido as características do problema e do Especialista, diversas soluções podem ser obtidas para o mesmo problema, permitindo que estratégias de aprendizagem baseadas em exemplos possam ser exploradas e aliadas a *feedback* especializados.

Sendo assim, como trabalhos futuros, sugere-se um estudo aprofundado acerca do processo de mediação pedagógico que pode ser implementado utilizando a teoria de ensino baseado em exemplo, bem como um mecanismo de sugestão de novos exercícios baseados nos diferentes exemplos de soluções que podem ser obtidos.

REFERÊNCIAS

Albus, J.; Bostelman, R.; Hong, T.; Chang, T.; Shackleford, W.; Shneier, M. THE LAGR PROJECT integrating learning into the 4D/RCS control hierarchy. In: **International conference in control, automation and robotics**, 2006.

Aleven, V. Rule-Based Cognitive Modeling for Intelligent Tutoring Systems. In **R. Nkambou; et al. (eds.) Advances in Intelligent Tutoring Systems**. SCI 308. p. 33–62, 2010.

Anderson, J. R. **The Architecture of Cognition.** Harvard University Press, 1983.

Anderson, J. R.; Conrad, F. G.; Corbett, A. T. Skill acquisition and the LISP tutor. **Cognitive Science**, v. 13, n. 4, p. 467-505, 1989.

Anderson, J. R. **Rules of the Mind.** Lawrence Erlbaum Associates, 1993.

Anderson, J. R.; Corbett, A. T.; Koedinger, K. R.; Pelletier, R. Cognitive tutor: Lesson learned. **The Journal of the Learning Sciences**. v. 4, n. 2, p. 167–207, 1995.

Anderson, J. R.; Reder, L. M; Lebiere, C. Working memory: activation limitations on retrieval. **Cognitive Psychology**, v. 30, p. 221–256, 1996.

Angwin, J.; Larson, J.; Mattu, S.; Kirchner, L. Machine Bias. PROPUBLICA, 2016. Disponível em: https://www. propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing. Acessado: 09/03/2020.

Atkinson, R. K., Derry, S. J., Renkl, A. Wortham D. Learning from examples: instructional principles from the worked examples research. **Review of Educational Research**. v. 70, n. 2, p. 181–214, 2000.

Arrabales, R.; Ledezma, A., Sanchis, A. A cognitive approach to multimodal attention. **Journal of Physical Agents**, v. 3, n. 1, p. 53–63, 2009.

ATKINSON, R. C.; SHIFFRIN, R. M. Human memory: a proposed system and its control processes. **Psychology of learning and motivation. Advances in Research and Theory**, n. 2, v. 1, p. 89–195, 1968.

Bach, J. Modeling motivation in MicroPsi 2. In: **International conference on artificial general intelligence**, p. 3–13, 2015.

Bach, J.; Bauer, C.; Vuine, R. MicroPsi: contributions to a broad architecture of cognition. In: **Annual conference on artificial intelligence**, p. 7–18, 2007.

BÄCK, T.; FOGEL, D. B.; MICHALEWICZ, Z. **Evolutionary Computation 1: Basic Algorithms and Operators**. Institute of Physics Publishing, 2000.

BADDELEY, A. D.; HITCH, G. Working memory. **Psychology of learning and motivation. Advances in Research and Theory**, n. 8, v. C, p. 47–89, 1974.

Bandera, A.; Bustos, P. Toward the development of cognitive robots. In: **International workshop on brain-inspired computing**, 2013.

Bandura A. **Social Learning Theory**. Englewood Cliffs, N.J: Prentice Hall; 1977.

Bellas, F.; Duro, R. J.; Faiña, A.; Souto, D. Multilevel Darwinist brain (MDB): artificial evolution in a cognitive architecture for real robots. **IEEE transactions on autonomous mental development**, v. 2, n. 4, p. 340–354, 2010.

Bellman, R. **An Introduction to Artificial Intelligence: Can Computers Think?** San Francisco: Boyd & Fraser, 1978.

Blessing, S.; Gilbert, S.; Ourada, S.; Ritter, S. Authoring Model-Tracing Cognitive Tutors. **Artificial Intelligence in Education**, v. 19, p. 189-210, 2009.

Boicu, C.; Tecuci, G; Boicu, M. Improving agent learning through rule analysis. In: **Proceedings of the international conference on artificial intelligence**, 2005.

Bostelman, R.; Hong, T.; Chang, T.; Shackleford, W.; Shneier, M. Unstructured facility navigation by applying the NIST 4D/RCS architecture. In: **Proceedings of international conference on cybernetics and information technologies, systems and applications**, p. 328–333, 2006.

Brasil. Lei Federal 13.005, de 25 de junho de 2014. Aprova o Plano Nacional de Educação - PNE e dá outras providências. Brasília, DF, 25. Jun. 2014. Disponível em: http://www.planalto.gov.br/ccivil_03/_ato2011-2014/2014/lei/l13005.htm. Acessado em: 17/04/2020.

Breazeal, C.; Aryananda, L. Recognition of affective communicative intent in robot-directed speech. **Autonomous Robots**, v. 12, n. 1, p. 83–104, 2002.

Breazeal, C.; Brooks, R. Robot emotion: a functional perspective. In: **Fellous, J. M.; Arbib, M. A. Who needs emotions? The brain meets the robot**. Oxford University Press, Oxford, p. 271–310, 2004.

BREMERMANN, H. J. **Optimization through evolution and recombination Self-Organizing Systems**. ed. M. C. Yovits et al., Washington, DC: Spartan, 1962.

BREMERMANN, H. J.; Rogson, J.; Salaff, S. Global properties of evolution process. In: **Pattee, H. H. Natural Automata and Useful Simulations**, p.3-42, 1964.

Brett. B, E; Doyal, J. A.; Malek, D. A.; Martin, E. A.; Hoagland, D. G.; Anesgart, M. N. **The combat automation requirements testbed (CART) task 5 interim report: modeling a strike fighter pilot conducting a time critical target mission.** Technical report, 2002.

Brick, T.; Schermerhorn, P.; Scheutz, M. Speech and action: integration of action and language for mobile robots. In: **Proceedings of IEEE international conference on intelligent robots and systems**, p. 1423–1428, 2007.

Bridewell, W.; Bello, P. F. Incremental object perception in an attention-driven cognitive architecture. In: **Proceedings of the 37th annual meeting of the cognitive science society**, p. 279–284, 2015.

BROOKS, R. A. Planning is just a way of avoiding figuring out what to do next. **Technical report working paper 303**, [S. I.], 1987.

BROOKS, R. A. Elephants don't play chess. **Robotics and Autonomous System**, v. 6, n. 1–2, p. 3–15, 1990.

Bustos, P.; Martinez-Gomez, J.; Garcia-Varea, I.; Rodriguez-Ruiz, L.; Bachiller, P.; Calderita, L.; Manso, L. J.; Sanchez, A.; Bandera, A.; Bandera, J. P. Multimodal interaction with Loki. In: **workshop of physical agents**, 2013.

Cao, S.; Qin, Y.; Zhao, L.; Shen, M. Modeling the development of vehicle lateral control skills in a cognitive architecture. **Transportation Research Part F: Traffic Psychology**, v. 32, p. 1–10, 2015.

CARPENTER, G. A.; GROSSBERG, S.; REYNOLDS, J. H. ARTMAP: supervised real-time learning and classification of nonstationary data by a self-organizing neural network. **Neural Networks**, v. 4, n. 5, p. 565–588, 1991.

Charniak, E.; McDermott, D. **Introduction to Artificial Intelligence.** Addison-Wesley Publ, 1985.

Coq The Coq Proof Assistant. Disponível em: https://coq.inria.fr/ Acessado em: 09/03/2020

Corbett, A.; Cognitive Tutor Algebra I: Adaptive Student Modeling in Widespread Classroom Use. In **Proceedings: Technology and Assessment: Thinking Ahead**, p. 50-62, 2002.

Corbett, A. T.; Anderson, J. R. Knowledge tracing: Modeling the acquisition of procedural knowledge. **User Modeling and User-Adapted Interaction**, v. 4, p. 253-278, 1995.

Corbett, A. T.; Anderson, J. R. Feedback control and learning to program with CMU LISP Tutor. **Annual Meeting of the American Educational Research Association**, Chicago, IL, 1991.

Corbett, A.; Kauffman, L.; MacLaren, B.; Wagner, A.; Jones, E. A Cognitive Tutor for genetics problem solving: Learning gains and student modeling. **Journal of Educational Computing Research**, v. 42, n. 2, p. 219–239, 2010.

Corbett, A.T.; Koedinger, K.R.; Anderson, J.R. Intelligent Tutoring Systems. In **Helander, T. K.; Landauer, P. Handbook of Human-Computer Interaction.** Amsterdam: Elsevier Science, 1997.

Cox, M. T.; Oates, T.; Paisner, M.; Perlis, D. Noting anomalies in streams of symbolic predicates using A-distance. **Advances in Cognition Systems**, v. 2, p. 167–184, 2012.

Dannenhauer, D.; Cox, M. T.; Gupta, S.; Paisner, M.; Perlis, D. Toward meta-level control of autonomous agents. In: **Proceedings of the 5th annual international conference on biologically inspired cognitive architectures**, v. 41, p. 226–232, 2014.

Davis, L. **Handbook of Genetic Algorithms**. Van Nostrand Reinhold, New York, 1991.

De Jong, K. A. **Genetic Algorithms in Search, Optimization, and Machine Learning**. Addison-Wesly, Reading, Massachusetts, 1975.

Deutsch, S. E.; Macmillan, J.; Camer, N. L; Chopra, S. **Operability model architecture: demonstration final report.** Technical report AL/HR-TR-1996-0161, 1997.

E-Prover. The E Theorem Prover. Disponível em: https://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html. Acessado em: 09/03/2020

Epstein, S. L.; Passonneau, R.; Gordon, J.; Ligorio, T. The role of knowledge and certainty in understanding for dialogue. In: **AAAI fall symposium: advances in cognitive systems**, 2012.

Fan, X.; McNeese, M.; Sun, B.; Hanratty, T.; Allender, L.; Yen, J. Human-agent collaboration for time-stressed multicontext decision making. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 40, n. 2, p. 306–320, 2010.

FIPA. FIPA Specifications – Repository. 2018. Disponível em: http://www.fica.org/specs/fipa00023/SC00023,K.pdf. Acessado em: 09/03/2020

Firby, J. R.; Kahn, R. E.; Prokopowicz, P. N.; Swain, M. J. An architecture for vision and action. In: **Proceedings of the 14th international joint conference on artificial intelligence**, 1995.

Flavell, J. H. Metacognition and cognitive monitoring: a new area of cognitive-developmental inquiry. **American Psychologist**, n. 34, v. 10, p. 906–911, 1979.

Fogel, L. J. Autonomous automata. **Industrial Res.**, v. 4, p. 14–9, 1962.

Ford, M., **RISE OF THE ROBOTS: TECHNOLOGY AND THE THREAT OF A JOBLESS FUTURE**. Basic Books, MARTIN FORD A Member of the Perseus Books Group, New York XVI, 2015.

Fraser, A. Simulation of genetic systems by automatic digital computers. **Australian journal of biological sciences**. v. 10, n. 4, p. 484–491, 1957.

Galafassi, P. F. F.; Galafassi, C.; Vicari, R. M.; Gluz, J. C. Identifying Knowledge from the Application of Natural Deduction Rules in Propositional Logic. In: Demazeau Y., Matson E., Corchado J., De la Prieta F. (eds) **Advances in Practical Applications of Survivable Agents and Multi-Agent Systems: The PAAMS Collection**. p. 66-77, 2019a.

Galafassi, F. F. P. **Identificando conhecimentos a partir de aplicações de regras de dedução natural na lógica proposicional, um estudo prático.** Tese. Programa de pós-graduação em Informática na Educação. Centro Interdisciplinar de Novas Tecnologias na Educação. Universidade Federal do Rio Grande do Sul. 262 páginas, 2019b.

GARDNER, H. **A nova ciência da mente.** São Paulo: Editora da Universidade de São Paulo, 1996.

GAT, E. Integrating planning and reacting in a heterogeneous asynchronous architecture for controlling real-world mobile robots. In: **Proceedings of the annual AAAI Conference on Artificial Intelligence**, p. 809–815, 1992.

GEIGER, M. Genetic Algorithms for multiple objective vehicle routing. **Proceedings of the Metaheuristics International Conference**, p.349-353, 2008.

GEORGEFF, M. P.; LANSKY, A. L. Procedural knowledge. **Proceedings of the IEEE**, n. 74, p. 1383–1398, 1986.

GIRARDI, R. Engenharia de Software Baseada em Agentes. **IV Congresso Brasileiro de Computação**, n. 4, p. 25, 2004.

GLUZ, J.C.; GALAFASSI, F. F. P.; MOSSMANN, M.; VICARI, R. M. Heráclito: a Dialectical Tutor for Logic. In: **Portuguese Conference on Artificial Intelligence, EPIA**, v. 8154, p. 1-2, 2013.

GLUZ, J.C.; PY, M. X. **Introdução À Lógica Proposicional.** Grupo TAOS3, PIPCA, UNISINOS, 2015.

Goertzel, B.; Pennachin, C.; Souza, S. D. An inferential dynamics approach to personality and emotion driven behavior determination for virtual animals. In: **AISB 2008 convention on communication, interaction and social intelligence**, 2008.

Goertzel, B.; Sanders, T.; O'Neill, J. Integrating deep learning based perception with probabilistic logic via frequent pattern mining. In: **International conference on artificial general intelligence**, 2013.

van Gog, T, Rummel N. Example-based learning: Integrating cognitive and social-cognitive research perspectives. **Educational Psychology Review**. v. 22, n. 2, p. 155–74, 2010.

GOLDBERG, D. E.; LINGLE, R. Alleles, loci and the traveling salesman problem. In: **Grefenstette, J. J. Proceedings of an International Conference on Genetic Algorithms and Their Applications**. Lawrence Erlbaum Associates, Hillsdale, New Jersey, p. 154-159, 1985.

Gore, B. F.; Hooey, B. L.; Wickens, C. D.; Scott-Nash, S. A computational implementation of a human attention guiding mechanism in MIDAS v5. In: **International conference on digital human modeling**, 2009.

Große, C. S., Renkl, A. Finding and Fixing Errors in Worked Examples: Can this foster learning outcomes? **Learning and Instruction**. v. 17, pp.612-634, 2007.

Haugeland, J. **Artificial Intelligence: The Very Idea.** Cambridge, MA, USA: Massachusetts Institute of Technology, 1985.

Hayes-Roth, B.; Lalanda, P.; Morignot, P.; Pfleger, K.; Balabanovic, M. Plans and behavior in intelligent agents. **KSL report**, p. 93-43, 1993.

Henderson, T. C.; Joshi, A. The cognitive symmetry engine. **Technical report UUCS-13-004**, 2013.

Hol. Interactive Theorem Prover. Disponível em: https://hol-theorem-prover.org/ Acessado em: 09/03/2020

HOLLAND, J. H. **Adaptation in Natural and Artificial Systems**. University of Michigan Press, Ann Arbor, Michigan, 1975.

Houghton, R. S. Rationale for Multimedia Use and Instruction in Education, v7.81. Western Carolina University. Disponível em: http://www.wcu.edu/ceap/HOUGHTON/MM/rationale/rationaleMM.html Acessado em: 09/03/2020

HU, X. B.; DI PAOLO, E. An Efficient Genetic Algorithm with Uniform Crossover for the Multi-Objective Airport Gate Assignment Problem. **Evolutionary Computation**, p.55-62, 2007.

Huang, X.; Weng, J. Inherent value systems for autonomous mental development. **International Journal of Humanoid Robotics**, v. 4, n. 2, p. 407–433, 2007.

Hudlicka, E. Beyond cognition: modeling emotion in cognitive architectures. In: **Proceedings of the sixth international conference on cognitive modeling**, p. 118–123, 2004.

Hudlicka, E. Modeling cultural and personality biases in decision making. In: **Proceedings of the 3rd international conference on applied human factors and ergonomics (AHFE)**, 2010.

Ikle, M.; Goertzel, B. Nonlinear-dynamical attention allocation via information geometry. In: **International conference on artificial general intelligence**, 2011.

Isabelle. Isabelle proof assistant. Disponível em: https://isabelle.in.tum.de/. Acessado em: 09/03/2020

Ivaldi, S.; Nguyen, S. M.; Lyubova, N.; Droniou, A.; Padois, V.; Filliat, D.; Oudeyer, P. Y.; Sigaud, O. Object learning through active exploration. **IEEE transactions on autonomous mental development**, v. 6, n. 1, p. 56–72, 2014.

JAPE. Disponível em: http://japeforall.org.uk/ Acessado em: 09/03/2020

Khaleghi, B.; Khamis, A.; Karray, F. O. Multisensor data fusion: a review of the state-of-the-art. **International Journal on Multi-Sensor, Multi-Source Information Fusion**, v. 14, n. 1, p. 28–44, 2013.

Kieras, D. E.; Wakefield, G. H.; Thompson, E. R.; Iyer, N.; Simpson, B. D. Modeling two-channel speech processing with the EPIC cognitive architecture. **Topics in Cognitive Science**, v. 8, n. 1, p. 291–304, 2016.

Kochen, M. How clinicians recall experiences. **Methods of Information in Medicine**. v. 22, n. 2, p. 83–6, 1983.

Koedinger, K. R.; Anderson, J. R. Abstract planning and perceptual chunks: Elements of expertise in geometry. **Cognitive Science**, v. 14, p. 511-550, 1990.

Koedinger, K. R.; Anderson, J. R. Effective use of intelligent software in high school math classrooms. **Artificial Intelligence in Education: Proceedings of the World Conference on AI in Education**, AACE: Charlottsville, VA, 1993.

Koedinger, K.R.; Corbett, A.T.; Sawyer, R.K. **Cognitive tutors: Technology bringing learning sciences to the classroom**. The Cambridge handbook of the learning sciences. New York: Cambridge University Press, 2006.

Kolodner, JL. Educational implications of analogy. A view from case-based reasoning. **American Psychologist**. v. 52, n. 1, p. 57–66, 1997.

Kotseruba, I., Tsotsos, J. K. 40 years of cognitive architectures: core cognitive abilities and practical applications. **Artificial Intelligence review**, v. 53, p. 17-94, 2020.

KUOKKA, D. R. Integrating planning, execution, and learning. In: **Proceedings of the NASA conference on space telerobotics**, p. 377–386, 1989.

Kurzweil, R. **The Age of Intelligent Machines**. Cambridge, MA, USA: MIT Press, 1990.

Laird, J. E. The soar cognitive architecture. **AISB/IACAP World Congress**, v. 171, n. 134, p. 224–235, 2012.

Laird, J. E.; Mohan, S. A case study of knowledge integration across multiple memories in Soar. **Biologically Inspired Cognitive Architectures**, v. 8, p. 93–99, 2014.

Lavin, A.; Ahmad, S. Evaluating real-time anomaly detection algorithms—the numenta anomaly benchmark. In: **Proceedings of the 14th international conference on machine learning and applications (ICMLA)**, 2015.

Lebiere, C.; Pirolli, P.; Thomson, R.; Paik, J.; Rutledge-Taylor, M.; Staszewski, J.; Anderson, J. R. A functional model of sensemaking in a neurocognitive architecture. **Computational Intelligence and Neuroscience**, v. 5, 2013.

Lesta L., Yacef K. An Intelligent Teaching Assistant System for Logic. In: Cerri S.A., Gouardères G., Paraguaçu F. (eds) Intelligent Tutoring Systems. ITS 2002. **Lecture Notes in Computer Science**, v. 2363. p. 421-431, 2002.

Lindes, P.; Laird, J. E. Toward integrating cognitive linguistics and cognitive language processing. In: **Proceedings of international conference on cognitive modeling**, 2016.

Lukins, S.; Levicki, A. and Burg, J. A Tutorial Program for Propositional Logic with Human/Computer Interactive Learning. **Proceedings of the 33rd SIGCSE Technical Symposium on Computer Science Education**. p. 381-385, 2002.

Madl, T.; Franklin, S.; Chen, K.; Montaldi, D.; Trappl, R. Towards real-world capable spatial memory in the LIDA cognitive architecture. **Biologically Inspired Cognitive Architectures**, v. 16, p.87–104, 2015.

Manso, L. J.; Calderita, L. V.; Bustos, P.; Garcia, J.; Martinez, M.; Fernandez, F.; Romero-Garces, A.; Bandera, A. A general-purpose architecture to control mobile robots. In: **XV workshop of physical agents: book of proceedings (WAF 2014)**, 2014.

Marr, D. **Vision: a computational investigation into the human representation and processing of visual information.** MIT Press, Cambridge, 2010.

Martin, D.; Rincon, M.; Garcia-Alegre, M. C.; Guinea, D. ARDIS: knowledge-based architecture for visual system configuration in dynamic surface inspection. **Expert Systems**, v. 28, n. 4, p. 353–374, 2011.

McCARTHY, J. What is artificial intelligence. 2017. Disponível em: http://www-formal.stanford.edu/jmc/whatisai/. Acessado em: 09/03/2020.

Murray, T. An Overview of Intelligent Tutoring System Authoring Tools: Updated Analysis of the State of the Art. In **Authoring Tools for Advanced Technology Learning Environments**. Springer, 491–544, 2003.

Ng, G. W.; Xiao, X.; Chan, R. Z.; Tan, Y. S. Scene understanding using DSO cognitive architecture. In: **Proceedings of the 15th international conference on information fusion (FUSION)**, p. 2277–2284, 2012.

Nilsson, N. **Artificial Intelligence: A New Synthesis.** Elsevier Science, 1998.

Novianto, R.; Johnston, B.; Williams, M. A. Habituation and sensitisation learning in ASMO cognitive architecture. **Lecture Notes in Computer Science**, v. 8239, p. 249–259, 2013.

NWANA, H. S. **Intelligent Tutoring Systems: an overview**. Artificial Intelligence Review. Springer, 1990.

O'Reilly, R. C.; Wyatte, D.; Herd, S.; Mingus, B.; Jilk, D. J. Recurrent processing during object recognition. **Frontiers in Psychology**, v. 4, p. 124, 2013.

Ozturk, P. Levels and types of action selection: the action selection soup. **Adaptive Behavior**, v. 17, p. 537–554, 2009.

Paisner, M.; Cox, M. T.; Maynord, M.; Perlis, D. Goal-driven autonomy for cognitive systems. In: **Proceedings of the 36th annual conference of the cognitive science society**, p 2085–2090, 2013.

Pandora. Proof Assistant for Natural Deduction using Organised Rectangular Areas. http://www.doc.ic.ac.uk/pandora/newpandora/quick_start.html. Acessado em: 09/03/2020.

Pattacini, U.; Nori, F.; Natale, L.; Metta, G.; Sandini, G. An experimental evaluation of a novel minimum-jerk Cartesian controller for humanoid robots. In: **Proceedings of the international conference on intelligent robots and systems (IROS)**, p. 1668–1674, 2010.

Paviotti, G., Rossi, P. G., Zarka, D. **Intelligent tutoring systems: an overview**. Pensa Multimedia, 2012.

Perner, A.; Zeilinger, H. Action primitives for bionics inspired action planning system: abstraction layers for action planning based on psychoanalytical concepts. In: **IEEE international conference on industrial informatics (INDIN)**, p. 63–68, 2011.

Peters, R. A.; Kawamura, K.; Wilkes, D. M.; Hambuchen, K. A.; Rogers, T. E.; Alford, W. A. ISAC humanoid: an architecture for learning and emotion. In: **Proceedings of the IEEE-RAS international conference on humanoid robots**, v. 1, p. 459, 2001.

Pollock, J. L. OSCAR: an agent architecture based on defeasible reasoning. In: **AAAI spring symposium: emotion, personality, and social behavior**, 2008.

Pollock, J. L. Planning in OSCAR. **Minds and Machines**, v. 2, p. 113–144, 1993.

Poole, D.; MacWorth, A.; Goebel, R. **Computational Intelligence: A Logical Approach**. New York, NY, USA: Oxford University Press, Inc., 1997.

Prover9/Mace4. Prover9 and Mace4 Disponível em: https://www.cs.unm.edu/~mccune/prover9/ Acessado em: 09/03/2020

Rankin, C. H.; Abrams, T.; Barry, R. J.; Bhatnagar, S.; Clayton, D.; Colombo, J.; Coppola, G.; Geyer, M. A.; Glanzman, D. L.; Marsland, S.; Mcsweeney, F.; Wilson, D. A.; Wu, C. F.; Thompson, R. F. Habituation revisited: an updated and revised description of the behavioral characteristics of habituation. **Neurobiology of Learning and Memory**, v. 92, n. 2, p. 135–138, 2009.

RECHENBERG, I. **Cybernetic Solution Path of an Experimental Problem**. Royal Aircraft Establishment Library, 1965.

REEVES, C. Genetic Algorithms. **In: Glover, F., Kochenberger, G. A., Handbook of Metaheuristics**, Kluwer Academic Publishers, 2003.

Renkl, Alexander. **Instruction based on examples**. Handbook of research on learning and instruction. 272-295, 2011.

Renkl, A. Toward an instructionally oriented theory of example-based learning. **Cognitive Science**. v38, n. 1, p. 1–37, 2014.

Rich, E.; Knigth, K. **Inteligência artificial**. 2. Ed. Rio de Janeiro: McGraw-Hill, 1994.

Ritter, F. E. Two cognitive modeling frontiers. Emotions and usability. **Information and Media Technologies**, v. 4, n. 1, p. 76–84, 2009.

Rohrer, B. BECCA version 0.4.5. User's Guide, 2013. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.413.8777&rep=rep1&type=pdf. Acessado em: 09/03/2020

Romero-Garcés, A.; Calderita, L. V.; Martinez-Gomez, J.; Bandera, J. P.; Marfil, R.; Manso, L. J.; Bustos, P.; Bandera, A. The cognitive architecture of a robotic salesman. In: **Conference of the Spanish association for artificial intelligence**, v. 15, n. 6, 2015.

Ruesch, J.; Lopes, M.; Bernardino, A.; Hornstein, J.; Santos-Victor, J.; Pfeifer, R. Multimodal saliency-based bottom-up attention a framework for the humanoid robot iCub. In: **Proceedings of the IEEE international conference on robotics and automation**, p. 962–967, 2008.

Rumbaugh, J.; Jacobson, I.; Booch, G. **The Unified Modeling Language reference manual.** Addison-Wesley Longman, UK, 1998.

RUSSEL, J. S.; NORVIG, P. **Artificial Intelligence: A Modern Approach.** Third Edition. Editora Prentice Hall (Pearson Education), 2010.

Schofield, J. W.; Evan-Rhodes, D. Artificial Intelligence In the classroom: The impact of a computer-based tutor in teachers and students. In: **Proceedings of the 4th International Conference on AI and Education**, p. 238-243, 1989.

SCHWEFEL, H-P. **Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik.** Thesis, Technical University of Berlin, Hermann Fottinger Institute for Hydrodynamics, 1965.

SELF, J. A. Bypassing the intractable problem of student modelling. **Intelligent tutoring systems: At the crossroads of artificial intelligence and education**, v. 41, p. 1-26, 1990.

Seth, A. K.; McKinstry, J. L.; Edelman, G. M.; Krichmar, J. L. Visual binding through reentrant connectivity and dynamic synchronization in a brain-based device. Cerebral cortex, v. 14, n. 11, p. 1185–1199, 2004.

Sieg, W. The AProS project: Strategic thinking & computational logic. **Logic Journal of IGPL**, v. 15, n. 4, p. 359-368, 2007.

Singley, M. K.; Anderson, J. R.; Gevins, J. S.; Hoffman, D. The algebra word problem tutor. **Artificial Intelligence and Education**, p. 267-275, 1989.

Slam, N.; Wang, W.; Xue, G.; Wang, P. A framework with reasoning capabilities for crisis response decision support systems. **Engineering Applications of Artificial Intelligence**, v. 46, p. 346–353, 2015.

SPASS Classic SPASS: An Automated Theorem Prover for First-Order Logic with Equality. Disponível em: https://www.mpi-inf.mpg.de/departments/automation-of-logic/software/spass-workbench/classic-spass-theorem-prover/. Acessado em: 09/03/2020

Squire LR Declarative and nondeclarative memory: multiple brain systems supporting learning. **Journal of Cognitive Neuroscience**, v. 4, n. 3, p. 232–243, 1992.

Starzyk, J. A.; Graham, J. T. MLECOG: motivated learning embodied cognitive architecture. **IEEE Systems Journal**, v. 11, n. 3, p. 1272–1283, 2015.

Stewart, T. C.; Eliasmith, C. Parsing sequentially presented commands in a large-scale biologically realistic brain model. In: **Proceedings of the 35th annual conference of the cognitive science society**, p. 3460–3467, 2013.

Stokes, D.; Biggs, S. The dominance of the visual. In: **Stokes, D.; Matthen, M.; Biggs, S. Perception and its modalities**. Oxford University Press, Oxford, p. 1–35, 2014.

Sun, R. **Anatomy of the mind: exploring psychological mechanisms and processes with the clarion cognitive architecture.** Oxford University Press, Oxford, 2016.

Sun, R. Memory systems within a cognitive architecture. **New Ideas in Psychology**, v. 30, n. 2, p. 227–240, 2012.

Sun, R.; Peterson, T.; Merrill, E. A hybrid architecture for situated learning of reactive sequential decision making. **Applied Intelligence**, v. 11, p. 109–127, 1999.

Tecuci, G.; Boicu, M.; Bowman, M.; Marcu, D.; Shyr, P.; Cascaval, C. An experiment in agent teaching by subject matter experts. **International Journal of Human-Computer Studies**, v. 53, n. 4, p. 583–610, 2000.

TEIXEIRA, J. F. **Mentes e máquinas: uma introdução à ciência cognitiva.** Porto Alegre: Artes Médicas, 1998.

Thorisson, K. R. Mind model for multimodal communicative creatures and humanoids. **Engineering Applications of Artificial Intelligence**, v. 13, n. 4–5, p. 449–486, 1999.

Trafton, J. G.; Cassimatis, N. L.; Bugajska, M. D.; Brock, D. P.; Mintz, F. E.; Schultz, A. C. Enabling effective human robot interaction using perspective-taking in robots. **IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans**, v. 35, n. 4, p. 460–470, 2005.

Trivedi, N.; Langley, P.; Schermerhorn, P.; Scheutz, M. Communicating, interpreting, and executing highlevel instructions for human-robot interaction. In: **Proceedings of AAAI fall symposium: advances in cognitive systems**, 2011.

Tsotsos, J. K. **A computational perspective on visual attention.** MIT Press, Cambridge, 2011.

Turing, A. M. Computing Machinery and Intelligence. **Mind, New Series**, v. 59, n. 236, p. 433-460, 1950.

Tyler, S. W.; Neukom, C.; Logan, M.; Shively, J. The MIDAS human performance model. In: **Proceedings of the human factors and ergonomics society**, p. 320–324, 1998.

VanLehn, K. The behavior of tutoring systems. **International Journal of Artificial Intelligence in Education**, v. 16, n. 3, p. 227–265, 2006.

Varma, S. **A computational model of Tower of Hanoi problem solving.** Tese. Faculty of the Graduate School of Vanderbilt University, 2006.

Veloso, M. M.; Blythe, J. Linkability: examining causal link commitments in partial-order planning. In: **Proceedings of the second international conference on artificial intelligence planning systems**, 1994.

VERE, S.; BICKMORE, T. A basic agent. **Computational Intelligence**, v. 6, n. 1, p. 41–60, 1990.

Wang, P. Natural language processing by reasoning and learning. In: **Proceedings of the international conference on artificial general intelligence**, p. 160–169, 2013.

Weitekamp, D.; Harpstead, E.; Koedinger, K.R. An Interaction Design for Machine Teaching to Develop AI Tutors. In **Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems (CHI'20)**. Association for Computing Machinery, New York, NY, USA, 1–11, 2020.

Weng, J.; Hwang, W. S. From neural networks to the brain: autonomous mental development. **IEEE Computational Intelligence Magazine**, v. 1, n. 3, p. 15–31, 2006.

Wertheimer, R. The Geometry Proof Tutor: An "Intelligent" Computer-based tutor in the classroom. **Mathematics Teacher**, p. 308-313, 1990.

Wilson, J. R.; Forbus, K. D.; McLure, M. D. Am I really scared? A multi-phase computational model of emotions. In: **Proceedings of the second annual conference on advances in cognitive systems**, p. 289–304, 2013.

Winston, P. H. **Artificial Intelligence.** Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

Wong, C.; Kortenkamp, D.; Speich, M. A mobile robot that recognizes people. In: **Proceedings of 7th IEEE international conference on tools with artificial intelligence**, 1995.

WOOLF, B. P. **Building intelligent interactive tutors: student-centered strategies for revolutionizing e-learning.** Department of Computer Science, University of Massachusetts, Amherst. Editora Elsevier, 2009.

Yacef, K. The Logic-ITA in the classroom: a medium scale experiment. **International Journal of Artificial Intelligence in Education**. v. 15, p. 41-62, 2005.

Zachary, W.; Santarelli, T.; Ryder, J.; Stokes, J. **Developing a multi-tasking cognitive agent using the COGNET/iGEN integrative architecture.** Technical report, 2000.

Zachary, W.; Mentec, J. C. L.; Ryder, J. M. Interface agents in complex systems. Human Interaction with Complex Systems, v. 14, n. 1, p. 260–264, 2016.

Zmigrod, S.; Hommel, B. Feature integration across multimodal perception and action: a review. **Multisensory Research**, v. 26, p. 143–157, 2013.

APÊNDICE A

DEDUÇÃO NATURAL NA LÓGICA PROPOSICIONAL (DNLP)

A razão pela qual a lógica é relevante para a representação do conhecimento e o raciocínio é simplesmente porque, pelo menos de acordo com uma visão, a lógica é o estudo das relações de vinculação-linguagens, condições de verdade e regras de inferência. Nesta seção é abordado o conteúdo de Dedução Natural na Lógica Proposicional (DNLP), uma linguagem de representação de conhecimento muito popular, comumente chamada de lógica de primeira ordem (FOL). Essa linguagem foi inventada pelo filósofo Gottlob Frege na virada do século XX para a formalização da inferência matemática, mas foi coadaptada pela comunidade de IA para fins de representação do conhecimento.

Dedução natural é um dos sistemas dedutivos utilizados para construir demonstrações formais na Lógica e uma proposição lógica (ou apenas uma proposição) é uma frase ou sentença da língua portuguesa (ou de qualquer outro idioma) que pode assumir apenas um de dois valores verdade: ou a frase é verdadeira (ela diz uma verdade) ou ela é falsa (diz uma falsidade) [GLUZ e PY, 2010].

Para que ocorra uma derivação formal, é necessário formalizar a expressão que queremos demonstrar. Formalizar significa traduzir da forma linguística usual para uma notação lógica, ou seja, uma forma que é entendível para qualquer um, independente da língua que fala, e que também reduz o espaço ocupado pela frase escrita, tendo em vista que podemos utilizar uma notação mais econômica, a lógica.

Dentre estas notações, as proposições são usualmente simbolizadas (representadas) por letras maiúsculas do início do alfabeto: A, B, C, ... Os valores lógicos das proposições são representados de forma resumida usando V para verdadeiro e F para falso.

As proposições podem ser simples ou compostas. As proposições compostas são formadas de proposições simples conectadas através de operadores (ou conetivos) lógicos. Estes operadores ou conetivos representam as seguintes operações lógicas:

- Conjunção
- Disjunção
- Negação
- Implicação (ou condicional)
- Bi-implicação (ou bicondicional)

Devido à riqueza da língua portuguesa, palavras com significados semelhantes são representadas pelo mesmo operador lógico. A Tabela 1 mostra expressões comuns em português associadas a diversos operadores lógicos.

Tabela 1 – Expressões em português e operadores lógicos

Expressão em português	Operador Lógico	Expressão Lógica
E; mas; também	Conjunção	$A \wedge B$
Ou	Disjunção	$A \vee B$

Se A, então B A implica B A, logo B A somente se B A é condição suficiente para B	Condicional	$A \rightarrow B$
A se e somente se B	Bicondicional	$A \leftrightarrow B$
Não A É falso que A... Não é verdade que A... Não é o caso que A...	Negação	$\neg A$

Fonte: [GLUZ e PY, 2010].

A lógica formal lida com um tipo particular de argumento, denominado de argumento dedutivo, que nos permite deduzir uma conclusão Q, com base num conjunto de proposições P_i, onde $i=\{1, ..., n\}$, onde Q e P_i representam fórmulas inteiras bem-formadas da lógica proposicional.

Um argumento dedutivo pode ser representado de forma simbólica da seguinte forma:

$$P_1 \wedge P_2 \wedge P_3 \wedge \ldots \wedge P_n \rightarrow Q$$

As proposições P_i, onde $i=\{1, ..., n\}$ são denominadas de **hipóteses** ou **premissas** do argumento. A proposição é denominada de **conclusão** do argumento. Em termos de língua natural este tipo de simbolismo pode ser lido como:

"P_1, P_2, ... P_n **acarretam** Q" ou

"Q **decorre** de P_1, P_2, ... P_n" ou

"Q se **deduz** de P_1, P_2, ... P_n" ou ainda

"Q se **infere** de P_1, P_2, ... P_n"

Este método é baseado na aplicação de **regras de dedução** (ou **regras de inferência**) que modificam fórmulas de modo a preservar seu valor lógico.

A.1 Regras de Dedução Natural

Existem dois tipos básicos de regras de dedução:

- Regras de inferência que se baseiam em implicações tautológicas[13], ou seja, implicações materiais provadamente tautológicas.

[13] Onde um argumento é considerado verdadeiro, válido.

- Regras de equivalência que se baseiam nas equivalências tautológicas permitem substituir uma fórmula pela outra, já que ambas são equivalentes.

As regras que são baseadas em implicações que já se tenha demonstrado serem tautológicas serão denominadas de Regras de Inferência. A Tabela 2 apresenta as regras básicas de inferência da lógica proposicional.

Tabela 2 – Regras Básicas de Inferência

Inclusão de Operadores	**Exclusão de Operadores**
Redução ao absurdo (**raa**) - **¬I** | P | … | Q∧¬Q ——— ¬P	Dupla negação (**dn**) - **¬E** ¬¬P ——— P
Prova condicional (**pc**) - **→I** | P | ... | Q ——— P→Q	*Modus Ponens* (**mp**) - **→E** P P→Q ——— Q
Conjunção(**cj**) - **∧I** P Q ——— P∧Q	Simplificação(**sp**) - **∧E** P∧Q P∧Q ——— ——— P Q
Adição(**ad**) - **∨I** P P ——— ——— P∨Q Q∨P	Eliminação da disjunção - **∨E** P∨Q P→R Q→R ——————— R
Introdução da equivalência - **↔ I** P→Q Q→P ——— P↔Q	Eliminação da equivalência - **↔ E** P↔Q P↔Q ——— ——— P→Q Q→P

Fonte: [GLUZ e PY, 2010].

Estas 10 regras são completas no sentido que permitem a manipulação de todos os operadores da lógica proposicional. Do lado esquerdo estão listadas as regras que permitem a inclusão de um dado operador em uma nova fórmula a ser adicionada no decorrer da prova. As regras do lado direito permitem adicionar uma nova prova à demonstração, onde um determinado operador foi eliminado.

Além destas regras básicas, existe também um conjunto de regras derivadas que também podem ser usadas em demonstrações. Embora todas as regras derivadas apresentadas na Tabela 3 possam ser substituídas por demonstrações sem o uso delas, elas podem ser bastante úteis em determinados casos, facilitando o processo de demonstração.

Tabela 3 – Regras de Inferência Derivadas

Modus Tollens (**mt**) $P \rightarrow Q \quad \neg Q$ $\neg P$	Silogismo Hipotético (**sh**) $P \rightarrow Q \; Q \rightarrow R$ $P \rightarrow R$
Silogismo Disjuntivo (**sd**) $P \vee Q \quad \neg P$ Q	Dilema Construtivo (**dc**) $P \vee Q \quad P \rightarrow R \quad Q \rightarrow S$ $R \vee S$
Exportação (**exp**) $(P \wedge Q) \rightarrow R$ $P \rightarrow (Q \rightarrow R)$	Inconsistência (**inc**) $P \quad \neg P$ Q

Fonte: [GLUZ e PY, 2010].

A ideia básica deste método é começar com as premissas P_1, P_2,... P_n (supostamente verdadeiras) e tentar aplicar regras de dedução até terminar com a conclusão Q. Esta conclusão teria que ser, então, verdadeira uma vez que os valores lógicos são sempre preservados sob as regras de inferência [GLUZ e PY, 2010].

Dessa forma uma demonstração formal da lógica proposicional seria formada por uma série de linhas numeradas com a seguinte estrutura:

1. P_1 (hipótese 1)

2. P_2 (hipótese 2)

...

n. P_n (hipótese n)

n+1. F_1 (fórmula obtida aplicando-se uma regra de dedução sobre as fórmulas anteriores)

n+2. F_2 (fórmula obtida aplicando-se uma regra de dedução sobre as fórmulas anteriores)

...

n+m. F_m (fórmula obtida aplicando-se uma regra de dedução sobre as fórmulas anteriores)

Q (fórmula obtida aplicando-se uma regra de dedução sobre as fórmulas anteriores)

Neste tipo de argumento a conclusão Q simplesmente é a fórmula contida na última linha obtida através da aplicação de uma regra de dedução.

A sequência de fórmulas obtidas por este processo é denominada de **sequência de demonstração** ou apenas de **demonstração formal** da conclusão em função de suas premissas.