UNIVERSIDADE ESTADUAL DO PARANÁ - UNESPAR

PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO

O PENSAMENTO COMPUTACIONAL NA PERSPECTIVA TEÓRICO-EPISTEMOLÓGICA DO MODELO DOS CAMPOS SEMÂNTICOS

Felipe de Oliveira Teixeira

Programa de Pós-Graduação em Educação Matemática
PRPGEM

Campo Mourão,
2025

PRPGEM
PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA

UNESPAR
Universidade Estadual do Paraná

UNIVERSIDADE ESTADUAL DO PARANÁ – UNESPAR
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM EDUCAÇÃO MATEMÁTICA – PRPGEM

O PENSAMENTO COMPUTACIONAL NA PERSPECTIVA TEÓRICO-EPISTEMOLÓGICA DO MODELO DOS CAMPOS SEMÂNTICOS

FELIPE DE OLIVEIRA TEIXEIRA

Dissertação apresentada ao Curso de **Mestrado do Programa de Pós-Graduação em Educação Matemática** da **Universidade Estadual do Paraná – UNESPAR**, linha de pesquisa *Tecnologia, diversidade e cultura em Educação Matemática*, como parte dos requisitos necessários à obtenção do título de **Mestre em Educação Matemática**.

Orientador:

Prof. Dr. Sérgio Carrazedo Dantas.

Campo Mourão – PR,
Abril de 2025

Ficha catalográfica elaborada pelo Sistema de Bibliotecas da UNESPAR e
Núcleo de Tecnologia de Informação da UNESPAR, com Créditos para o ICMC/USP
e dados fornecidos pelo(a) autor(a).

de Oliveira Teixeira, Felipe
O Pensamento Computacional na perspectiva teórico-epistemológica do Modelo dos Campos Semânticos / Felipe de Oliveira Teixeira. -- Campo Mourão-PR,2025.
264 f.: il.

Orientador: Sérgio Carrazedo Dantas.
Dissertação (Mestrado - Programa de Pós-Graduação Mestrado Acadêmico em Educação Matemática) -- Universidade Estadual do Paraná, 2025.

1. Pensamento Computacional. 2. Leitura plausível. 3. Modelo dos Campos Semânticos. 4. Conhecimento. 5. Significados. I - Carrazedo Dantas, Sérgio (orient). II - Título.

FELIPE DE OLIVEIRA TEIXEIRA

O PENSAMENTO COMPUTACIONAL NA PERSPECTIVA TEÓRICO-EPISTEMOLÓGICA DO MODELO DOS CAMPOS SEMÂNTICOS

Comissão Examinadora:

Documento assinado digitalmente
gov.br
SERGIO CARRAZEDO DANTAS
Data: 20/05/2025 15:59:36-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Sérgio Carrazedo Dantas – Presidente da Comissão Examinadora
Universidade Estadual do Paraná (UNESPAR), *Campus* de Apucarana

Documento assinado digitalmente
gov.br
PAULO HENRIQUE RODRIGUES
Data: 20/05/2025 16:05:41-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Paulo Henrique Rodrigues – Membro da Banca
Universidade Estadual do Paraná (UNESPAR), *Campus* de Paranavaí

Documento assinado digitalmente
gov.br
GUILHERME FRANCISCO FERREIRA
Data: 20/05/2025 16:33:18-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. Guilherme Francisco Ferreira – Membro da Banca
Universidade Estadual Paulista Júlio de Mesquita Filho (UNESP), *Campus* de Bauru

Documento assinado digitalmente
gov.br
JOAO PEDRO ANTUNES DE PAULO
Data: 20/05/2025 18:00:22-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. João Pedro Antunes de Paulo – Membro da Banca
Universidade Federal do Sul e Sudeste do Pará (UNIFESSPA), *Campus* de Marabá

Resultado: Aprovado.

Campo Mourão – PR,
Abril de 2025

À minha mãe, Eunice (in memorian).

AGRADECIMENTOS

Agradeço, em primeiro lugar, ao meu namorado, Bruno, por sempre ser tão atencioso e querido, pelo apoio incondicional ao longo desta jornada, especialmente nos momentos em que precisei me afastar para me dedicar à pesquisa, e por compartilhar comigo o entusiasmo pelo desenvolvimento deste trabalho.

Ao meu orientador, professor Dr. Sérgio Carrazedo Dantas, por ter sido mais do que um orientador, sempre disponível para auxiliar e contribuir generosamente na construção deste estudo.

Aos membros da banca, professores Dr. Guilherme Francisco Ferreira, Dr. João Pedro Antunes de Paulo e Dr. Paulo Henrique Rodrigues, pelas valiosas contribuições na qualificação e na defesa e por aceitarem compor a banca, enriquecendo esta pesquisa com suas perspectivas e sugestões.

Ao meu amigo e professor Me. Caio Juvanelli, cuja participação foi essencial desde o início desta jornada, revisando, lendo e relendo inúmeros trechos da dissertação, discutindo ideias e sugerindo caminhos, sempre me incentivando a extrair o melhor de mim. Seu apoio constante e dedicação foram fundamentais para a realização deste trabalho.

À minha amiga e professora Me. Vânia Sara Doneda de Oliveira, por me apresentar o Scratch em uma tarde de quinta-feira e, mais do que isso, por me auxiliar e me incentivar a procurar o professor Sérgio como orientador no mestrado, sugerindo o Pensamento Computacional como tema de pesquisa. Seu olhar atento e sua confiança no meu potencial foram decisivos para que eu trilhasse este caminho.

Ao meu amigo e professor Me. Leonardo Ferreira Zanatta, por sua generosa contribuição no processo de construção e revisão deste trabalho.

À todos os meus amigos e colegas de trabalho, pela companhia, incentivo e pelas inúmeras formas de apoio ao longo desta trajetória.

Sem cada um de vocês, a realização desta pesquisa não teria sido possível!

"And all this devotion I never knew at all
And the crashes are heaven for a sinner released
And the arms of the ocean delivered me"

Florence Welch, Ceremonials

RESUMO

O Pensamento Computacional tem se destacado na esfera educacional devido ao seu suposto potencial para desenvolver habilidades de resolução de problemas e promover uma compreensão profunda dos processos computacionais. Contudo, apesar de sua crescente inclusão nos currículos escolares, ainda não há consenso sobre uma definição do que constitui o Pensamento Computacional. Inserido nesse contexto, este estudo, caracterizado como uma pesquisa de desenvolvimento teórico, tem como objetivo apresentar uma perspectiva de Pensamento Computacional fundamentada no Modelo dos Campos Semânticos (MCS) no âmbito da Educação Matemática. O MCS é adotado como referencial metodológico e epistemológico para analisar os processos de produção de significados associados ao Pensamento Computacional, considerando não apenas as ações observáveis de um sujeito, mas também suas manifestações cognitivas subjacentes, acessadas por meio de suas enunciações. A metodologia baseia-se nos movimentos da leitura plausível, que incluíram uma revisão teórica dos conceitos de Pensamento Computacional e do MCS, seguida de ensaios teóricos que exploram processos como design e depuração em diferentes contextos e com variados objetos técnicos. A análise identificou a constituição de núcleos e campos semânticos que apontam, plausivelmente, para manifestações dos processos do Pensamento Computacional. Os resultados sugerem que o MCS oferece um quadro robusto para compreender o processo de produção de significados em atividades computacionais ou não computacionais, permitindo uma análise aprofundada das possíveis justificações e estratégias emergentes nessas atividades.

Palavras-chave: Pensamento Computacional; Leitura plausível; Modelo dos Campos Semânticos; Conhecimento; Significados.

ABSTRACT

Computational Thinking has gained prominence in the education sphere due to its supposed potential to develop problem-solving skills and foster a deeper understanding of computational processes. However, despite its growing inclusion in school curricula, there is still no consensus on what constitutes Computational Thinking. Within this context, this study, characterized as theoretical development research, aims to present a perspective on Computational Thinking based on the Semantic Fields Model (SFM) in the field of Mathematics Education. The SFM is adopted as a methodological and epistemological framework to analyze the processes of meaning-making associated with Computational Thinking, considering not only a subject's observable actions but also their underlying cognitive manifestations, accessed through their enunciations. The methodology is based on the movements of plausible reading, which included a theoretical review of the concepts of Computational Thinking and the SFM, followed by theoretical essays exploring processes such as design and debugging in different contexts and with various technical objects. The analysis identified the constitution of semantic nuclei and fields that plausibly indicate manifestations of Computational Thinking processes. The results suggest that the SFM provides a robust framework for understanding the process of meaning-making in both computational and non-computational activities, enabling a deeper analysis of the possible justifications and emerging strategies in theses activities.

Keywords: Computational Thinking; Plausible Reading; Semantic Fields Model; Knowledge; Meanings.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

BNCC	Base Nacional Comum Curricular
CAS	*Computer Algebra System*
CPU	Unidade Central de Processamento
MCS	Modelo dos Campos Semânticos
NSF	*National Science Foundation*
PC	Pensamento Computacional
PRPGEM	Programa de Pós-Graduação em Educação Matemática
SBC	Sociedade Brasileira de Computação
UAL	Unidade de Aritmética e Lógica
UNESPAR	Universidade Estadual do Paraná

SUMÁRIO

1 CONTEXTUALIZANDO A INVESTIGAÇÃO

1.1 Apresentação

Para contextualizar o desenvolvimento da pesquisa de mestrado relatada neste documento, considero[1] relevante apresentar brevemente minha trajetória acadêmica e como cheguei ao tema Pensamento Computacional.

Em 2017, ingressei no curso de Licenciatura em Matemática da Universidade Estadual do Paraná (Unespar), *Campus* de Campo Mourão. Nos primeiros anos do curso, mantive incerteza quanto à escolha da carreira docente, especialmente ao enfrentar as disciplinas de Estágio Supervisionado I e II. Ainda assim, decidi prosseguir com o curso.

Concluí a graduação em 2020, em meio à pandemia de Covid-19, ainda sem uma decisão definitiva sobre a carreira docente. Contudo, em abril de 2022, comecei a atuar como professor temporário na rede estadual de ensino, assumindo turmas de Matemática da 2ª série do Ensino Médio. Desse modo, minha primeira experiência em sala de aula foi com o ensino de equações e funções trigonométricas.

Algumas semanas após o início das aulas, em uma quinta-feira à tarde, fui chamado à secretaria da escola. A secretária me ofereceu a oportunidade de assumir mais algumas aulas adicionais, em virtude da saída de uma professora que assumiria a função de direção auxiliar. Ao analisar a planilha de aulas, notei a sigla "PC" e perguntei seu significado, sendo informado de que se tratava da disciplina de Pensamento Computacional.

Diante da proposta, manifestei minha preocupação, explicando que não possuía conhecimentos em programação e, por isso, não me sentia preparado para lecionar a disciplina. A secretária então sugeriu que eu conversasse com a professora responsável para obter mais informações. Perguntei quem era a professora e fui informado de que seu nome era Vânia Sara. Decidi, assim, procurá-la para esclarecer minhas dúvidas.

Após caminhar alguns minutos pelos corredores da escola, a encontrei e ela, de

[1] Nesta seção, opta-se pelo uso da primeira pessoa do singular, em razão do caráter descritivo e pessoal dos acontecimentos narrados, que traçam a história do pesquisador até o desenvolvimento desta pesquisa. Nas demais seções do trabalho, adota-se uma linguagem impessoal, utilizando-se construções verbais na voz passiva.

forma bastante atenciosa, me apresentou o Scratch, uma plataforma de programação visual que poderia ser utilizada para o ensino de Pensamento Computacional. Convencido com sua explicação, retornei à secretaria e aceitei o desafio de ministrar as aulas.

Quando perguntei sobre a data de início, imaginei que teria o final de semana para estudar o Scratch e me preparar melhor, mas fiquei surpreso ao saber que as aulas começariam no dia seguinte. Voltei para casa e imediatamente comecei a explorar o Scratch, criando uma conta na plataforma e esboçando alguns projetos. Enviei uma mensagem para a professora Vânia solicitando orientações ou materiais de apoio. Ela prontamente me enviou diversos recursos e ainda compareceu à aula no dia seguinte, apresentando-me às turmas e sugerindo metodologias de ensino.

Alguns meses depois, a professora Vânia me abordou na sala dos professores e sugeriu que eu considerasse ingressar em um mestrado focado em Pensamento Computacional. Ela recomendou que eu procurasse o professor Sérgio, do Programa de Pós-Graduação em Educação Matemática (PRPGEM), pois acreditava que eu tinha potencial para ser aprovado. Embora não estivesse completamente decidido a seguir para a pós-graduação naquele momento, elaborei um projeto de pesquisa e me inscrevi no processo seletivo em janeiro de 2023, sendo aprovado sob a orientação do professor Dr. Sérgio Carrazedo Dantas.

Nos primeiros encontros e discussões no Grupo de Pesquisa Autômato[2], comecei a aprofundar meus estudos sobre Pensamento Computacional. No entanto, uma questão permanecia em aberto e frequentemente surgia nas orientações: as definições e concepções encontradas na literatura não esclareciam de forma satisfatória o que se entende por "pensamento" ou como o Pensamento Computacional se configura como um tipo específico de pensamento.

No segundo semestre de 2023, participei da disciplina "Conhecimento, Educação Matemática e Práticas Pedagógicas", ministrada pela professora Dra. Márcia Cristina da Costa Trindade Cyrino e pelo professor Dr. Everton José Goldoni Estevam[3]. Uma das

[2] Grupo de Pesquisa e Estudos em Educação Matemática vinculado à Universidade Estadual do Paraná (Unespar), *Campus* de Apucarana, o qual o pesquisador autor deste trabalho faz parte e tem como líder o Prof. Dr. Sérgio Carrazedo Dantas, que orienta esta pesquisa.

[3] A referida disciplina foi ofertada pelo Programa de Pós-Graduação em Educação Matemática

primeiras leituras discutidas em aula foi o texto de Lins (1993), que explora o papel da epistemologia na Educação Matemática.

Ao aprofundar meus estudos nos escritos de Lins (1993, 1999), procurei o professor Everton para discutir possíveis pesquisas relacionadas ao Modelo dos Campos Semânticos (MCS), pois percebi que esses textos tinham uma abordagem peculiar, na minha perspectiva, para a noção de conhecimento. Ele recomendou a leitura do livro "Modelo dos Campos Semânticos: estabelecimentos e notas de teorizações" (Angelo et al., 2012). Para minha surpresa, identifiquei o nome do professor Sérgio como um dos autores, o que indicava que meu orientador poderia me oferecer suporte adequado para desenvolver uma pesquisa fundamentada no MCS.

Com o avanço das aulas no mestrado e as discussões realizadas no Grupo de Pesquisa Autômato, iniciei, juntamente com o meu orientador, discussões e reflexões sobre a *leitura plausível* como uma metodologia de pesquisa e a possibilidade de utilizar o MCS para desenvolver uma perspectiva teórica sobre o Pensamento Computacional. Assim, delineamos a justificativa, a problemática e o objetivo de pesquisa, que emergiram naturalmente ao longo das orientações e são apresentadas na sequência.

1.2 Introdução

O Pensamento Computacional[4] tem ganhado destaque tanto na Educação quanto na Ciência da Computação, especialmente por seu suposto potencial em desenvolver habilidades de resolução de problemas e promover uma compreensão mais profunda dos processos computacionais. Entretanto, apesar da notoriedade alcançada e da crescente inclusão dessa expressão nos currículos escolares, ainda não há uma definição clara e consensual sobre o que constitui, de fato, o Pensamento Computacional.

Raabe, Couto e Blikstein (2020) apontam que a literatura atual não estabelece

(PRPGEM) da Universidade Estadual do Paraná (Unespar), *Campus* de Campo Mourão e União da Vitória. Informações sobre o programa e a disciplina podem ser encontradas no site https://prpgem.unespar.edu.br/informacoes-gerais/disciplinas. Acesso em: 16 mar. 2025.

[4] A partir deste momento, adota-se uma linguagem impessoal e empregam-se construções na voz passiva sintética. Essa escolha reflete o caráter coletivo e objetivo do processo de pesquisa, desenvolvido em colaboração e sob a orientação do professor Dr. Sérgio Carrazedo Dantas.

com precisão se o Pensamento Computacional é uma forma distinta de pensamento, uma combinação de processos mentais preexistentes, ou se é apropriado caracterizá-lo como um tipo específico de "pensamento". Essa indefinição afeta diretamente o desenvolvimento, a avaliação e a aplicação do conceito, principalmente no contexto educacional, onde se tem buscado promover e mensurar essa habilidade[5].

Diante desse panorama, este trabalho se propõe a investigar uma perspectiva ainda pouco explorada: descrever o Pensamento Computacional em termos efetivamente cognitivos. As abordagens correntes frequentemente se baseiam em ações observáveis dos indivíduos para inferir a ocorrência do Pensamento Computacional. Por exemplo, quando um aluno decompõe um problema ou depura uma solução, deduz-se que ele está "pensando computacionalmente". No entanto, essa abordagem se concentra nas ações objetivas, deixando de lado os processos mentais subjacentes que as originam. Portanto, pode-se inferir uma lacuna na compreensão dos aspectos cognitivos envolvidos nesse conceito.

Contudo, as discussões desenvolvidas neste trabalho não pretendem esgotar a temática do Pensamento Computacional, fornecendo respostas definitivas para questões como "O que é Pensamento Computacional?" ou "Como saber se uma pessoa possui Pensamento Computacional?". Pelo contrário, busca-se contribuir para o avanço das pesquisas que têm sido realizadas no âmbito educacional, especialmente no contexto da Educação Matemática.

Este trabalho se configura como uma pesquisa de desenvolvimento teórico, voltada ao conceito de Pensamento Computacional e à sua presença nos processos de produção de *significados*[6] no âmbito educacional, com ênfase na Educação Matemática. Para isso, este trabalho fundamenta-se teórica e epistemologicamente[7] no Modelo dos Campos Semânticos (MCS), que compreende a cognição como um processo de produção de *significados* (Ferreira, 2020). Segundo (Lins, 2012, p. 18), o MCS possibilita "leituras suficientemente

[5] Embora não seja o foco deste trabalho, ao longo do texto serão apresentados alguns possíveis motivos para a inserção do Pensamento Computacional na Educação Básica.

[6] Neste trabalho, termos e expressões próprias do Modelo dos Campos Semânticos (MCS) são destacadas em itálico.

[7] Neste trabalho, o termo epistemologia é compreendido desde a perspectiva de Lins (1993), que o define como a "a atividade humana que estuda as seguintes questões: (i) o que é conhecimento?; (ii) como é que conhecimento é produzido?; e, (iii) como é que conhecemos o que conhecemos?" (Lins, 1993, p. 77).

finas de processos de produção de significados", permitindo caracterizar o Pensamento Computacional para além das ações observáveis, considerando os processos subjacentes de produção de *significados* por meio de enunciações.

O objetivo desta pesquisa é, portanto, *apresentar uma perspectiva de Pensamento Computacional fundamentada no MCS no âmbito da Educação Matemática*. Propõe-se identificar os processos de produção de *significados*, as *direções de interlocução*, os *núcleos* e os *campos semânticos* que caracterizam o Pensamento Computacional, discutindo como esses processos se manifestam em atividades diversas.

Caracterizar um tipo específico de pensamento – neste caso, o Pensamento Computacional – sob a perspectiva do MCS envolve descrever suas manifestações e identificar os processos cognitivos específicos a essa forma de pensar (Lins; Gimenez, 1997). Assim, a construção teórica desenvolvida neste trabalho visa descrever o Pensamento Computacional a partir das enunciações que emergem em diferentes atividades, configurando-se em enunciações que, *plausivelmente*, sugerem processos de decomposição, abstração ou reconhecimento de padrões, por exemplo. Nesse sentido, a *leitura plausível* assume um papel central na formulação dessa perspectiva, pois permite dialogar com a literatura sobre o tema e identificar, ao longo dos exemplos apresentados, as enunciações que apontam para cada processo cognitivo.

Para alcançar o objetivo proposto, este trabalho está organizado em três capítulos. O capítulo intitulado *O Pensamento Computacional* elucida diferentes perspectivas sobre o conceito na literatura, apresentando um contexto histórico e destacando a crescente incorporação nos currículos educacionais. Argumenta-se que, apesar de sua ampla adoção, ainda não há uma construção teórica que permita constituir o Pensamento Computacional em termos cognitivos.

Na sequência, o capítulo *O Modelo dos Campos Semânticos* introduz as noções centrais do MCS, que servirão como base para a construção teórica proposta. Discute-se a cognição como um processo de produção de *significados*, além das noções de *direção de interlocução*, *núcleo* e *campo semântico*. Destaca-se também a relevância da *leitura plausível* como metodologia de pesquisa no contexto da Educação Matemática.

No capítulo *O Pensamento Computacional desde a perspectiva do Modelo dos Campos Semânticos*, propõe-se uma teorização sobre como o Pensamento Computacional pode ser compreendido à luz do MCS. Cada um dos processos característicos do Pensamento Computacional é analisado em termos de produção de *significados*, identificando as enunciações que sugerem processos de design, decomposição, abstração, produção de algoritmos, reconhecimento de padrões ou depuração.

Encerra-se este texto com o capítulo *Considerações Finais*, sintetizando os principais resultados do desenvolvimento teórico e dos ensaios conduzidos.

Cabe salientar que este trabalho contém diversas figuras elaboradas especificamente para ilustrar métodos matemáticos e construções realizadas em softwares, com o objetivo de exemplificar os processos do Pensamento Computacional. Essas figuras, quando visualizadas por meio de softwares leitores de PDF, como Adobe Acrobat ou Foxit Reader, apresentam animações que contribuem para a dinamização e sustentação das argumentações aqui desenvolvidas. Assim, recomenda-se que o leitor acesse este texto em um computador e utilize um software adequado para a leitura de arquivos PDF, a fim de visualizar corretamente as animações incorporadas.

2 O PENSAMENTO COMPUTACIONAL

A expressão Pensamento Computacional ganhou notoriedade nos últimos anos, tornando-se tema de várias pesquisas acadêmicas (Wing, 2006; Brennan; Resnick, 2012; Tedre; Denning, 2016; Brackmann, 2017; Valente et al., 2017; Boucinha et al., 2019; Denning; Tedre, 2019; Valente, 2019; Barros, 2020; Ribeiro; Foss; Cavalheiro, 2020; Espadeiro, 2021; Bertazini, 2022; Dantas, 2022, 2023; Teixeira; Juvanelli; Dantas, 2024). No âmbito educacional, há diversas perspectivas que orientam a relação entre Computação, Pensamento Computacional e Educação.

Diante dessa diversidade que será abordada ao longo deste texto, é ingênuo assumir a possibilidade de uma leitura que abarque todo o cenário. As diferentes perspectivas de Pensamento Computacional apontam conceitos, práticas e habilidades que remontam desde a civilização grega até a recente constituição dos currículos escolares. Contudo, uma característica predominante das definições ou concepções[1] sobre Pensamento Computacional é o foco em ações objetivas, argumentando-se que um determinado indivíduo "possui" Pensamento Computacional a partir do que ele faz. Nesse contexto, este capítulo tem como objetivo levantar discussões e reflexões sobre possíveis posições epistemológicas a respeito do Pensamento Computacional.

Fundamentando-se em autores que apresentam perspectivas de Pensamento Computacional na Ciência da Computação e na Educação Matemática, propõe-se uma leitura sobre as discussões acerca do Pensamento Computacional no âmbito educacional. Não se trata de um Estado da Arte ou Estado do Conhecimento[2], mas sim de trazer considerações com base em resultados apresentados por outros estudos, sustentando que ainda

[1] Neste texto, entende-se que uma definição de Pensamento Computacional busca delimitar a noção de forma mais objetiva e estruturada, estabelcendo critérios e elementos essenciais. Já uma concepção refere-se a compreensões mais amplas, que podem variar conforme perspectivas teóricas, contextos de aplicação e abordagens educacionais.

[2] "Para Romanowski e Ens (2006), existe diferença entre essas expressões: Estado da Arte tem maior amplitude, pois seu caráter inventariante abrange um conjunto maior de fontes de análises referentes a determinada área do conhecimento (dissertações, teses, livros, artigos publicados em eventos e artigos publicados em revistas científicas). O Estado do Conhecimento, por sua vez, diz respeito a um recorte menor do universo dessas publicações (textos acadêmicos de uma determinada revista científica ou anais de eventos acadêmicos e teses/dissertações publicadas)" (Moraes, 2016, p. 27-28).

não há uma construção teórica que permita definir, avaliar ou mensurar o Pensamento Computacional em termos cognitivos.

Para sustentar a argumentação proposta, este capítulo organiza-se da seguinte maneira: inicialmente, realiza-se um resgate histórico traçando possíveis raízes históricas do Pensamento Computacional em métodos que remontam à Grécia antiga até a invenção dos computadores eletrônicos no século XX; em seguida, apresentam-se as mudanças na percepção dos computadores por cientistas e pesquisadores entre as décadas de 1950 e 1990, evidenciando que, apesar dos avanços teóricos, não houve impacto significativo no meio educacional, em termos metodológicos e epistemológicos; na sequência, são apresentadas e problematizadas algumas perspectivas de Pensamento Computacional pós-Wing (2006), com ênfase em suas repercussões educacionais; por fim, discute-se sobre a Base Nacional Comum Currilar (BNCC) como catalisadora da ascenção do Pensamento Computacional no contexto educacional brasileiro.

2.1 Possíveis raízes históricas do Pensamento Computacional

Diante das diferentes posições epistemológicas sobre os termos *pensamento* e *computacional*, a busca por uma definição consistente de Pensamento Computacional tem sido tema de diversas discussões e pesquisas na esfera acadêmica (Wing, 2006; Brennan; Resnick, 2012; Tedre; Denning, 2016; Brackmann, 2017; Valente et al., 2017; Boucinha et al., 2019; Denning; Tedre, 2019; Valente, 2019; Barros, 2020; Ribeiro; Foss; Cavalheiro, 2020; Espadeiro, 2021; Bertazini, 2022; Dantas, 2022, 2023; Teixeira; Juvanelli; Dantas, 2024). Nesse contexto, embora seja possível discutir o Pensamento Computacional sob várias perspectivas, uma das definições modernas mais aceitas é a de *formular problemas de modo que suas soluções possam ser expressas como etapas computacionais executadas por um computador* (Denning; Tedre, 2019). Essas definições geralmente associam o termo computador à estrutura das máquinas de computação eletrônicas com arquitetura von Neumann[3].

Assumir o Pensamento Computacional a partir de tais definições implica difundi-lo

[3] A estrutura da máquina eletrônica de von Neumann será detalhada mais adiante, nesta seção.

como uma forma de pensar oriunda da Ciência da Computação, conforme apontam Wing (2006) e Brackmann (2017). Contudo, Denning e Tedre (2019) argumentam que muitos dos princípios fundamentais do Pensamento Computacional existiam antes mesmo da invenção das máquinas de computação eletrônicas na década de 1940.

Desse modo, esta seção propõe-se a apresentar recortes históricos no qual se identificam manifestações de práticas e conceitos associados ao Pensamento Computacional, desde registros na antiguidade até a invenção dos computadores eletrônicos em meados do século XX. O objetivo é argumentar que habilidades e estratégias frequentemente relacionadas ao Pensamento Computacional podem ser reconhecidas na concepção, organização e execução de diversos métodos e procedimentos utilizados para a resolução de problemas, assim como na elaboração de máquinas de computação anteriores à era moderna.

Para tanto, as discussões são sustentadas pelas ideias de Denning e Tedre (2019), considerando que este trabalho oferece uma perspectiva que permite evidenciar o avanço da computação e como esse modo de pensar sofreu alterações ao longo do tempo. A partir das discussões levantadas, torna-se possível argumentar que, desde os primórdios, o Pensamento Computacional tem sido tratado como um modo de pensar, mesmo sem explicitar quais são os processos cognitivos que constituem esse tipo de pensamento.

Para esta problematização inicial[4], foram considerados elementos essenciais do Pensamento Computacional os quatro pilares ou dimensões do Pensamento Computacional, comumente retratados na literatura dedicada ao tema, a saber: decomposição, reconhecimento de padrões, abstração e os algoritmos[5] (Brackmann, 2017; Boucinha et al., 2019; Barros, 2020; Bertazini, 2022). Brackmann (2017) sumariza esses processos conforme descrito a seguir:

> O Pensamento Computacional envolve identificar um problema complexo e

[4] Evidentemente, existem outras perspectivas de Pensamento Computacional. Ribeiro, Foss e Cavalheiro (2020), por exemplo, defendem a perspectiva de que o Pensamento Computacional pode ser resumido em apenas três pilares: abstração, análise e automação. Outros autores, como Dantas (2023), afirmam que o Pensamento Computacional não pode ser sintetizado em apenas quatro pilares, e que se deve agregar a formulação do problema e a depuração para compreender todas as ações mentais englobadas no processo de resolução de problemas com o Pensamento Computacional. No entanto, opta-se, para esta problematização inicial, considerar os quatro pilares pois são os que aparecem com maior frequência na literatura consultada (Brackmann, 2017; Boucinha et al., 2019; Barros, 2020; Bertazini, 2022).

[5] Cada uma dessas dimensões será mais bem detalhada em uma seção posterior.

> quebrá-lo em pedaços menores e mais fáceis de gerenciar (DECOMPOSIÇÃO). Cada um desses problemas menores pode ser analisado individualmente com maior profundidade, identificando problemas parecidos que já foram solucionados anteriormente (RECONHECIMENTO DE PADRÕES), focando apenas nos detalhes que são importantes, enquanto informações irrelevantes são ignoradas (ABSTRAÇÃO). Por último, passos ou regras simples podem ser criados para resolver cada um dos subproblemas encontrados (ALGORITMOS) (Brackmann, 2017, p. 33).

Esta seção organiza-se da seguinte maneira: inicialmente, evidencia-se a relação entre métodos da civilização grega para o cálculo de áreas de figuras curvilíneas e as definições contemporâneas de Pensamento Computacional; em seguida, apresenta-se uma análise semelhante com método modernos do Cálculo Diferencial e Integral, no século XIX; na sequência, elucidam-se as dificuldades encontradas por matemáticos, engenheiros e inventores na elaboração de máquinas de computação para automatizar processos e a relação com elementos de Pensamento Computacional; por fim, discorre-se sobre o processo de concepção dos computadores eletrônicos modernos e os elementos de Pensamento Computacional presentes nessa concepção.

2.1.1 Pensamento Computacional em métodos da civilização grega

Denning e Tedre (2019) sustentam que os primórdios do Pensamento Computacional remontam à civilização babilônica, entre 1800 e 1600 a.C., evidenciando-se em práticas rudimentares de cálculo e procedimentos gerais para resolver problemas matemáticos. Segundo os autores, esses procedimentos baseados em regras possuem características que, sob a perspectiva atual[6], poderiam ser identificados como manifestações de Pensamento Computacional.

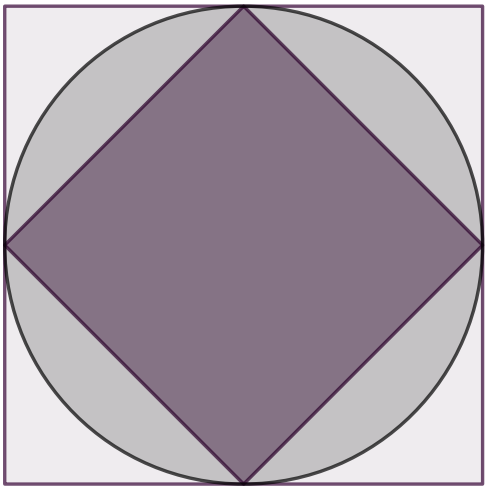
Esta pesquisa, no entanto, inicia as discussões com o Método da Exaustão, concebido por Eudoxo e posteriormente aprimorado por Arquimedes e Euclides. O aprimoramento deste método permitiu sua aplicação a problemas mais gerais, visando obter estimativas para a área de figuras curvilíneas. Segundo Rezende (2003), este método foi empregado para manipular operações com limites sem invocar o conceito de infinito.

[6] Não se pretende cometer um anacronismo ao identificar práticas matemáticas antigas como Pensamento Computacional moderno. O intuito é apenas elucidar que essas práticas possuem elementos que, sob uma perspectiva atual, poderiam ser vistos como manifestações iniciais desse tipo de pensamento, conforme será explorado e argumentado ao longo do texto.

O Método da Exaustão consiste em envolver figuras curvilíneas com formas geométricas mais simples, como triângulos ou quadrados, e calcular a área dessas formas. A soma das áreas dessas formas aproxima a área da figura original. Assim, ao reduzir gradualmente as medidas das formas utilizadas, é possível obter uma estimativa progressivamente mais precisa da área da figura original.

De acordo com Baron (1985), Arquimedes aprimorou o Método da Exaustão para determinar a área de um círculo. O método consiste em aproximar a área A de um círculo de raio r pela média das áreas de dois polígonos regulares semelhantes, um inscrito e outro circunscrito ao círculo. A Figura 2.1 ilustra geometricamente esse processo iterativo, considerando polígonos regulares de quatro até vinte lados.

Figura 2.1 – Exemplos de utilização do método para polígonos de quatro até vinte lados



Fonte: Os autores.

Considerando o objetivo desta seção, evidenciar a relação entre métodos da civilização grega para o cálculo de áreas de figuras curvilíneas e as definições contemporâneas de Pensamento Computacional, é possível sistematizar o método apresentado na Figura 2.1 no formato de um algoritmo, ou seja, em uma sequência de ações. Uma possível solução é apresentada a seguir, no Quadro 2.1.

Quadro 2.1 – Método de Arquimedes para o cálculo da área de um círculo.

Passo	Comando
1	Defina o raio do círculo $r > 0$.
2	Defina uma tolerância para o erro máximo a ser cometido $\varepsilon > 0$.
3	Defina um número inicial de lados para os polígonos inscritos e circunscritos $l \geq 3$.
4	Calcule as áreas dos polígonos incritos e circunscritos para o número de lados definido.
5	Calcule a diferença d entre a área do polígono circunscrito e inscrito.
6	Repita os passos 4 e 5 até que se obtenha a precisão almejada, isto é, até que $d < \varepsilon$, tomando, a cada nova iteração, $l \leftarrow l + 1$.
7	Calcule a média das áreas dos polígonos inscritos e circunscritos refinados.

Fonte: Elaborado pelo autor (2025).

O algoritmo descrito no Quadro 2.1 apresenta o processo de aproximação da área de um círculo inscrevendo e circunscrevendo polígonos regulares, e refinando esses polígonos para obter uma estimativa progressivamente mais precisa da área do círculo. Considerando as dimensões do Pensamento Computacional apresentadas (decomposição, reconhecimento de padrões, abstração e algoritmos), é possível relacionar elementos dessa perspectiva com o processo de obtenção da estimativa da área do círculo. A seguir, explanam-se essas relações.

A decomposição pode ser evidenciada no processo de se buscar a solução em partes, inicialmente definindo a quantidade de lados dos polígonos que serão utilizados para aproximar a área do círculo. Calculam-se as áreas desses polígonos e avalia-se a diferença entre elas em relação à tolerância estabelecida inicialmente.

O reconhecimento de padrões pode ser identificado em ao menos três momentos distintos. Primeiramente, destaca-se o reconhecimento de padrões de convergência, no qual a diminuição da diferença entre as áreas dos polígonos inscritos e circunscritos em relação à área do círculo sugere uma maior precisão na estimativa à medida que o número de lados aumenta. Além disso, o padrão de redução do erro indica uma maior precisão na estimativa da área à medida que se refinam os polígonos utilizando-se mais lados. Por fim,

é possível observar que os polígonos inscritos e circunscritos formam padrões geométricos conforme o número de lados aumenta, evidenciando a relação entre o círculo e os polígonos utilizados para aproximá-lo.

A abstração é identificável no processo ao se desconsiderar as complexidades envolvidas na obtenção da área do círculo, concentrando-se exclusivamente nas propriedades geométricas dos polígonos inscritos e circunscritos. Esse enfoque permite a utilização de técnicas geométricas conhecidas para calcular as áreas desses polígonos e, consequentemente, estimar a área do círculo.

Por fim, o algoritmo do método de Arquimedes descreve uma sequência de passos para calcular uma estimativa para a área do círculo. Esses passos podem ser seguidos de maneira sistemática para obter uma estimativa tão precisa quanto necessário, demonstrando a formulação de um algoritmo para a resolução do problema.

Não se pretende, com esta problematização inicial, sugerir que Arquimedes seguiu procedimentos inteiramente modernos ao empregar o método para estimar a área de um círculo. Pelo contrário, almeja-se evidenciar que as ações de conceber, refinar, organizar e até mesmo empregar o método exposto se assemelham ao que atualmente poderia ser caracterizado como um tipo primitivo de Pensamento Computacional.

Na sequência, realiza-se um movimento de análise semelhante para um método do Cálculo Diferencial e Integral, a Soma de Riemann.

2.1.2 Pensamento Computacional em métodos do Cálculo Diferencial e Integral

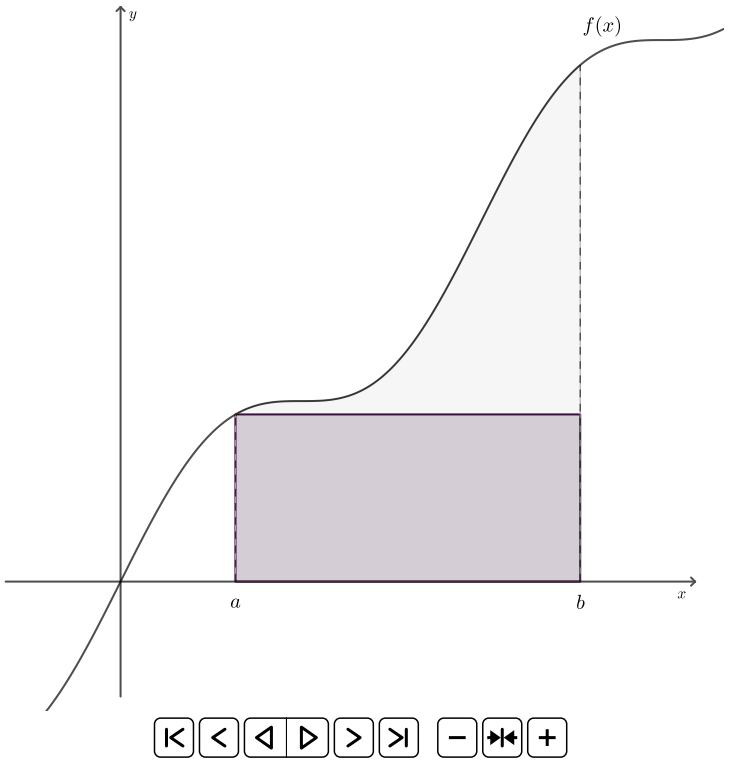
Aproximadamente dois mil anos após a formalização do Método da Exaustão, o Cálculo foi proposto independentemente por Newton (1642-1727) e Leibniz (1646-1716) no final do século XVII, aperfeiçoando a abordagem de aproximação de objetos e curvas por meio de cálculos em séries infinitas (Denning; Tedre, 2019). Leibniz mostrou como o Cálculo poderia ser utilizado para se encontrar a área de uma região limitada por uma curva, determinando uma integral definida por antidiferenciação (Leithold, 1994).

No século XIX, Cauchy (1789-1857) definiu a integral definida de uma função

contínua pelo limite de uma soma parcial finita de retângulos, que aproximam inicialmente a região dada, e cujas bases vão tendendo a zero à medida que o número de retângulos aumenta (Rezende, 2003). Essa ideia foi posteriormente aperfeiçoada e rigorosamente formalizada por Riemann (1826-1866), sendo conhecida atualmente como Soma de Riemann.

A Figura 2.2 ilustra um exemplo da utilização de retângulos para aproximar a área limitada por uma curva $f(x)$ em um intervalo $[a, b]$, tal que $a, b \in \mathbb{R}$.

Figura 2.2 – Exemplos de utilização de retângulos para aproximar a área limitada por uma curva



Fonte: Os autores.

Assim como no Método da Exaustão, é possível organizar a Soma de Riemann aos moldes de um algoritmo. Uma possível solução é apresentada a seguir, no Quadro 2.2.

Quadro 2.2 – Soma de Riemann para aproximação da área abaixo de uma curva.

Passo	Comando
1	Considere uma função contínua $y = f(x)$ definida em um intervalo $[a, b]$.
2	Divida o intervalo $[a, b]$ em n segmentos de mesmo comprimento ao longo do eixo x.
3	Calcule o comprimento de cada segmento infinitesimal $\Delta x = (b - a)/n$.
4	Construa retângulos de modo que as bases sejam os segmentos Δx, possuam a altura dada pelo valor da função no ponto correspondente ao segmento, ou seja, $f(x)$ e possuam a largura do próprio segmento Δx.
5	Calcule a área de cada um dos retângulos construídos.
6	Some a área de todos os retângulos ao longo do intervalo $[a, b]$.

Fonte: Elaborado pelo autor (2025).

O método apresentado no Quadro 2.2 elucida, em uma notação algorítmica, o procedimento aperfeiçoado por Riemann para estimar a área sob uma curva definida por uma função contínua $f(x)$ em um intervalo $[a, b]$. Em um movimento semelhante ao realizado com o método de Arquimedes para a aproximação da área de um círculo, é possível estabelecer relações entre a organização do método da Soma de Riemann e os pilares do Pensamento Computacional conforme proposto por Brackmann (2017).

A decomposição manifesta-se no método ao dividir inicialmente o intervalo $[a, b]$ em intervalos menores e, em seguida, construir retângulos para aproximar a área sob a curva. Soma-se então as áreas dos retângulos, obtendo-se uma aproximação tão precisa quanto necessário. Assim, focar em cada uma das partes do método poderia reduzir a complexidade do problema, tornando a identificação de erros no processo mais evidente e facilitando a implementação de correções.

É possível inferir que o reconhecimento de padrões é um dos processos mentais que permitiu a própria concepção da Soma de Riemann. Isso pode ser identificado ao notar que quanto menor o intervalo considerado, mais precisa é a estimativa. Desse modo, quando o número de segmentos infinitesimais tende ao infinito ($n \to \infty$), o valor encontrado seria não uma estimativa, mas o valor exato da área sob a curva definida por $f(x)$.

A abstração também desempenhou um papel central na formalização desse método.

A abordagem de simplificação da curva contínua $f(x)$ em figuras geometricamente mais simples, cuja fórmula para obtenção da área já era conhecida, permitiu a construção e a formalização da Soma de Riemann. A organização do método em procedimentos claros, precisos e inequívocos, sem ambiguidade, explicita a importância da abordagem algorítmica da Soma de Riemann. A sistematização do método permitia sua utilização por não especialistas; com refinamentos adequados, qualquer pessoa capaz de seguir as instruções poderia empregá-lo.

Novamente, não se pretende afirmar que Riemann seguiu exatamente os procedimentos descritos nos parágrafos anteriores para estimar a área abaixo de uma curva. No entanto, a divisão do problema em várias etapas, a identificação que a precisão da estimativa melhora conforme o comprimento dos segmentos infinitesimais Δx diminui, bem como a abstração de uma curva em segmentos menores são características das perspectivas modernas de Pensamento Computacional.

Na sequência, discorre-se sobre o Pensamento Computacional a partir da invenção das máquinas de computação.

2.1.3 Pensamento Computacional na automatização de processos

À medida que os métodos do Cálculo Diferencial e Integral e de outras áreas foram aprimorados e a Matemática evoluiu como área do conhecimento, a complexidade envolvida na aplicação desses métodos aumentou significativamente. Mesmo com os denominados computadores[7], especialistas matematicamente treinados e dedicados à realização de cálculos complexos, o processo tornou-se cada vez mais propenso a erros.

Denning e Tedre (2019) argumentam que alguns métodos eram excessivamente complexos para serem facilmente memorizados, enquanto outros precisavam ser repetidos muitas vezes. Isso tornava difícil para seres humanos, sujeitos a distrações, concluí-los sem erros. Por essa razão, matemáticos e inventores buscaram desenvolver máquinas de computação e auxiliares de cálculo que permitissem a realização de cálculos mais longos com menor propensão a erros.

[7] O próprio termo "computador" significa "aquele que calcula" e data do início dos anos 1600 (Denning; Tedre, 2019).

O Pensamento Computacional foi fundamental para que os matemáticos, engenheiros e inventores idealizassem e desenvolvessem as primeiras máquinas de computação. A abstração de operações matemáticas, *loops*[8] e condicionais em mecanismos físicos, como rodas e engrenagens, permitiu a elaboração dessas máquinas (Denning; Tedre, 2019).

Assim, em meados do século XVII, surgiram as primeiras máquinas de computação, como a régua de cálculo e a calculadora aritmética, que, embora populares, apresentavam limitações operacionais. A régua de cálculo, por exemplo, realizava multiplicações baseadas na soma de logaritmos, mas não era capaz de somar ou subtrair. Em contrapartida, a calculadora aritmética permitia somar e subtrair, mas não multiplicar ou dividir.

Denning e Tedre (2019) destacam que a inclusão de softwares[9] armazenados internamente na memória das máquinas foi crucial para superar tais restrições. Segundo os autores, um software poderia realizar cálculos adicionais e ser ajustado para alterar a operação da máquina, ampliando significativamente sua funcionalidade. No século XIX, esse conceito permitiu que Charles Babbage (1791-1871) avançasse na projeção de máquinas de computação, incluindo a automação de tabelas de logaritmos e senos e o desenvolvimento de um computador de uso geral, capaz de realizar qualquer função computável (Denning; Tedre, 2019).

Contudo, a construção das máquinas de computação programáveis via software era complexa. Os métodos de engenharia mecânica da época não permitiam a fabricação precisa e em larga escala das engrenagens e alavancas necessárias, resultando em frequentes falhas e emperramentos.

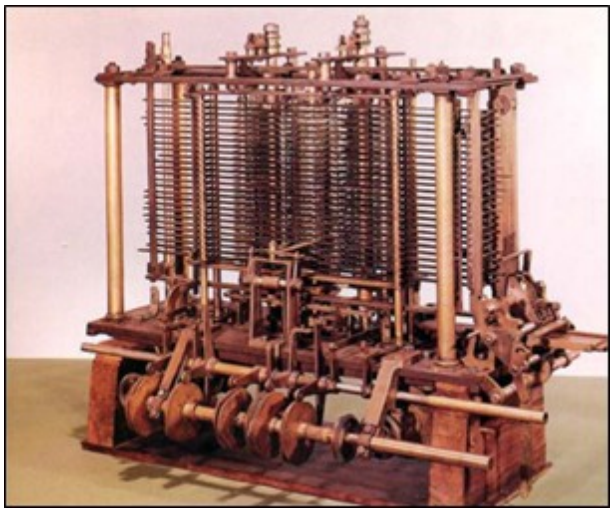
Na década de 1830, Babbage desenvolveu um novo projeto, a *Analytical Engine*, uma máquina mais eficiente, que requeria menos componentes e possuía maior potencial (Denning; Tedre, 2019). A Figura 2.3 ilustra uma versão parcialmente construída da *Analytical Engine*, atualmente no *Science Museum* de Londres. Cabe salientar que, segundo Costa (2012), essa construção não é uma réplica, pois a máquina jamais foi construída no

[8] Um *loop* é uma sequência de instruções que são repetidas várias vezes até que uma determinada condição de parada seja satisfeita.

[9] Softwares são programas, isto é, um conjunto de instruções que permitem realizar tarefas específicas em computadores.

período de Babbage.

Figura 2.3 – *Analytical Engine* de Charles Babbage



Fonte: Costa (2012).

Para "instruir mecanicamente" a máquina sobre os cálculos a serem realizados, Babbage adotou cartões perfurados, solução que, segundo Costa (2012), permitia à máquina "memorizar" sequências de instruções e conjuntos de dados. Segundo o autor, com o uso dos cartões perfurados, a *Analytical Engine* pôde incorporar uma capacidade praticamente ilimitada de operações, repetições, variáveis e constantes, atendendo a uma ampla gama de cálculos.

Costa (2012) explica que, para operar a *Analytical Engine*, eram necessários três tipos de cartões: cartões de operação, que determinavam a natureza das operações (adição, subtração, multiplicação e divisão); cartões de variáveis, que definiam as variáveis a serem manipuladas; e cartões numéricos, que introduziam dados específicos de tabelas preparadas, como as de logaritmos e trigonometria. A máquina acessava esses cartões conforme necessário[10], durante a execução das operações algébricas, possibilitando tanto a tomada de decisões com base em resultados anteriores (seleção) quanto a repetição de partes do programa (*looping*). Além disso, a *Analytical Engine* foi projetada com unidades separadas para as diferentes funções de entrada, processamento, memória e saída.

Segundo Denning e Tedre (2019), Ada Lovelace (1815-1852), uma matemática

[10] A explicação detalhada do funcionamento das máquinas de Babbage foge do escopo deste trabalho. Ao leitor interessado, recomenda-se a leitura de Costa (2012).

inglesa que colaborou com Babbage no desenvolvimento de algoritmos para suas máquinas de computação, argumentou que essas máquinas poderiam ser mais do que simples calculadoras de números. Para ela, essas máquinas eram processadores de qualquer informação que pudesse ser codificada em símbolos.

Os projetos de máquinas de computação de Babbage e Lovelace introduziram muitas ideias que hoje são consideradas características distintivas do Pensamento Computacional. Os computadores modernos incorporaram muitos aspectos desses projetos, como a representação digital de dados, a programação, o *looping*, os algoritmos executáveis por máquina e as estruturas de controle para escolha de casos.

Todavia, os projetos lógicos de Babbage e Lovelace não puderam ser realizados com a tecnologia da época. Algumas décadas depois, o advento da era da eletrônica abriu novas possibilidades. Assim, a partir do final da década de 1930, houve intensa experimentação na construção de novos modelos de máquinas de computação, conforme será discutido na sequência.

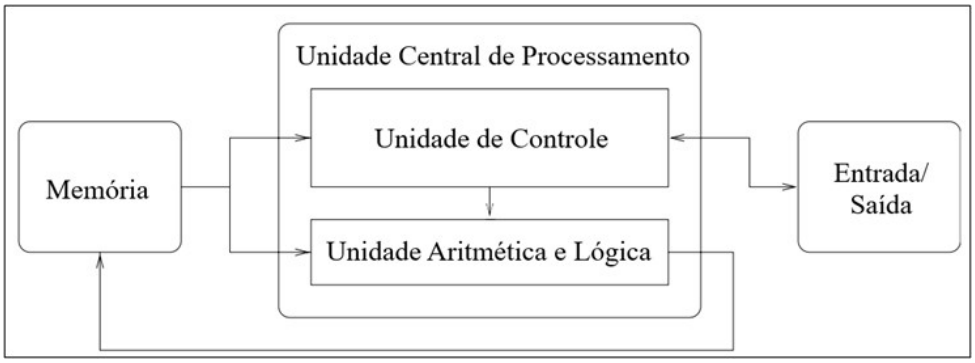
2.1.4 Pensamento Computacional na concepção dos computadores eletrônicos

Na subseção anterior, discute-se a evolução das máquinas de computação e a transição para o uso de software. Esse processo foi motivado pela necessidade de acelerar cálculos e eliminar erros humanos, destacando a engenhosidade de pioneiros como Babbage e Lovelace. Nesta subseção, analisa-se uma alteração profunda na concepção dos computadores eletrônicos na década de 1940, que causou uma revolução no Pensamento Computacional.

Em 1945, John von Neumann (1903-1957) projetou uma máquina de computação que se destacou por sua organização inovadora, introduzindo o conceito de "computador de programa armazenado", em que as instruções de operação são armazenadas na memória juntamente com os dados (Denning; Tedre, 2019). Esse conceito eliminou a necessidade de cartões perfurados ou fitas externas para carregar instruções, permitindo que o computador acessasse tanto os dados quanto as instruções diretamente da memória. Essa abordagem simplificou o processo de programação e execução, facilitando a modificação

e o armazenamento das instruções de forma similar aos dados. A Figura 2.4 ilustra um sistema de computador com arquitetura von Neumann.

Figura 2.4 – Componentes básicos de um computador com arquitetura von Neumann



Fonte: Adaptado de Eigenmann e Lilja (1998).

Um computador com arquitetura von Neumann divide-se em três subsistemas principais: a unidade central de processamento (CPU), a memória e a unidade de entrada e saída (Eigenmann; Lilja, 1998). A CPU executa operações aritméticas e lógicas e coordena a execução de cada instrução, buscando (lendo) dados de instruções na memória principal. Ela geralmente é composta pela Unidade de Aritmética e Lógica (UAL), que realiza operações como adição, subtração, AND e OR, e pela Unidade de Controle, responsável por interpretar as instruções e coordenar o funcionamento do sistema.

A memória consiste em células de armazenamento, cada uma em dois estados possíveis, representando os valores binários 0 e 1. Assim, cada célula pode armazenar um *bit*[11] de informação. A unidade de entrada e saída, por sua vez, conecta o sistema ao ambiente externo, permitindo a entrada de programas e dados, bem como a saída dos resultados do processamento.

Embora essa estruturação tenha facilitado o processamento de informações, a codificação e representação dos dados processáveis por computadores eletrônicos apresentaram novos desafios. A abstração tornou-se uma habilidade central nesse contexto, pois envolve a representação adequada das informações a serem processadas pelos computadores. Na sequência, exploram-se algumas decisões dos matemáticos e engenheiros para otimizar a construção de computadores, utilizando abstrações para torná-los mais eficientes e

[11] O termo *bit* é uma abreviação de "dígito binário".

confiáveis.

Denning e Tedre (2019) destacam que os engenheiros escolheram usar códigos binários para representar números, pois a aritmética em binário exigia menos componentes do que a codificada em decimal. Além disso, os circuitos que distinguem dois níveis de tensão eram mais confiáveis do que aqueles que distinguiam múltiplos níveis. Essa abstração em códigos binários levou ao desenvolvimento de circuitos e mídias de armazenamento mais simples e confiáveis. Nesse contexto,

> É importante ter em mente que, internamente, o computador não processa números e símbolos. Os circuitos do computador lidam apenas com tensões, correntes, interruptores e materiais maleáveis. Os padrões de zeros e uns são abstrações inventadas pelos projetistas para descrever o que seus circuitos fazem [...] (Denning; Tedre, 2019, p. 48, tradução nossa).[12]

Além da representação binária, a abstração de circuitos eletrônicos com o uso de álgebra booleana foi essencial para o avanço do software e do hardware[13] dos computadores modernos (Denning; Tedre, 2019). A utilização de símbolos matemáticos bem formulados, como variáveis com valores verdadeiros ou falsos, possibilitava resolver problemas de maneira quase automática.

Quase um século após as contribuições de George Boole (1815-1864) para a lógica, Claude Shannon (1916-2001) percebeu que a lógica booleana poderia ser aplicada para descrever a função dos circuitos de relé em sistemas telefônicos (Denning; Tedre, 2019). Shannon aperfeiçoou essa lógica para projetar circuitos eletrônicos, incluindo os de computadores. Nos componentes de hardware atuais, como portas lógicas e microprocessadores, a álgebra booleana é empregada para realizar operações lógicas fundamentais, essenciais ao processamento de dados. Assim, a habilidade de abstrair circuitos por meio da álgebra booleana tornou-se indispensável no projeto dos computadores modernos.

Na estrutura dos computadores modernos concebidos a partir da arquitetura von Neumann, os algoritmos também desempenham um papel central. Segundo Denning e

[12] "It is important to keep in mind that internally the computer does not process number and symbols. Computer circuits deal only with voltages, currents, switches, and malleable materials. The patterns of zeroes and ones are abstractions invented by designers to describe what their circuits do [...]" (Denning; Tedre, 2019, p. 48).

[13] Hardware é o conjunto de todos os componentes físicos de um computador.

Tedre (2019), esses sistemas consistem em registros que armazenam padrões de *bits* e circuitos lógicos responsáveis por computar funções baseadas nos dados armazenados nos registros. Para assegurar a estabilidade na propagação dos sinais dos registros de entrada para os de saída, os computadores requerem um relógio interno, que sincroniza operações como soma, subtração e comparação, garantindo a conclusão de cada etapa antes do início da próxima, prevenindo erros de processamento. Assim, a estrutura física dos computadores eletrônicos, com seus registros e circuitos lógicos, opera de maneira que segue processos lógicos no formato de algoritmos.

Ainda sobre algoritmos, Denning e Tedre (2019) argumentam que uma máquina deve ser capaz de decidir as próximas instruções, que nem sempre seguem uma sequência linear. Na arquitetura von Neumann, após a execução de uma instrução específica, o padrão é que a próxima instrução na sequência seja realizada (por exemplo, após a instrução K, segue-se a instrução $K+1$). Contudo, é comum haver desvios dessa sequencialidade, em que uma instrução é executada em uma localização de memória diferente, como X. A decisão de desviar é determinada por uma condição C (por exemplo, "A é igual a B?"). Essa ideia de desvios no programa, que também é uma característica dos algoritmos, é também refletida no Pensamento Computacional, manifestando-se na estrutura se-então-senão, amplamente utilizada em linguagens de programação modernas.

Outra capacidade geralmente associada ao Pensamento Computacional é a habilidade de conceber, organizar, construir, compreender e executar algoritmos que incluem estruturas de repetição como *while*, *for* e *foreach*. Essas estruturas possibilitam a realização de cálculos muito além do tamanho do programa inicialmente concebido. No entanto, um erro recorrente na programação é a ocorrência de "*loop* infinito", em que uma condição de parada defeituosa impede a saída do *loop*. Alan Turing demonstrou que não há algoritmo capaz de verificar se algum dos *loops* em um programa é infinito.

Nesse contexto, Denning e Tedre (2019) argumentam que a depuração, uma das dimensões do Pensamento Computacional defendidas por alguns autores[14], é uma atividade que não pode ser completamente automatizada, sendo essencialmente uma habilidade

[14] A noção de depuração será mais bem detalhada adiante, em outra seção.

humana, conforme Espadeiro (2021), Dantas (2023) e Teixeira, Juvanelli e Dantas (2024) defendem em seus trabalhos.

A partir do exposto, pode-se inferir que o Pensamento Computacional não é uma ideia oriunda da Ciência da Computação, como amplamente divulgado na literatura dedicada ao tema (Wing, 2006; Brackmann, 2017). Estas considerações iniciais visam destacar que, pelo menos até meados do século XX, o Pensamento Computacional estava integrado aos conceitos, práticas e habilidades de diversos campos de conhecimento que têm acompanhado a evolução humana desde seus primórdios. Denning e Tedre (2019) atestam que

> Essa discussão sobre as origens do Pensamento Computacional deve deixar claro que o PC não se refere à forma como os cientistas da computação pensam. A ciência da computação moderna é o último 1% da linha do tempo histórica do Pensamento Computacional. Os cientistas da computação herdaram e depois aperfeiçoaram o Pensamento Computacional de uma longa linhagem de matemáticos, filósofos naturais, cientistas e engenheiros, todos interessados em realizar grandes cálculos e inferências complexas sem erros. O PC é uma característica de muitos campos, não apenas da computação (Denning; Tedre, 2019, p. 41, tradução nossa).[15]

Na sequência, com o objetivo de sintetizar as habilidades relacionadas ao Pensamento Computacional ao longo do tempo, discorre-se brevemente sobre os pontos centrais discutidos nesta seção.

2.1.5 Considerações sobre possíveis raízes históricas do Pensamento Computacional

Esta seção teve como objetivo apresentar e discutir a evolução do Pensamento Computacional desde métodos que remontam à civilização grega até a origem dos computadores eletrônicos modernos, destacando as habilidades centrais desse modo de pensar ao longo do tempo. A obra de Denning e Tedre (2019) foi utilizada como referência central, por sua análise abrangente sobre o desenvolvimento da computação e as transformações

[15] "It should be clear from this discussion of the origins of computational thinking that CT is not about how scientists think. Modern computer science is the last 1 percent of the historical timeline of computational thinking. Computer scientists inherited and then perfected computational thinking from a long line of mathematicians, natural philosophers, scientists, and engineers all interested in performing large calculations and complex inferences without error. CT is a feature of many fields, not only computing" (Denning; Tedre, 2019, p. 41).

do Pensamento Computacional ao longo da história.

Na Antiguidade, Arquimedes aperfeiçoou o Método da Exaustão para aproximar a área de um círculo, evidenciando a habilidade de decomposição ao dividir o problema em partes menores, usando polígonos inscritos e circunscritos. O reconhecimento de padrões se manifesta ao identificar que o aumento do número de lados dos polígonos resulta em maior precisão na aproximação. A abstração se releva ao simplificar a complexidade da área do círculo, focando nas propriedades geométricas dos polígonos; e a formulação do método em um algoritmo sistemático reflete características do Pensamento Computacional.

Com o advento do Cálculo, Leibniz introduziu o uso de séries infinitas para a aproximação de curvas e determinação de áreas sob essas curvas. Posteriormente, a integral definida, formalizada por Cauchy e Riemann por meio da Soma de Riemann, passou a ser organizada como um algoritmo, evidenciando a importância dos procedimentos sistemáticos e as habilidades de abstração, decomposição e reconhecimento de padrões.

Entre os séculos XVII e XIX, a abstração desempenhou papel central no desenvolvimento de máquinas de computação, como a régua de cálculo e a calculadora aritmética. Nesse ínterim, Babbage e Lovelace avançaram o conceito de máquinas programáveis por software, utilizando cartões perfurados para instruções e destacando a abstração de operações matemáticas em mecanismos físicos. Essas inovações introduziram conceitos fundamentais para o Pensamento Computacional.

A arquitetura von Neumann, projetada em 1945, revolucionou a computação ao estruturar o computador em três subsistemas principais: CPU, memória e unidade de entrada e saída, aumentando a eficiência e a flexibilidade no processamento de informações. A abstração desempenhou um papel fundamental ao possibilitar a representação e manipulação de dados em códigos binários e álgebra booleana, simplificando o design de circuitos eletrônicos. A implementação de algoritmos e estruturas de controle, incluindo sincronização de operações e mecanismos para desvios condicionais e *loops*, reflete as características centrais do Pensamento Computacional.

Na seção seguinte, discute-se como, a partir da década de 1940, o Pensamento Computacional ascendeu como um elemento central em uma nova profissão e campo do

conhecimento, transformando paradigmas científicos e abrindo caminho para a expansão da computação.

2.2 O Pensamento Computacional após o advento dos computadores eletrônicos

Conforme argumentado na seção anterior, o Pensamento Computacional não precisa ser considerado um modo de pensar exclusivo ou derivado da Ciência da Computação. Contudo, apesar de auxiliar na formulação, resolução e análise de problemas de diversas áreas do conhecimento, até o final da década de 1940 não havia a preocupação em formalizar e inserir essas ideias no currículo das escolas e universidades. Também não se discutia objetivamente sobre as habilidades envolvidas no Pensamento Computacional e as consequências educacionais do ensino de computação.

Um ponto de inflexão ocorreu na década de 1950, com o fortalecimento do estudo dos fenômenos relacionados aos computadores. Desse modo, esta seção visa discutir sobre como as habilidades necessárias para os cientistas da computação e engenheiros de software sofreram alterações e aprimoramentos ao longo das décadas, transformando o Pensamento Computacional em um elemento fundamental para a ciência. Este contexto permite evidenciar como essas transformações influenciaram a inserção da computação e do Pensamento Computacional no âmbito educacional, por meio de perspectivas teóricas cada vez mais aprofundadas nas questões epistemológicas que o uso dos computadores poderia ter nos processos educacionais. Para embasar teoricamente as problematizações propostas nesta seção, são articuladas principalmente as ideias de Denning e Tedre (2019), Raabe, Couto e Blikstein (2020) e Vieira, Campos e Raabe (2020).

Denning e Tedre (2019) discutem o avanço da computação e como o Pensamento Computacional sofreu alterações ao longo das décadas, passando de um modo de pensar dos matemáticos e engenheiros para se tornar fundamental para a ciência e o avanço em diversas áreas do conhecimento. Os autores abordam o desenvolvimento e as mudanças do Pensamento Computacional, de modo que as discussões serão direcionadas a partir de cinco perspectivas: a fundação da Ciência da Computação como área do conhecimento

em meados da década de 1950, a crise na engenharia de software nos anos de 1960, a ascensão do design como uma capacidade fundamental para os desenvolvedores de software na década de 1970, a consolidação da Ciência da Computação e mudança de paradigmas na ciência a partir de 1980 e, por fim, a disseminação global da computação na década de 1990.

Raabe, Couto e Blikstein (2020) permitem compreender as primeiras abordagens para a inserção da computação na Educação Básica, em meados da década de 1960. Os autores realizam apontamentos sobre as posições epistemológicas do uso dos computadores nos processos educacionais, ou seja, os valores, crenças e objetivos que influenciaram cada uma das abordagens de inserção da computação na Educação Básica.

Em Vieira, Campos e Raabe (2020) discute-se o construcionismo de Seymour Papert, um dos pioneiros a conceber uma fundamentação teórica robusta para o uso de computadores no processo de aprendizagem, seguindo uma perspectiva piagetiana. Na perspectiva de Papert (1980), o computador seria uma ferramenta poderosa para a construção de conhecimento por meio do aprender fazendo e do refletir sobre o que se está fazendo.

Desse modo, os trabalhos de Denning e Tedre (2019), Raabe, Couto e Blikstein (2020) e Vieira, Campos e Raabe (2020) foram selecionados para embasar esta seção devido à sua abrangência e profundidade na análise histórica e epistemológica do Pensamento Computacional, permitindo uma compreensão ampla e crítica das transformações e repercussões do Pensamento Computacional no campo educacional e científico.

Na sequência, iniciam-se as problematizações sobre a fundação da Ciência da Computação na década de 1950.

2.2.1 Origem da Ciência da Computação

À medida que houve o fortalecimento do estudo dos fenômenos que envolvem os computadores na década de 1950, acadêmicos começaram a defender a formação de programas de Ciência da Computação nas universidades para atender a demanda de formação de profissionais para lidar com as novas tecnologias (Denning; Tedre, 2019).

Apesar de haver uma preocupação com aspectos mercadológicos e a preparação de pessoas capazes de manusear os computadores, Denning e Tedre (2019) argumentam que, mesmo naquela época, alguns educadores de computação viam a computação como uma ferramenta de pensamento para o aprendizado e para lidar com problemas e questões em muitos campos além da Ciência da Computação.

Contudo, os cientistas da computação pioneiros enfrentaram resistências dos outros departamentos nas universidades, que não consideravam a computação como uma ciência ou engenharia, nem acreditavam que ela proporcionasse uma perspectiva intelectual única. Os críticos argumentavam que a computação carecia de conteúdo intelectual exclusivo e de uma base teórica adequada.

Para superar essa resistência, os pioneiros da computação se esforçaram para articular uma identidade única e justificar seu campo, a Ciência da Computação. Havia uma dúvida profunda sobre se a computação tinha substância acadêmica além da matemática, da engenharia elétrica e da física (Denning; Tedre, 2019).

Os pioneiros da Ciência da Computação delinearam uma perspectiva de Pensamento Computacional, concentrada na construção de programas, máquinas de computação e sistemas operacionais. Desenvolveram muitos conceitos de computação, como variáveis nomeadas, estruturas de controle, tipos de dados, linguagens de programação formais, compiladores e interfaces. Denning e Tedre (2019) afirmam que a metodologia de programação e a arquitetura de sistemas de computadores, ou seja, as formas características de pensar e praticar computação, foram os principais impulsionadores do desenvolvimento do Pensamento Computacional na década de 1950.

Entretanto, a programação na década de 1950 era considerada um “trabalho solitário”, envolvendo apenas um problema, um programador, um computador e uma pequena biblioteca de sub-rotinas (Denning; Tedre, 2019). A maioria dos programadores se concentrava em programas para uso pessoal ou imediato do grupo de trabalho, mas não em programas para uso fora da organização. Segundo os autores, o Pensamento Computacional dessa década era rico, mas fragmentado, focando em fazer com que programas únicos funcionassem em máquinas específicas.

2.2.2 Crise entre os desenvolvedores de software e habilidades de decomposição

Na década de 1960, à medida que a demanda por computadores e programas crescia, houve uma crise entre os desenvolvedores de software. Grandes empresas buscavam máquinas de computação para automatizar processos, mas havia uma escassez de profissionais capacitados para lidar com as complexidades do desenvolvimento de softwares. Nesse contexto, muitos projetos de software eram entregues com atraso, erros, falhas de segurança ou causavam prejuízos significativos (Denning; Tedre, 2019).

Programas que antes eram desenvolvidos por uma pessoa passaram a exigir equipes com dezenas de programadores. Os engenheiros de software perceberam que suas habilidades de Pensamento Computacional não eram adequadas para esse novo contexto. Havia uma diferença qualitativa entre um programa escrito por um único programador e um sistema que exigia uma equipe com muitos programadores (Denning; Tedre, 2019). As ferramentas para programação em pequena escala não eram adequadas para a programação em grande escala.

Denning e Tedre (2019) afirmam que as habilidades necessárias para escrever um programa de mil linhas de código são diferentes das necessárias para criar um software de um milhão de linhas. O principal motivo é que os sistemas de software de grande porte precisam ser criados por equipes. Assim, pode-se inferir que uma característica marcante do Pensamento Computacional nesta década foi a capacidade de decomposição de problemas, além da organização e gerenciamento de equipes, para desenvolver softwares com êxito.

Ainda na década de 1960, houve a primeira tentativa de levar a computação para as escolas de ensino fundamental e médio. Denning e Tedre (2019) afirmam que essa experiência foi complemente diferente de levar o ensino de computação para as universidades. Nessa primeira investida, o foco era no ensino do funcionamento das máquinas e de linguagens de programação, sem uma preocupação específica com a educação, mas com aspectos mercadológicos e a preparação de mão de obra qualificada para as empresas de tecnologia (Raabe; Couto; Blikstein, 2020). Contudo, embora os computadores tivessem se tornado mais comuns, os educadores, por falta de recursos financeiros e esforços políticos, não conseguiram implementar o ensino de computação em larga escala nas escolas.

Em 1967, Seymour Papert (1928-2016) e seus colaboradores criaram a primeira versão da linguagem Logo, parte de uma estrutura integrada de ideias pedagógicas, tecnológicas e educacionais (Denning; Tedre, 2019). Nas décadas seguintes, Papert foi considerado por teóricos, professores e estudantes um dos pioneiros na argumentação sobre o uso de computadores na aprendizagem, devido à sua compreensão fundamentada de como as crianças aprendem. No entanto, de acordo com Vieira, Campos e Raabe (2020), apesar de acreditarem que o uso do computador poderia trazer benefícios à educação, Papert e sua equipe iniciaram pesquisas para fundamentar teoricamente esses benefícios, explorando novas maneiras de usar o computador na aprendizagem[16].

2.2.3 Design e o pensamento algorítmico de Knuth

Na década de 1970, os engenheiros de software buscaram tornar a qualidade dos softwares mensurável, tornando isso um ponto central do Pensamento Computacional na engenharia de software. Entre os 20 fatores mensuráveis para avaliar a qualidade geral de um sistema de software estavam eficiência, flexibilidade e funcionalidade. No entanto, segurança e proteção não estavam presentes na lista, pois, segundo Denning e Tedre (2019), não se sabia como medir os softwares em relação a esses aspectos.

Um ponto que ganhou destaque a partir desta década, principalmente devido aos esforços de George Forsythe (1917-1972), um dos pioneiros na defesa da Computação como área do conhecimento, foram as questões relacionadas ao design (Denning; Tedre, 2019). Isto incluiu o design de computadores e sistemas, de linguagens para processadores e algoritmos, e de métodos para representar e processar informações.

No entanto, Denning e Tedre (2019) argumentam que o design vai além da construção de sistemas. Para os autores, é um processo de criação e modelagem de artefatos que atendem às preocupações humanas. Especificamente para softwares, design significa criar um software que realize as tarefas que os usuários desejam. Os designers de software, nessa perspectiva, fazem mais do que construir para atender às especificações funcionais,

[16] Embora Papert e sua equipe tenham realizado diversas pesquisas visando fundamentar teoricamente os benefícios do uso do computador na aprendizagem, não conseguiram apresentar uma justificativa consistente para tais benefícios, conforme será discutido nas próximas seções.

eles apoiam intencionalmente as práticas, os mundos, os contextos e as identidades dos usuários do software. Contudo, Denning e Tedre (2019) atestam que

> O PC de engenharia de software é especialmente útil para grandes sistemas que precisam ter um desempenho confiável em ambientes críticos de segurança. As pessoas querem sistemas de controle de tráfego aéreo, sistemas de controle de usinas nucleares e veículos espaciais de Marte cuidadosamente projetados. O PC de design é especialmente útil para softwares que precisam se adequar às comunidades de clientes, facilitar a adoção e oferecer grande valor. O PC de design não abandona o PC de engenharia de software; ele fica atento às oportunidades de incluir funções interessantes que os clientes ainda não solicitaram (Denning; Tedre, 2019, p. 111, tradução nossa).[17]

Dessa forma, pode-se inferir que o Pensamento Computacional nesta década foi marcado pela incorporação do design no corpo de conhecimento da engenharia de software. Além da compreensão das regras estruturais das linguagens de programação, da capacidade de trabalho em equipe e de depuração, as habilidades de design tornaram-se essenciais para os engenheiros de software. Isso inclui a capacidade de produzir um objeto funcional para as pessoas, em um contexto de valores e necessidades, proporcionando resultados de qualidade e uma experiência satisfatória (Denning; Tedre, 2019).

Em meados da década de 1970, surgiram novas iniciativas para inserir a computação no currículo das escolas de ensino fundamental e médio. Além das contribuições de Papert, Donald Knuth foi um dos pioneiros a discutir os efeitos educacionais do conhecimento na área da Ciência da Computação. Sua perspectiva era focada no pensamento algorítmico, argumentando que os algoritmos seriam o núcleo central dessa área (Knuth, 1974).

Em seu artigo publicado em 1974, *Computer Science and its relation to mathematics*, Knuth (1974) define um algoritmo como uma sequência de regras definida com precisão que informa como produzir uma informação de saída específica a partir de uma determinada informação de entrada em um número finito de etapas. Nessa perspectiva, uma pessoa com conhecimento profundo sobre algoritmos estaria preparada para mais do que escrever programas de computador. Esse conhecimento seria uma ferramenta mental de uso

[17] "Software engineering CT is especially useful for large systems that must perform reliably in safety critical environments. People want carefully engineered air traffic control systems, nuclear plant control systems, and Mars rovers. Design CT is especially useful for software that must fit customer communities, facilitate adoption, and deliver great value. Design CT does not abandon software engineering CT; it listens for opportunities to include delightful functions customers have not yet asked for" (Denning; Tedre, 2019, p. 111).

geral, auxiliando na compreensão de outros assuntos, como química, linguística e música. Essa ideia de transferência das habilidades de computação para a solução de problemas em geral relaciona-se com perspectivas posteriores de Pensamento Computacional e será abordada mais adiante.

Ainda na década de 1970, Papert e sua equipe avançaram nas discussões sobre a importância da computação na esfera educacional. Baseados nos experimentos com Logo e fundamentados nas teorias de Jean Piaget (1896-1980), o construcionismo de Papert se originou como uma vertente do construtivismo, em que a aprendizagem ainda se fundamentava na construção de conhecimento, mas o estudante construiria seu conhecimento a partir do "fazer", criando objetos concretos e compartilháveis (Raabe; Couto; Blikstein, 2020).

De acordo com Vieira, Campos e Raabe (2020), os aprimoramentos realizados na linguagem Logo nesta década foram direcionadas pela ideia de que o indivíduo não precisava ser especialista em programação para utilizar a linguagem. A proposta do uso dessa linguagem era que qualquer pessoa com um mínimo de iniciação pudesse utilizá-la, incluindo crianças. Contudo, a linguagem Logo e o construcionismo ganhariam força somente na década seguinte.

2.2.4 Ciências computacionais e o construcionismo de Papert

Na década de 1980, houve a consolidação da Ciência da Computação como campo do conhecimento, marcada pelo desenvolvimento de supercomputadores. Esses computadores possuíam poder de computação suficiente para resolver problemas de diversas áreas do conhecimento por meio de simulações (Denning; Tedre, 2019). Devido à complexidade dos cálculos envolvidos, os supercomputadores tornaram-se essenciais para as ciências. Somente esses computadores podiam resolver numericamente equações diferenciais complexas, que modelavam muitos dos problemas da época.

Nesse contexto, Denning e Tedre (2019) destacam que tantos cientistas experimentais, que reúnem dados para explorar e isolar fenômenos, quanto teóricos, que utilizam modelos matemáticos para fazer previsões, viram novas oportunidades ao usar computado-

res como simuladores. Isso permitiu uma nova abordagem científica baseada na exploração e na modelagem computacional.

> Durante séculos, a teoria e o experimento foram os dois modos de fazer ciência. Os supercomputadores mudaram isso, abrindo uma nova abordagem para fazer ciência com base na exploração e modelagem computacional. Foi a mudança de paradigma científico mais significativa desde a mecânica quântica. A revolução da ciência computacional deu início a uma nova onda de pensamento computacional (Denning; Tedre, 2019, p. 115, tradução nossa).[18]

A expressão "ciência computacional" entrou em uso em meados da década de 1980 (Denning; Tedre, 2019). Refere-se aos ramos de cada campo científico que usam a computação, como a física computacional, a bioinformática e a sociologia computacional. A computação mostrou ser produtiva para o avanço da ciência e da engenharia, levando muitos campos científicos a desenvolverem um ramo "computacional". Em muitos deles, esses ramos se tornaram fundamentais para suas áreas.

A revolução da ciência computacional iniciou uma nova onda de Pensamento Computacional. Essa habilidade tornou-se central e indispensável não apenas para os engenheiros de software, mas também para cientistas e pesquisadores. As principais características do Pensamento Computacional para os cientistas computacionais nesta década incluem habilidades de modelagem de processos físicos, implementação dessas simulações por meio de algoritmos e avaliação dos resultados quanto à eficiência, usabilidade e ausência de erros (Denning; Tedre, 2019).

As discussões sobre a inserção da computação e do Pensamento Computacional no âmbito educacional, incipientes até a década de 1970, tiveram novos desdobramentos a partir da década de 1980. A expressão "Pensamento Computacional", conforme é utilizada atualmente, possivelmente foi apresentada pela primeira vez em 1980 no livro *Mindstorms: children, computers, and powerful ideas* de Papert, que consolidava mais de uma década de pesquisas sobre o uso de computadores no processo de aprendizagem de crianças (Papert, 1980). Desse modo, embora os conceitos, práticas e habilidades do Pensamento

[18] "For centuries, theory and experiment were the two modes of doing science. Supercomputers changed this, opening a new approach to doing science based on computational exploration and modeling. It was the most significant scientific paradigm shift since quantum mechanics. The computational science revolution ushered in a new wave of computational thinking" (Denning; Tedre, 2019, p. 115).

Computacional possam ser identificados até em civilizações antigas, foi Papert quem primeiro utilizou essa expressão.

Durante a década de 1980, a linguagem Logo e o construcionismo de Papert ganharam força nos espaços educativos, impulsionados pelo avanço dos computadores e pela expansão das versões Logo como linguagem e filosofia (Vieira; Campos; Raabe, 2020). Tedre e Denning (2016) argumentam que Papert e seus colaboradores avançaram significativamente na direção de formalizar e aprofundar as discussões sobre como as crianças aprendem com o uso do computador. No entanto, apesar das discussões de Papert (1980) se relacionarem com vários aspectos do Pensamento Computacional, o autor preferiu as expressões "pensamento procedimental" e "pensamento combinatório".

Após a consolidação do construcionismo como fundamentação teórica para construção de conhecimento com o uso de computadores, é relevante apresentar os principais pressupostos epistemológicos dessa perspectiva, considerada a primeira abordagem fundamentada para a inserção da computação na Educação Básica (Raabe; Couto; Blikstein, 2020).

Partindo de uma perspectiva piagetiana, o construcionismo postula que o conhecimento não é transmitido passivamente do educador para o aluno, mas é ativamente construído pelo próprio aprendiz (Vieira; Campos; Raabe, 2020). Nessa perspectiva, o aprendizado ocorre predominantemente por meio da prática e da experimentação. O princípio do "aprender fazendo" é central, indicando que os alunos constroem o conhecimento quando estão diretamente envolvidos na criação e manipulação de objetos de aprendizagem.

Os objetos computacionais, situados entre a ideia e um objeto físico, oferecem novas possibilidades para que os alunos explorem fenômenos, criando seus próprios modelos por meio de experimentações (Vieira; Campos; Raabe, 2020), Assim, os computadores são tratados como um meio para o desenvolvimento da aprendizagem e, consequentemente, para a construção de conhecimento. A abordagem construcionista aborda os conceitos de computação como um tema transversal, que pode ser trabalhado de diversas formas e em várias áreas do conhecimento. Sobre esse aspecto, Raabe, Couto e Blikstein (2020) apontam que

> O computador é visto como uma ferramenta para aprender com e, portanto, enfatiza-se programar para aprender. Nessa abordagem, os fundamentos de computação e programação não são necessariamente ensinados como o conteúdo principal, pois o foco está em possibilitar que os estudantes se tornem fluentes em criar inovação e tecnologia com uso da computação e como forma de aprender em qualquer disciplina [...] (Raabe; Couto; Blikstein, 2020, p. 13).

Raabe, Couto e Blikstein (2020) elucidam alguns aspectos que são centrais na discussão em torno do Pensamento Computacional. Os autores defendem que a formação inicial de professores, nessa perspectiva, deve ser embasada em conceitos de computação, modelagem científica e programação, permitindo que esses aspectos sejam trabalhos em suas áreas ou de forma interdisciplinar. É possível inferir que essa perspectiva se articula com as características marcantes do Pensamento Computacional, principalmente das décadas de 1950 e 1980, conforme discutido anteriormente.

Tedre e Denning (2016) argumentam que Papert (1980) afirmava que a prática de programação desenvolve habilidades cognitivas que aumentam as habilidades de resolução de problemas dos alunos em outros domínios, promovendo uma mudança de "aprender a programar" para "programar para aprender". A ideia de transferência de conhecimento para outras áreas começou a receber destaque, mesmo sem evidências empíricas, com críticos apontando que estudos em ciências cognitivas não apoiavam essas afirmações.

Os críticos argumentam que a própria aprendizagem de programação depende de habilidades matemáticas, raciocínio lógico, raciocínio condicional, capacidade de memória e habilidades de pensamento processual (Tedre; Denning, 2016). Tedre e Denning (2016) concluem que possuir habilidades computacionais pode auxiliar em disciplinas como engenharia ou matemática e transformar a maneira como o indivíduo lida com problemas nesses domínios, mas isso não deve ser considerado uma transferência, e sim uma aplicação direta da computação em diferentes áreas.

Contudo, mesmo com os resultados contrários a partir da década de 1980, conforme apontam Tedre e Denning (2016), muitos pesquisadores do Pensamento Computacional continuaram disseminando a ideia de que ele melhora as competências cognitivas em todos os domínios do conhecimento, situação que será discutida em uma seção posterior.

2.2.5 Disseminação global da computação e o Letramento Computacional de diSessa

Durante a década de 1990, o ensino de computação começou a ser difundido nas escolas de ensino fundamental e médio, mas as discussões em torno do Pensamento Computacional ainda eram, em sua maioria, domínio das universidades. Segundo Denning e Tedre (2019), as escolas pré-universitárias ofereciam vários cursos de informática, focando principalmente na alfabetização digital e, em menor escala, na programação. Os autores reiteram que os cursos de alfabetização digital introduzidos naquela época eram considerados decepcionantes do ponto de vista do Pensamento Computacional, pois se concentravam no uso de ferramentas como processadores de texto e planilhas eletrônicas, em vez de explorar os processos do Pensamento Computacional.

Essa abordagem para a inclusão da computação na Educação Básica se relaciona com uma das perspectivas elencadas por Raabe, Couto e Blikstein (2020). Os autores argumentam que nessa perspectiva

> [...] a demanda não será necessariamente por programadores profissionais, mas sim por profissionais que terão que usar a computação e a programação para automatizar planilhas, fazer consultas de programação, realizar acesso *on-line* a bancos de dados, usar ferramentas de *software* de mineração de dados e operar dispositivos de computação física em arte interativa ou automação residencial (Raabe; Couto; Blikstein, 2020, p. 10, grifos do autor).

Apesar das críticas de acadêmicos e educadores por não abordar as discussões conceituais sobre computação ou Pensamento Computacional, essa abordagem recebeu significativo apoio financeiro, com investimentos públicos e privados massivos (Valente et al., 2017; Raabe; Couto; Blikstein, 2020). Focando em aspectos mercadológicos e de empregabilidade, o ensino de programação foi promovido de maneira rápida e intensiva, desconsiderando as complexidades de implementar o ensino de computação para crianças. Segundo Valente et al. (2017), isso reflete um embate filosófico sobre os propósitos da escola: formar cidadãos com conhecimento ou trabalhadores com habilidades pontuais.

Raabe, Couto e Blikstein (2020) argumentam que, nessa perspectiva, o interesse não reside na formação de professores com curso de Ensino Superior, mas em profissionais

do mercado de tecnologia com notório saber para atuar na docência. Dessa forma, o aprendizado se resume a cursos técnicos profissionalizantes, enfatizando programação e codificação em detrimento dos processos do Pensamento Computacional (Raabe; Couto; Blikstein, 2020).

Tedre e Denning (2016) apontam que esses movimentos na educação refletiram mudanças aceleradas na informatização da sociedade, mas não provocaram alterações na epistemologia geral e no método científico. Embora tenha havido uma mudança na percepção dos computadores por cientistas e pesquisadores, essa revolução não impactou, em termos metodológicos e epistemológicos, a implementação na Educação. Apesar de sua atração, esse movimento não produziu uma mudança generalizada no ensino de computação nas escolas de ensino fundamental e médio.

No final da década de 1990, à medida que a Internet se tornava uma mercadoria doméstica, emergiu um novo movimento educacional que favorecia a fluência nas tecnologias da informação em detrimento da alfabetização digital. A expressão "Letramento Computacional", cunhada por Andrea diSessa (DiSessa, 2000), impulsionou uma nova posição epistemológica para o uso dos computadores, similar ao construcionismo de Seymour Papert.

DiSessa (2000) parte do pressuposto de que diferentes mídias apresentam diversas propriedades expressivas e facilitam novas formas de pensar. Por exemplo, o modo de pensar ao escrever pode diferir o modo de pensar ao falar, desenhar, pintar ou programar. Nessa perspectiva, os computadores oferecem novas formas de interação com o conhecimento e o pensamento. Assim, o Letramento Computacional significa utilizar recursos computacionais para externalizar mecanismos mentais, melhorando a habilidade de representar o mundo, de lembrar e de raciocinar sobre ele (Valente, 2019).

De acordo com DiSessa (2000), o Letramento Computacional está fundamentado em três pilares: o material, o mental ou cognitivo, e o social. Esses pilares se interrelacionam para formar uma compreensão robusta do Letramento Computacional. Os pilares reconhecem a importância não apenas dos computadores em si, mas também de como os indivíduos interagem com eles cognitivamente e dentro de um contexto social, promovendo

uma compreensão mais profunda dos computadores.

Em síntese, para DiSessa (2000), os computadores são considerados um meio para desenvolver a aprendizagem, sendo ferramentas (material) para a exploração de ideias. Essas ferramentas potencializam as capacidades mentais dos indivíduos (cognitivo), e a adoção e utilização dessas ferramentas são moldadas pelas práticas sociais (social). Portanto, na visão de DiSessa (2000), o Letramento Computacional na educação não deve abordar apenas o domínio técnico dos computadores, mas também o desenvolvimento de habilidades cognitivas e a compreensão do papel social da computação. Essa abordagem mais ampla permite que os indivíduos utilizem a computação de maneira mais significativa, crítica e contextualizada, indo além do uso instrumental.

Para tornar o Letramento Computacional possível, DiSessa (2000) defende que deve haver uma mudança na infraestrutura das escolas, de modo que os estudantes utilizem o computador constantemente durante o processo de aprendizagem, e não somente em "aulas de computador" (Raabe; Couto; Blikstein, 2020). Assim como o construcionismo, essa perspectiva vê a computação de forma transversal ao currículo escolar. Portanto, os fundamentos de computação e programação não são necessariamente ensinados como conteúdo principal, mas como uma forma de possibilitar que os estudantes se tornem fluentes em criar inovação e tecnologia com o uso da computação, aplicando-a em diversas disciplinas (Raabe; Couto; Blikstein, 2020). No entanto, apesar do referencial teórico robusto, Valente (2019) aponta que diSessa não operacionaliza como o Letramento Computacional pode ser incorporado nas diferentes disciplinas ou nas diferentes fases da Educação Básica.

Apesar de não haver um motivo único para o Letramento Computacional não ter se consolidado como uma abordagem central no ensino de computação, alguns fatores podem ter contribuído para limitar seu impacto. O Letramento Computacional prioriza o desenvolvimento de habilidades críticas e reflexivas no uso de tecnologias, enfatizando a compreensão dos processos internos e das implicações sociais da computação. Essa ênfase, no entanto, pode ter dificultado sua inserção em currículos escolares, uma vez que muitas instituições privadas e governamentais priorizam abordagens mais voltadas aos aspectos mercadológicos (Valente et al., 2017).

Esse contexto é especialmente relevante diante das demandas atuais por competências práticas e aplicáveis ao mercado de trabalho, que exigem que profissionais utilizem a computação e programação para uma variedade de tarefas, como automação de planilhas, e consultas programáticas a bancos de dados (Raabe; Couto; Blikstein, 2020). A proposta de DiSessa (2000), voltada para uma abordagem mais ampla e reflexiva, teve menos aderência frente a essas exigências práticas, favorecendo, em contraste, abordagens como o Pensamento Computacional, conforme será explorado na sequência.

A seguir, com a finalidade de evidenciar as principais habilidades relacionadas ao Pensamento Computacional no período entre as décadas de 1950 e 1990, discorre-se brevemente sobre os pontos centrais discutidos nesta seção.

2.2.6 Considerações sobre o Pensamento Computacional após o advento dos computadores

Esta seção teve como objetivo evidenciar como o Pensamento Computacional se tornou essencial para cientistas e engenheiros em diversas áreas, promovendo mudanças de paradigma nas ciências. Na sequência, apresentam-se as principais características do Pensamento Computacional e da computação em cada década, bem como as iniciativas de sua inserção na Educação Básica.

Na década de 1950, as habilidades fundamentais do Pensamento Computacional incluíam a criação e manipulação de variáveis, estruturas de controle, linguagens de programação, compiladores e interfaces. Nesse momento, a ênfase estava na construção de programas e sistemas operacionais, com foco na funcionalidade de softwares específicos para determinadas máquinas.

Na década de 1960, o crescimento dos projetos de software demandou habilidades organizacionais, especialmente a decomposição de problemas, evidenciando sua importância como uma habilidade essencial. Nesse ínterim, surgiram as primeiras iniciativas de ensino de computação, inicialmente focadas no funcionamento das máquinas e nas linguagens de programação, impulsionadas por interesses mercadológicos. Foi nesse período que Seymour Papert introduziu a linguagem Logo e explorou o uso de computadores na Educação,

marcando o início do desenvolvimento do construcionismo.

Durante a década de 1970, o foco do Pensamento Computacional passou para a qualidade do software. Engenheiros de software desenvolveram modelos de avaliação de atributos como eficiência, flexibilidade e funcionalidade, tornando a mensurabilidade da qualidade um aspecto central do Pensamento Computacional aplicado à engenharia de software.

Nos anos de 1980, o desenvolvimento de supercomputadores permitiu a resolução de problemas complexos por meio de simulações e modelagens computacionais. Essa expansão resultou na criação das Ciências Computacionais, como a física computacional e a bioinformática. Na esfera educacional, o construcionismo, consolidado por Papert, enfatizou a importância do "aprender fazendo" e da criação de objetos computacionais como parte do processo de aprendizagem, com a linguagem Logo e o construcionismo reforçando o papel dos computadores na educação infantil.

A década de 1990 foi caracterizada pela expansão da educação em computação nas escolas de Ensino Fundamental e Médio, com um foco predominante na alfabetização digital, por meio de cursos voltados para o uso de ferramentas como processadores de texto e planilhas eletrônicas. Nesse contexto, DiSessa (2000) introduziu o conceito "Letramento Computacional", ressaltando a importância dos recursos computacionais no desenvolvimento das habilidades cognitivas e sociais.

As discussões desta seção ilustram como a percepção dos computadores evoluiu entre cientistas e pesquisadores. Contudo, apesar das mudanças teóricas e dos avanços tecnológicos, a revolução computacional não resultou, em termos metodológicos e epistemológicos, na implementação ampla e efetiva da computação e do Pensamento Computacional na Educação Básica. Embora conceitos como o construcionismo de Papert e o Letramento Computacional de diSessa tenham trazido contribuições significativas, eles não foram suficientes para consolidar as ideias do Pensamento Computacional.

A busca por uma integração mais efetiva da computação nos sistemas escolares continuou entre educadores da área. Um ponto de inflexão ocorreu em 2006, com a publicação do ensaio *Computational Thinking* por Jeannette Wing (Wing, 2006), que redefiniu a

trajetória do Letramento Computacional em direção ao Pensamento Computacional. Esse ensaio impulsionou a disseminação global das ideias de Wing e de outros pesquisadores, reformulando o campo e reforçando a importância do Pensamento Computacional.

Na sequência, discorre-se sobre perspectivas de Pensamento Computacional que surgiram a partir da popularização dessa expressão após a publicação do ensaio de Wing.

2.3 Ascenção e perspectivas de Pensamento Computacional

Conforme tratado na seção anterior, o movimento em torno da inserção do Pensamento Computacional na Educação Básica estava incipiente até meados da década de 2000. A única perspectiva de Computação que conseguiu penetrar com profundidade no âmbito educacional, recebendo políticas governamentais e investimentos de empresas públicas e privadas, foi a mercadológica e de mão de obra, com foco na programação, sem preocupação com o aspecto educacional.

Um ponto de inflexão ocorreu em 2006, quando a pesquisadora Jeannette Marie Wing, na época influente na *National Science Foundation*[19] (NSF), publicou o ensaio *Computational Thinking* (Wing, 2006). Este ensaio tornou-se um dos mais citados sobre Educação em Computação, marcando o início de um movimento global de introdução do Pensamento Computacional nas escolas de Ensino Fundamental e Médio. Assim, a expressão Pensamento Computacional ganhou destaque, tornando-se tema de inúmeros capítulos de livros, artigos, projetos de pesquisa, dissertações, teses e atraindo interesses políticos e empresariais para o debate.

Propondo uma discussão inserida neste cenário, esta seção tem como objetivo apresentar e problematizar algumas perspectivas de Pensamento Computacional pós-Wing (2006), desdobrando-se principalmente nos seus impactos educacionais. Para embasar teoricamente as discussões propostas, apresentam-se as perspectivas de Pensamento Computacional de Wing (2006, 2008, 2011, 2017), Brackmann (2017), Denning e Tedre (2019), Brennan e Resnick (2012) e Dantas (2023). A seguir, discute-se sobre essas escolhas.

[19] A *National Science Foundation* é uma agência governamental dos Estados Unidos independente que promove a pesquisa e educação fundamental em diversos campos da ciência e engenharia.

Apresentar os trabalhos de Wing (2006, 2008, 2011, 2017) oferece uma perspectiva para explorar o Pensamento Computacional devido à sua influência fundamental no desenvolvimento e popularização do conceito. Traçando a trajetória das ideias de Wing desde seus primeiros trabalhos (2006) até publicações mais recentes (2017), elucidam-se os refinamentos e expansões de sua visão sobre o Pensamento Computacional.

Complementando as contribuições de Wing (2006, 2008, 2011, 2017), apresentam-se algumas críticas direcionadas aos seus trabalhos, com o intuito de enriquecer as discussões e problematizar as perspectivas de Pensamento Computacional por ela defendidas. Denning e Tedre (2019) argumentam que, embora as ideias de Wing (2006) tenham sido amplamente influentes, algumas de suas afirmações, como a suposta transferência automática de habilidades de Pensamento Computacional para outros domínios do conhecimento, carecem de suporte teórico e já haviam sido contestadas por pesquisas na área de educação em décadas anteriores. Além disso, são abordadas críticas referentes à falta de especificidade em certas definições de Pensamento Computacional propostas por Wing (2006, 2008).

Como precursor do tema no contexto educacional nacional, Brackmann (2017) corrobora muitas das ideias de Wing (2006, 2008, 2011), podendo ser alvo de críticas semelhantes. Contudo, apresentar a perspectiva de Brackmann (2017) sobre Pensamento Computacional, pode fornecer uma base robusta para a compreensão deste processo mental, pois o trabalho explora definições e sistematiza os quatro pilares do Pensamento Computacional, comumente retratados na literatura dedicada ao tema.

Denning e Tedre (2019) apresentam diversos aspectos do Pensamento Computacional, descrevendo-o não como um conceito monolítico, mas através de seis dimensões principais: métodos, máquinas, ciência da computação, engenharia de software, design e ciência computacional. Esta abordagem abrangente permite investigar o Pensamento Computacional em suas diversas aplicações, desde o desenvolvimento de algoritmos até a interpretação de fenômenos naturais como processos informacionais. A distinção entre o Pensamento Computacional para iniciantes e para profissionais fornece um embasamento teórico para avaliações diferenciadas para cada nível de pensamento, permitindo direcionar o foco dos conceitos a serem inseridos na Educação Básica.

Apesar das perspectivas sólidas de Wing (2006, 2008, 2011, 2017), Brackmann (2017) e Denning e Tedre (2019), esses trabalhos não exploram como o Pensamento Computacional pode ser operacionalizado em termos objetivos. Assim, discorre-se, na sequência, sobre pesquisas que embasam o uso de Tecnologias Digitais para operacionalização do Pensamento Computacional, Brennan e Resnick (2012) e Dantas (2023).

Brennan e Resnick (2012) expandem a perspectiva para três dimensões interdependentes: conceitos, práticas e perspectivas. Essa abordagem holística permite investigar como essas dimensões se manifestam e se desenvolvem em jovens estudantes no contexto da programação na plataforma Scratch. Além disso, o estudo apresenta diversas abordagens de pesquisa e avaliação do Pensamento Computacional, trazendo considerações significativas para a elaboração de uma metodologia de pesquisa robusta, conforme será discutido em um capítulo posterior.

Dantas (2023) trata de uma perspectiva operacional de Pensamento Computacional no âmbito da Educação Matemática, de modo que o autor apresenta exemplos utilizando o software GeoGebra. Essa abordagem prática, exemplificada por meio da resolução detalhada de um problema matemático, torna o trabalho especialmente relevante para pesquisas da área de Educação, pois oferece um norte para aplicação do Pensamento Computacional em sala de aula.

Cabe salientar que outras perspectivas de Pensamento Computacional também poderiam ser discutidas, tais como Ribeiro, Foss e Cavalheiro (2020) e Espadeiro (2021). Contudo, considerando o escopo deste estudo, que não visa esgotar o assunto, mas sim, apresentar *uma* perspectiva de Pensamento Computacional fundamentada no MCS, entende-se que as cinco abordagens apresentadas são suficientes para sustentar a argumentação aqui proposta.

Portanto, esta seção organiza-se da seguinte maneira: inicialmente, apresenta-se a perspectiva de Wing (2006, 2008, 2011, 2017) sobre Pensamento Computacional e a transformação de suas ideias, além das críticas recebidas; em seguida, apresenta-se a perspectiva de Brackmann (2017), fundamentada predominantemente nas ideias de Wing; na sequência, discute-se o trabalho de Denning e Tedre (2019) e como essa perspectiva de

Pensamento Computacional se distancia significativamente das outras; por fim, apresentam-se as perspectivas de Brennan e Resnick (2012) e Dantas (2023), visando operacionalizar o Pensamento Computacional com o Scratch e o GeoGebra, respectivamente.

2.3.1 Pensamento Computacional na perspectiva de Wing

Wing (2006), como precursora na disseminação mundial da expressão Pensamento Computacional, apresenta esse modo de pensar como essencial para todos, e não apenas para cientistas da computação, equiparando sua importância à leitura, à escrita e à aritmética. Em seu trabalho inicial sobre o tema, a autora argumenta que o Pensamento Computacional capacita indivíduos a resolver problemas e desenvolver sistemas que seriam impossíveis de serem tratados sem o uso desse conjunto de processos mentais.

Embora em Wing (2006) não seja possível identificar uma definição precisa de Pensamento Computacional, a autora afirma que é uma habilidade que "[...] envolve a resolução de problemas, o design de sistemas e a compreensão do comportamento humano, utilizando conceitos que são fundamentais para a ciência da computação" (Wing, 2006, p. 33, tradução nossa).[20] A autora enfatiza que o Pensamento Computacional também envolve as habilidades de abstração e decomposição para resolver problemas complexos, destacando a importância de representar adequadamente os dados e modelar aspectos relevantes do problema para torná-lo tratável computacionalmente.[21]

É possível afirmar que em Wing (2006) há um subtexto de que o Pensamento Computacional é pensar como um cientista da computação, englobando a capacidade de pensar em múltiplos níveis de abstração. Além disso, na perspectiva da autora, o Pensamento Computacional utiliza conceitos da computação para solucionar problemas de forma eficiente, considerando a dificuldade, as melhores abordagens e os recursos computacionais disponíveis.

[20] "[...] involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (Wing, 2006, p. 33).

[21] Cabe salientar que em seus ensaios iniciais (Wing, 2006, 2008), a autora se refere à "tratável computacionalmente" no sentido de algoritmos implementados e resolvidos por computadores eletrônicos. Em publicações posteriores, a autora expande sua definição, se referindo a soluções obtidas por computadores, sejam humanos ou máquinas, ou uma combinação de ambos (Wing, 2011, 2017), conforme será detalhado a diante.

No ensaio de 2008, a autora aprofunda as discussões do trabalho anterior, caracterizando o Pensamento Computacional como uma abordagem para solucionar problemas, projetar sistemas e compreender o comportamento humano, baseando-se em conceitos da computação (Wing, 2008). Segundo a autora, o Pensamento Computacional é uma forma de pensamento analítico que se assemelha ao pensamento matemático na abordagem geral para a resolução de problemas, ao pensamento de engenharia na forma como projeta e avalia sistemas complexos dentro das restrições do mundo real, e ao pensamento científico na maneira como busca compreender a computabilidade, a inteligência, a mente e o comportamento humano.

Wing (2008) argumenta que a essência do Pensamento Computacional reside na abstração, na qual noções são abstraídas além das dimensões físicas de tempo e espaço. No contexto da computação, as abstrações tendem a ser mais ricas e complexas em comparação com as abstrações matemáticas ou físicas, por duas razões principais. Primeiro, as abstrações computacionais não possuem, necessariamente, as propriedades algébricas bem definidas e elegantes encontradas em abstrações matemáticas, como números reais ou conjuntos. Segundo, as abstrações computacionais precisam considerar as limitações do mundo físico, levando em conta todas as suas variáveis e complexidades.

A importância de definir a abstração "correta" é destacada como um passo crucial no processo de Pensamento Computacional (Wing, 2008). Esse processo envolve a criação de camadas de abstração, permitindo a construção de sistemas complexos por meio de interfaces bem definidas entre essas camadas. Wing (2008) argumenta que a automação das abstrações, camadas de abstração e seus relacionamentos constitui a essência da computação. Essa automação é possibilitada por meio de notações e modelos precisos, que podem ser interpretados por um computador, ou seja, um ser humano ou uma máquina física, ou uma combinação de ambos.

Wing (2008) conclui seu ensaio lançando um desafio para as comunidades de Ciência da Computação e Educação: encontrar maneiras eficazes de ensinar Pensamento Computacional para crianças. Esse desafio levanta questões sobre quais são os conceitos elementares do Pensamento Computacional, como eles se relacionam com a cognição

humana e como podem ser integrados de forma eficaz ao ensino, utilizando ferramentas computacionais como apoio ao aprendizado.

Em seu trabalho seguinte, Wing (2011) trouxe uma definição mais precisa e rigorosa para a noção de Pensamento Computacional. Segundo a autora,

> Pensamento Computacional são os processos de pensamento envolvidos na formulação de problemas e suas soluções, de modo que as soluções sejam representadas em uma forma que possa ser efetivamente executada por um agente de processamento de informações. Informalmente, o pensamento computacional descreve a atividade mental de formular um problema para admitir uma solução computacional. A solução pode ser executada por um ser humano ou uma máquina ou, de modo mais geral, por combinações de seres humanos e máquinas (Wing, 2011, s. p., tradução nossa).[22]

Enquanto o ensaio de 2008 apresenta o Pensamento Computacional como uma "visão" com potencial para impactar diversos campos, o ensaio de 2011 busca defini-lo de forma mais concreta e aplicável. É possível observar uma mudança de foco, do campo da Ciência da Computação para as implicações práticas e educacionais do Pensamento Computacional.

Wing (2017) avança nas discussões relacionadas ao tema, trazendo uma definição semelhante[23] à de Wing (2011). A autora traz uma definição mais precisa das palavras "expressar" e "efetivo". Na perspectiva da autora, "expressar" significa criar uma representação linguística com o objetivo de comunicar uma solução a pessoas ou máquinas. Por sua vez, "eficaz" tem o sentido de computável, no contexto do modelo da máquina de Turing de computação.

Embora o ensaio de 2017 aborde o progresso do Pensamento Computacional, ele não repete a perspectiva de 2011 sobre o Pensamento Computacional se beneficiar do pensamento matemático e das engenharias. Wing (2017) reconhece que a Matemática e as

[22] "Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively conducted by an information-processing agent. Informally, computational thinking describes the mental activity in formulating a problem to admit a computational solution. The solution can be carried out by a human or machine, or more generally, by combinations of humans and machines" (Wing, 2011, s. p.).

[23] "O Pensamento Computacional é o processo de pensamento envolvido na formulação de um problema e na expressão de sua(s) solução(ões) de forma que um computador – humano ou máquina – possa executar com eficácia" (Wing, 2017, p. 8, tradução nossa). No original: "Computational thinking is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer – human or machine – can effectively carry out" (Wing, 2017, p. 8).

engenharias são a base para a Ciência da Computação.

Apesar da perspectiva de Wing (2006, 2008, 2011, 2017) ganhar contornos mais específicos e pragmáticos em relação aos processos mentais abrangidos pelo Pensamento Computacional, pode-se concluir que não houve progresso significativo nas discussões propostas pela autora sobre a implementação de seu ensino na Educação Básica. Na sequência, apresentam-se algumas críticas direcionadas aos trabalhos de Wing.

Valente (2019) observa que, embora a proposta de Wing (2006) tenha gerado um movimento importante entre os pesquisadores preocupados com a inserção das tecnologias digitais na educação, suas ideias foram criticadas por outros pesquisadores em várias frentes. Três, das principais críticas, são discutidas a seguir.

Tedre e Denning (2016) argumentam que Wing (2006) não reconheceu trabalhos anteriores que já exploravam ideias e conceitos semelhantes ao Pensamento Computacional, conforme discutido em seções anteriores. Especificamente, o trabalho de Seymour Papert já abordava a importância da abstração e da programação no desenvolvimento do pensamento, utilizando a linguagem Logo como ferramenta. A crítica reside na percepção de que Wing (2006) apresentou o Pensamento Computacional como uma ideia totalmente nova, ignorando as contribuições anteriores de Papert e outros (Valente, 2019).

Outra crítica direcionada ao trabalho de Wing (2006) é a falta de uma definição clara e concisa do que constitui o Pensamento Computacional (Tedre; Denning, 2016). A crítica argumenta que a autora se concentra excessivamente na perspectiva da Ciência da Computação, utilizando exemplos e conceitos específicos dessa área para explicar o Pensamento Computacional. Essa abordagem, segundo Tedre e Denning (2016) e Valente (2019), limita a compreensão do Pensamento Computacional como uma habilidade universalmente aplicável, dificultando sua difusão e implementação em outras áreas do conhecimento.

O trabalho de Wing (2006) também foi criticado por fazer alegações exageradas e pouco fundamentadas sobre os benefícios do Pensamento Computacional (Tedre; Denning, 2016). A autora sugere que o Pensamento Computacional pode ser aplicado a qualquer área do conhecimento, promovendo melhorias significativas na maneira como profissionais de diferentes áreas pensam e resolvem problemas (Valente et al., 2017). No entanto, conforme

discutido na seção anterior, não há evidências suficientes para sustentar essas alegações.

Além disso, Tedre e Denning (2016) criticam o termo "formular" presentes nas definições de Wing (2011), pois o termo causaria ambiguidade e poderia minimizar a importância do design computacional, aspecto considerado fundamental no Pensamento Computacional moderno. Segundo os autores, o termo "formular" é usado em dois sentidos distintos. O primeiro seria como "projetar computações", no sentido de criar soluções computacionais. O segundo seria "expressar comandos", podendo ser interpretado de forma simplista, como a mera emissão de comandos ou ações básicas em um computador, sem necessariamente envolver o raciocínio característico do Pensamento Computacional. Desse modo, os autores propõem substituir o termo "formular" por "projetar", argumentando que essa mudança tornaria as definições mais claras e alinhas com a perspectiva de Pensamento Computacional defendida por eles.

Expressando preocupação com a forte influência de interesses comerciais, particularmente de grandes empresas do Vale do Silício, Valente et al. (2017) alertam para a necessidade de um debate mais amplo sobre o contexto político e econômico que impulsionou a rápida disseminação do Pensamento Computacional, especificamente após a publicação do ensaio de Wing (2006). A proliferação de iniciativas que prometem transformar qualquer pessoa em um programador e a ideia de que o Pensamento Computacional seria uma habilidade capaz de resolver problemas de qualquer área do conhecimento podem estar mais alinhadas à demanda por mão de obra na indústria de software do que às reais necessidades educacionais.

Na sequência, discorre-se sobre a perspectiva de Pensamento Computacional defendida por Brackmann (2017).

2.3.2 Pensamento Computacional na perspectiva de Brackmann

Brackmann (2017) foi pioneiro ao trazer as discussões sobre Pensamento Computacional, majoritariamente internacionais, para o contexto educacional brasileiro, promovendo impactos significativos em pesquisas subsequentes, incluindo no âmbito da Educação Matemática.

Fundamentado predominantemente nas ideias de Wing (2006, 2008, 2011), que defende o Pensamento Computacional como uma habilidade aplicável à resolução de problemas em diversas áreas do conhecimento, e relacionando essa forma de pensar a um conjunto de habilidades oriundas da Ciência da Computação, o trabalho de Brackmann (2017) pode ser alvo de críticas semelhantes às apresentadas na seção anterior.

Contudo, Brackmann (2017) avança nas discussões sobre o tema, definindo o Pensamento Computacional de forma pragmática. Mesmo admitindo a ausência de consenso sobre o significado exato dessa expressão, o autor define Pensamento Computacional como

> [...] uma distinta capacidade criativa, crítica e estratégica humana de saber utilizar os fundamentos da Computação, nas mais diversas áreas do conhecimento, com a finalidade de identificar e resolver problemas, de maneira individual ou colaborativa, através de passos claros, de tal forma que uma pessoa ou máquina possam executá-los eficazmente (Brackmann, 2017, p. 29).

Concentrando-se no desenvolvimento do Pensamento Computacional em estudantes da Educação Básica por meio de atividades desplugadas, ou seja, atividades que não dependem do uso de computadores ou dispositivos eletrônicos, Brackmann (2017) oferece uma alternativa para o ensino de Pensamento Computacional em contextos com limitações tecnológicas. Utilizando atividades desplugadas como ferramenta principal, o autor investiga a viabilidade, efetividade e impacto dessa abordagem em diferentes grupos de estudantes.

Sistematizando diversas pesquisas realizadas na área da Computação e da Educação, Brackmann (2017) propõe os denominados pilares do Pensamento Computacional, a saber: decomposição, reconhecimento de padrões, abstração e algoritmos. Segundo o autor, a aprendizagem do Pensamento Computacional, por meio da apropriação dos quatro pilares, permite que os estudantes identifiquem, compreendam e solucionem problemas de forma estruturada e eficiente.

De acordo com Brackmann (2017), a decomposição vai além de dividir um problema em partes menores. Em sua perspectiva, a decomposição envolve identificar subproblemas que podem ser resolvidos individualmente e que, quando combinados, resultam na solução do problema original. Como exemplo, o autor cita o funcionamento de uma bicicleta, cujo entendimento é mais fácil através do desmembramento de suas partes. A Figura 2.5 ilustra

o exemplo citado: uma bicicleta e os seus principais mecanismos.

Figura 2.5 – Partes da bicicleta exemplificando a decomposição

Fonte: Brackmann (2017).

Brackmann (2017) afirma que, ao invés de tentar entender a bicicleta como um todo, pode-se decompô-la em partes menores, como o sistema de estrutura, o sistema de tração e o sistema de transferência de energia. Essa abordagem facilita a identificação da função de cada componente e como eles interagem entre si. Além disso, a decomposição facilita a manutenção de sistemas, pois permite que partes individuais sejam examinadas e reparadas sem a necessidade de substituir o sistema inteiro.

Segundo Brackmann (2017), a decomposição facilita a análise, compreensão e resolução do problema original. No caso de problemas computacionais, a decomposição é frequentemente utilizada por programadores para dividir algoritmos em partes menores, tornando-as mais fáceis de entender e gerenciar. Essa técnica pode ser exemplificada pela divisão de um código-fonte em funções, procedimentos, objetos e módulos, simplificando a resolução de problemas complexos, facilitando a compreensão de novas situações e possibilitando o desenvolvimento de sistemas de grande porte.

O reconhecimento de padrões é outro pilar do Pensamento Computacional na perspectiva de Brackmann (2017). É compreendido como a capacidade de encontrar similaridades para resolver problemas complexos de forma mais eficiente, geralmente sendo seguido após a realização da decomposição de um problema. Esse pilar busca elementos iguais ou semelhantes em diferentes problemas, simplificando a busca por soluções e

agilizando a resolução de problemas.

Como exemplo da aplicação do reconhecimento de padrões, Brackmann (2017) cita a identificação de similaridades entre raças de cães, conforme apresenta a Figura 2.6.

Figura 2.6 – Similaridades entre raças de cachorros exemplificando o reconhecimento de padrões

Fonte: Brackmann (2017).

Segundo Brackmann (2017), apesar de características comuns como olhos, rabo e pelos, cada raça possui particularidades, como cor dos olhos, comprimento do rabo e cor dos pelos. No Pensamento Computacional, essas características são consideradas padrões. Assim, ao reconhecer o padrão “cachorro”, pode-se descrever diferentes raças apenas alterando suas características, como “olhos azuis, rabo curto e pelugem cinza” ou “olhos pretos, rabo longo e pelos ruivos”.

A habilidade de reconhecer padrões permite replicar a solução em diversos sub-problemas semelhantes (Brackmann, 2017). Dessa forma, encontrar mais padrões torna a solução mais dinâmica e rápida. Caso o reconhecimento de padrões não seja empregado, seria necessário refazer a análise das características do problema repetidamente. Segundo o autor, essa prática é ineficiente e aumenta a chance de erros, tornando a execução da tarefa possivelmente mais lenta.

A abstração, também defendida por Brackmann (2017) como um dos pilares do Pensamento Computacional, refere-se à capacidade de filtrar e classificar dados, ignorando elementos irrelevantes e concentrando-se nos aspectos essenciais para a resolução de um problema. Em outras palavras, trata-se da habilidade de criar uma representação

simplificada da realidade, focando nos detalhes importantes e omitindo os desnecessários.

De acordo com Brackmann (2017), a abstração é crucial para tornar os problemas mais fáceis de entender e gerenciar, sem comprometer a integridade da solução. O autor exemplifica a abstração na representação de um mapa de metrô, que omite detalhes geográficos e arquitetônicos irrelevantes, concentrando-se nas linhas, estações e conexões essenciais para o usuário, conforme apresentado na Figura 2.7.

Figura 2.7 – Um mapa de metrô exemplificando a abstração no mundo real

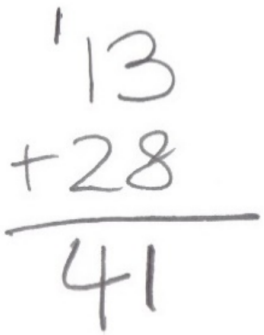
Fonte: Brackmann (2017).

Especificamente no Pensamento Computacional, Brackmann (2017) enfatiza a importância da abstração em diversos processos de resolução de problemas, como na escrita de algoritmos, na seleção de dados relevantes para o problema em questão, na formulação de perguntas claras e objetivas, focando nos pontos chave do problema e na compreensão de sistemas complexos, facilitando o processo de decomposição. Como exemplo, o autor afirma que um algoritmo é uma abstração de um processo que recebe uma entrada, executa uma sequência de passos e produz uma saída que satisfaça um objetivo específico.

Sobre o pilar dos algoritmos, Brackmann (2017) argumenta que este é o elemento

que agrega todos os demais, ou seja, o núcleo do Pensamento Computacional. Para o autor, um algoritmo pode ser entendido como um plano, uma estratégia ou um conjunto de instruções claras que levam à solução de um problema. Assim, as instruções em um algoritmo são organizadas e descritas em uma ordem específica para alcançar o objetivo desejado. Essas instruções podem ser representadas de diversas maneiras, como diagramas, pseudocódigo (linguagem similar à humana) e, também, código em uma linguagem de programação. Brackmann (2017) ilustra o conceito de algoritmo com o exemplo da soma com números sobrepostos, ou seja, a "conta armada", conforme ilustrado na Figura 2.8.

Figura 2.8 – Conta de soma "armada" exemplificando o uso de algoritmos



Fonte: Brackmann (2017).

Ao definir os passos para realizar a conta, como a soma das unidades, dezenas e centenas, cria-se um algoritmo que pode ser aplicado a quaisquer números. Nesse contexto, Brackmann (2017) destaca a importância da clareza e precisão na formulação de um algoritmo para que ele possa ser executado de forma eficiente, seja por uma pessoa ou por uma máquina. Assim, uma vez que o algoritmo esteja bem definido, ele pode ser usado repetidamente para resolver o mesmo tipo de problema, sem a necessidade de ser reinventado a cada uso.

A perspectiva de Pensamento Computacional defendida por Brackmann (2017) apresenta desdobramentos educacionais no contexto nacional, especialmente na democratização do acesso ao Pensamento Computacional por meio de atividades desplugadas. O autor reconhece a disparidade no acesso às Tecnologias Digitais nas escolas brasileiras e propõe o uso de atividades que não dependem de computadores como uma solução para essa questão.

Utilizando jogos, brincadeiras e materiais manipuláveis, Brackmann (2017) argumenta que essa abordagem pode tornar-se acessível a um público mais amplo, garantindo que a localização geográfica ou a infraestrutura da escola não sejam impedimentos para o desenvolvimento desse modo de pensar. No entanto, é possível inferir que a abordagem desplugada tem suas limitações e não substitui completamente a experiência prática com computadores.

Conforme argumentado por Brackmann (2017), embora a abordagem desplugada para o ensino do Pensamento Computacional ofereça uma via de acesso equitativa, superando barreiras geográficas e de infraestrutura escolar, é crucial reconhecer suas inerentes limitações. O próprio estudo, ao investigar a eficácia do uso exclusivo de atividades desplugadas, sugere uma potencial incompletude desta metodologia quando isolada (Brackmann, 2017). Segundo o autor, as atividades desplugdas, embora valiosas para a introdução de conceitos e para o desenvolvimento inicial do raciocínio computacional, podem não abranger todos os fundamentos da Computação nem proporcionar uma experiência prática plena.

Ademais, a ausência da interação direta com dispositivos digitais restringe a familiarização dos estudantes com o ambiente computacional real e com a prática da programação em si, considerada uma experiência fundamental e praticamente singular (Brackmann, 2017). Segundo o autor, a utilização exclusiva de atividades desplugadas pode, com o tempo, distanciar os alunos de uma compreensão mais completa e autêntica do que é a Computação e de suas aplicações práticas. Nesse sentido, Brackmann (2017) enfatiza que, embora a abordagem desplugada seja eficaz para a introdução ao Pensamento Computacional, ela não deve ser encarada como uma solução de ensino completa, sendo recomendável a sua integração com experiências que envolvam o uso de Tecnologias Digitais para uma formação mais abrangente.

Embora o autor não explore diretamente como os alunos constroem conhecimento sobre o Pensamento Computacional, é possível realizar algumas inferências sobre os pressupostos epistemológicos de Brackmann (2017). A ênfase na importância desse modo de pensar para além da área da Computação, como ferramenta para “compreender o

mundo", sugere uma visão em que a Computação oferece lentes poderosas para analisar e solucionar problemas em diversas áreas do conhecimento. Além disso, a busca por soluções eficazes para problemas, por meio dos pilares do Pensamento Computacional, sugere uma abordagem pragmática para o conhecimento, em que a aplicabilidade e a funcionalidade são valorizadas. Por fim, o uso de atividades e jogos para o ensino de Pensamento Computacional sugere uma abordagem que valoriza a experimentação, a prática e a interação como elementos importantes no processo de aprendizagem.

Na sequência, discorre-se sobre a perspectiva de Pensamento Computacional de Denning e Tedre (2019).

2.3.3 Pensamento Computacional na perspectiva de Denning e Tedre

Em seções anteriores, foram apresentados alguns aspectos da perspectiva de Denning e Tedre (2019) sobre o Pensamento Computacional. No entanto, apesar das discussões levantadas, até esse momento não foi apresentada a definição formal de Pensamento Computacional na perspectiva desses autores. Sendo assim, eles o definem como

> [...] as habilidades e práticas mentais para *projetar* cálculos que fazem com que os computadores realizem tarefas para nós, e *explicar* e interpretar o mundo como um complexo processo de informações. O aspecto de design reflete a tradição da engenharia da computação, na qual as pessoas criam métodos e máquinas para ajudar outras pessoas. O aspecto da explicação reflete a tradição científica da computação, na qual as pessoas procuram entender como a computação funciona e como ela aparece no mundo. O *design* caracteriza a imersão na comunidade que está sendo ajudada e a *explicação* caracteriza um observador externo imparcial (Denning; Tedre, 2019, p. 17, tradução e grifos nossos).[24]

Fundamentados na definição apresentada no parágrafo anterior, Denning e Tedre (2019) exploram a história do Pensamento Computacional, traçando suas raízes desde tempos antigos, antes do surgimento dos computadores eletrônicos. Os autores argumentam que o Pensamento Computacional não é um conceito novo, mas sim um modo de pensar que

[24] "[...] the mental skills and practices for *designing* computations that computers to do job for us, and *explaining* and interpreting the world as a complex of information processes. The design aspect reflects the engineering tradition of computing in which people build methods and machines to help other people. The explanation aspect reflects the science tradition of computing in which people seek to understand how computation works and how it shows up in the world. *Design* features immersion in the community being helped, *explanation* features being a dispassionate external observer" (Denning; Tedre, 2019, p. 17, grifos do autor).

evoluiu ao longo de milênios, moldado por matemáticos, lógicos, cientistas e engenheiros que buscavam realizar cálculos complexos e inferências sem erros, conforme apresentado em seções anteriores.

Essa perspectiva histórica lança luz sobre a natureza do Pensamento Computacional como uma forma de pensar inerente à experiência humana, evidenciando seis dimensões interrelacionadas: métodos, máquinas, ciência da computação, engenharia de software, design e ciência computacional. De acordo com Denning e Tedre (2019), cada dimensão oferece uma lente através da qual se pode examinar o Pensamento Computacional, relevando suas nuances e aplicações. Essas dimensões são discutidas brevemente a seguir.

A dimensão dos métodos encapsula a essência da computação como um processo de automatização de tarefas por meio de instruções precisas e inequívocas (Denning; Tedre, 2019). Desde os primeiros métodos numéricos desenvolvidos por matemáticos e engenheiros até as técnicas modernas de representação simbólica, a ênfase reside na criação de métodos passo a passo que eliminem a ambiguidade e possibilitem a execução por computadores, sejam humanos ou máquinas.

As máquinas constituem uma das dimensões do Pensamento Computacional na perspectiva de Denning e Tedre (2019). Segundo os autores, a natureza do Pensamento Computacional se molda às capacidades e peculiaridades de cada máquina, desde as primeiras máquinas de calcular, como o ábaco, até os computadores quânticos modernos. Com a invenção dos computadores eletrônicos, inaugurou-se uma nova era no Pensamento Computacional, exigindo uma compreensão profunda das operações de baixo nível, como representação binária e controle de fluxo.

A dimensão da ciência da computação explora a estrutura intelectual que sustenta o Pensamento Computacional. Denning e Tedre (2019) destacam como a Ciência da Computação forneceu as ferramentas conceituais essenciais para o Pensamento Computacional moderno, como linguagens de programação, sistemas operacionais e análise de algoritmos. Nesse contexto, a busca por linguagens de programação de alto nível demonstra a progressão do Pensamento Computacional para além da programação de máquina, em direção a abstrações mais poderosas e expressivas.

Reconhecendo que a criação de sistemas de software robustos e confiáveis apresenta desafios únicos, Denning e Tedre (2019) argumentam que a engenharia de software deve ser uma dimensão a ser explorada no contexto do Pensamento Computacional. Com o desenvolvimento de projetos e equipes de engenheiros de software cada vez maiores, os autores defendem que o Pensamento Computacional também deve se preocupar com questões para além da funcionalidade básica, abrangendo a segurança, confiabilidade e usabilidade.

Denning e Tedre (2019) enfatizam a importância crítica do design como uma das dimensões do Pensamento Computacional, centrando-se no ser humano e reconhecendo que softwares e sistemas computacionais devem atender às necessidades e expectativas dos usuários. A ascensão do design como um componente essencial do Pensamento Computacional é evidente na crescente ênfase na usabilidade, estética e experiência geral do usuário, indo além dos aspectos puramente técnicos da computação.

Por fim, Denning e Tedre (2019) destacam a dimensão da ciência computacional como uma influência transformadora do Pensamento Computacional na ciência, capacitando os cientistas a enfrentar problemas complexos e obter novos *insights* por meio de simulações computacionais e análise de dados em larga escala. Os autores ilustram como o Pensamento Computacional revolucionou campos como a física, biologia e química, permitindo a modelagem de fenômenos complexos, como dinâmica de fluidos, interações de proteínas e dinâmica molecular.

Ao inserir a discussão no âmbito educacional, Denning e Tedre (2019) alertam para a necessidade de se construir um entendimento sólido e abrangente do Pensamento Computacional na educação, evitando equívocos e simplificações. Os autores identificam diversos pontos de controvérsia em relação à definição e aos objetivos do ensino de Pensamento Computacional, incluindo a relação entre algoritmos e abstração, a dependência do contexto e a capacidade de transferência das habilidades computacionais. Eles defendem uma abordagem pluralista para o ensino de Pensamento Computacional, que reconheça a riqueza e a diversidade de perspectivas dentro da área da Computação, sem perder de vista os rigores da disciplina. Para tanto, os autores enfatizam a diferença entre o Pensamento

Computacional para iniciantes e para profissionais.

O Pensamento Computacional para iniciantes serve como introdução aos conceitos básicos da computação, tendo como objetivo familiarizar o aluno com os princípios básicos do pensamento algorítmico, como a decomposição de problemas, a representação de dados e a criação de algoritmos. Seu foco está em uma abordagem prática e introdutória, utilizando linguagens de programação, como Python, e ferramentas visuais, como Scratch. Os autores argumentam que esse nível de Pensamento Computacional não garante a transferência direta de habilidades para áreas especializadas ou a resolução de problemas complexos, sendo essencial evitar expectativas infladas sobre o alcance do Pensamento Computacional para iniciantes.

Por outro lado, o Pensamento Computacional para profissionais tem como objetivo desenvolver habilidades avançadas de design para construir sistema de softwares complexos, confiáveis e seguros, além de conduzir pesquisas científicas. Seu foco está no domínio de linguagens de programação, arquiteturas de sistemas, redes, design de software, métodos formais e conhecimento aprofundado da área de aplicação. Exemplos incluem o design de sistema operacionais, o desenvolvimento de softwares de inteligência artificial e a modelagem computacional em áreas como física, química e biologia. Entre os desafios enfrentados, destacam-se lidar com as crises de software, gerenciar equipes, garantir a segurança e a confiabilidade em sistemas de larga escala e integrar o Pensamento Computacional com as necessidades específicas de cada área.

Dessa forma, na perspectiva defendida pelos autores, o Pensamento Computacional é um primeiro passo importante, mas o desenvolvimento de habilidades de design complexas e a aplicação do Pensamento Computacional em áreas especializadas exigem tempo, prática e estudo contínuo. Criticando perspectivas que tratam esse modo de pensar como uma habilidade fundamental para todos, os autores alertam contra o “pensamento mágico” e enfatizam a importância de entender os limites do Pensamento Computacional.

Na sequência, apresenta-se uma perspectiva de Pensamento Computacional que permite operacionalizá-lo por meio do Scratch.

2.3.4 Pensamento Computacional na perspectiva de Brennan e Resnick

Embora as perspectivas de Pensamento Computacional apresentadas até o momento sejam sólidas, essas pesquisas não evidenciam claramente como essa habilidade pode ser aplicada no processo de resolução de problemas. Assim, as discussões disparadas na sequência visam demonstrar como o Pensamento Computacional pode ser operacionalizado por meio de Tecnologias Digitais[25], com foco no Scratch. Para tanto, nesta seção, as discussões são sustentadas pelas ideias de Brennan e Resnick (2012).

Propondo uma discussão sobre a utilização do Scratch para operacionalizar o Pensamento Computacional, Brennan e Resnick (2012), no texto *New frameworks for studying and assessing the development of computational thinking*, discorrem sobre sua perspectiva a respeito do Pensamento Computacional e como ele pode ser explorado e operacionalizado com o uso do Scratch.

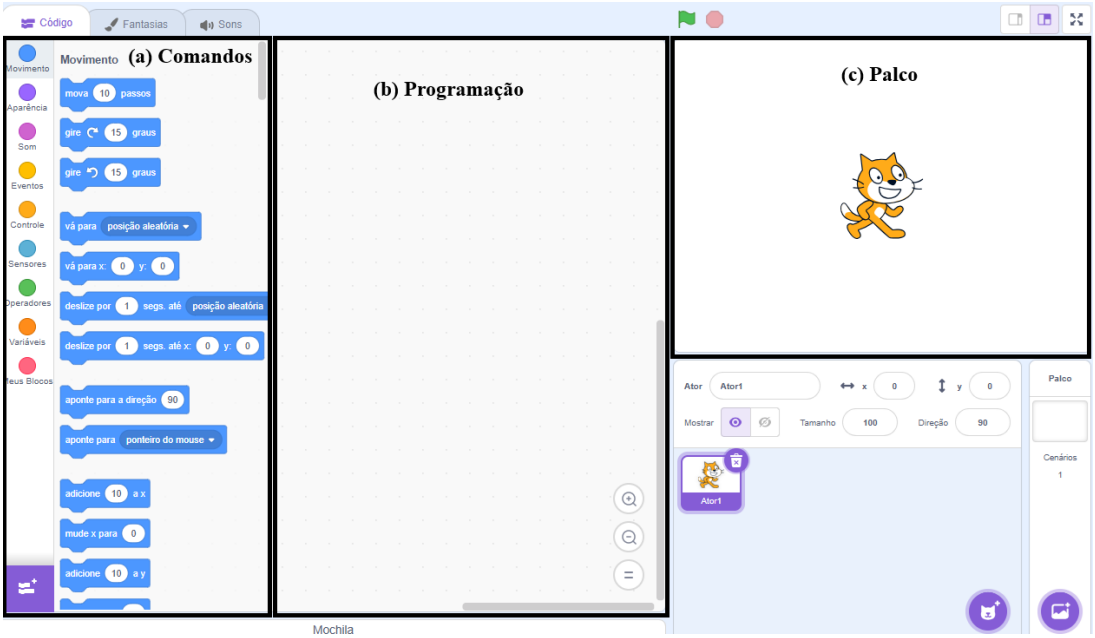
O Scratch é uma plataforma on-line de programação por bloco, desenvolvida pelo Grupo de Pesquisa Lifelong Kindergarten do Instituto de Tecnologia de Massachussets (MIT), liderado pelo professor Mitchel Resnick e lançada ao público em 2007. Segundo seu site oficial[26], o Scratch promove o Pensamento Computacional e habilidades de resolução de problemas, permitindo aos seus usuários criarem histórias, jogos e animações digitais.

De acordo com Raabe, Couto e Blikstein (2020), o Scratch é o principal "herdeiro" da linguagem Logo, mantendo muitos de seus princípios, mas diferenciando-se em alguns aspectos. A intenção do grupo foi criar um ambiente que possibilitasse a construção de narrativas, histórias interativas e jogos. A Figura 2.9 ilustra o ambiente de desenvolvimento de projetos do Scratch.

[25] A expressão "Tecnologias Digitais" é empregada neste trabalho segundo a perspectiva de Dantas (2022). De acordo com o autor, a expressão se refere "aos componentes físicos ou virtuais, que possibilitam que a informação seja codificada, organizada e recuperada quando necessário. São exemplos: computadores, dispositivos móveis, dispositivos imateriais – como programas para computadores ou aplicativos para celulares" (Dantas, 2022, p. 20).

[26] Disponível em: https://scratch.mit.edu/. Acesso em: 02 mar. 2024.

Figura 2.9 – Ambiente de desenvolvimento de projetos do Scratch



Fonte: Os autores.

A Figura 2.9 apresenta as três principais áreas da tela ao criar um projeto no Scratch: (a) área dos comandos, onde estão localizados todos os blocos de programação, divididos em categorias de acordo com suas funcionalidades; (b) área da programação, onde são desenvolvidos os algoritmos por meio da junção de blocos; (c) palco, onde são visualizados os atores (personagens/objetos), o cenário e o resultado do programa. Essas áreas formam a interface visualizada pelo usuário que programa com o Scratch.

Desse modo, estudando construções de crianças de 8 a 12 anos publicadas no Scratch, Brennan e Resnick (2012) coletaram dados via entrevistas com algumas delas e delinearam uma perspectiva de Pensamento Computacional baseada em três dimensões-chave. De acordo com os autores:

> [...] ao estudar a atividade na comunidade online do Scratch e nos workshops do Scratch, desenvolvemos uma definição de pensamento computacional que envolve três dimensões principais: *conceitos computacionais* (os conceitos que os designers empregam quando programam), *práticas computacionais* (as práticas que os designers desenvolvem enquanto programam) e *perspectivas computacionais* (as perspectivas que os designers formam sobre o mundo ao seu redor e sobre si mesmos) (Brennan; Resnick, 2012, p. 3, tradução e grifos nossos).[27]

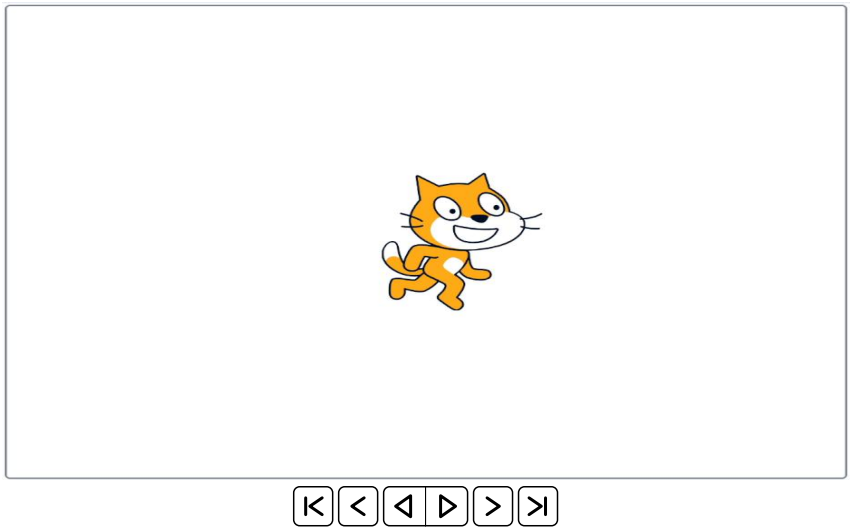
[27] "[...] by studying in the Scratch online community and in Scratch workshops, we have developed a definition of computational thinking that involves three key dimensions: *computational concepts* (the concepts designers employ as they program), *computational practices* (the practices designers develop

A seguir, discorre-se sobre as três dimensões do Pensamento Computacional na perspectiva de Brennan e Resnick (2012), com a apresentação de exemplos implementados no Scratch.

Os conceitos computacionais referem-se às técnicas que os designers empregam durante a codificação para obter resultados animados, dinâmicos, estéticos e interativos (Brennan; Resnick, 2012). Especificamente no Scratch, a codificação ocorre por meio da construção de blocos de instruções. Segundo os autores, há sete conceitos computacionais operacionais no Scratch: sequências, *loops*, paralelismo, eventos, condicionais, operadores e dados.

O conceito de sequências evidencia que as atividades ou tarefas são representadas em programação como uma série de etapas ou instruções individuais executadas pelo computador em uma ordem (Brennan; Resnick, 2012). Por exemplo, considere uma animação no Scratch em que um gato caminha na tela, para, diz "Miau!", aguarda alguns segundos, volta a caminhar e repete esse ciclo de ações. O conceito de sequência é utilizado para obter esse conjunto de ações integradas e em série. Na Figura 2.10, apresenta-se uma possível construção no Scratch exemplificando esse caso.

Figura 2.10 – Criação de uma sequência no Scratch



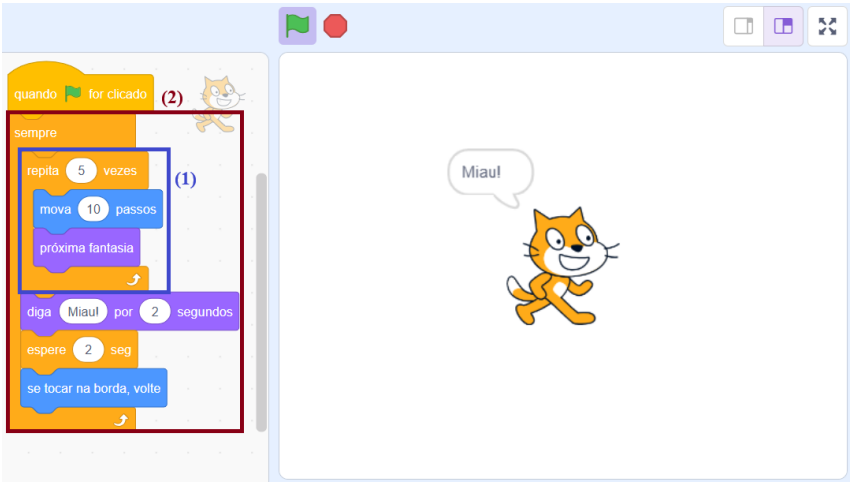
Fonte: Os autores.

De acordo com Brennan e Resnick (2012), os *loops* fornecem um mecanismo para

as they program), and *computational perspectives* (the perspectives designers form about the world around them and about themselves)" (Brennan; Resnick, 2012, p. 3, grifos do autor).

executar uma mesma sequência de instruções várias vezes, promovendo a concisão e a eficiência na programação. Em outras palavras, pode ser necessário repetir uma parte de um conjunto de blocos para produzir um efeito visual desejado ou processar a entrada de vários dados. Nesse caso, em vez de construir um conjunto de blocos que se repetem, utiliza-se um controle de repetição. Para realizar a construção apresentada na Figura 2.10, foram utilizados dois blocos de repetições, conforme destaca a Figura 2.11.

Figura 2.11 – Identificando *loops* no Scratch

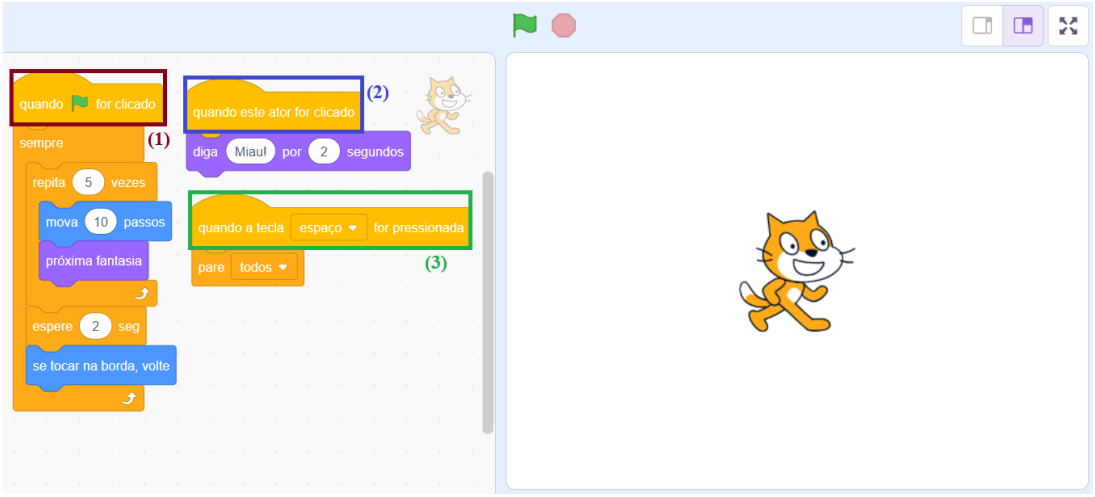


Fonte: Os autores.

Na Figura 2.11, observam-se duas sequências de blocos que se repetem, indicadas em azul (1) e em vermelho (2). O bloco `[repita 5 vezes]` engloba os blocos `[mova 10 passos]` e `[próxima fantasia]`, indicando que cada uma dessas ações deve ser executada cinco vezes pelo código, não sendo necessário repetir manualmente a sequência de blocos. Por sua vez, o bloco `[sempre]` destacado em vermelho (2), engloba todas as ações que o gato deve realizar nesse projeto, indicando que todas as ações alocadas dentro desse bloco devem se repetir indefinidamente, criando uma sequência de ações contínua.

Os eventos referem-se à ideia de que uma ação desencadeia outra, como clicar em um botão ou colidir objetos, resultando em respostas específicas (Brennan; Resnick, 2012). Especificamente no Scratch, existem diversos blocos que disparam outros, como a detecção de uma tecla pressionada, o clique do mouse ou o início de um projeto. A Figura 2.12 apresenta alguns exemplos de eventos.

Figura 2.12 – Identificando eventos no Scratch



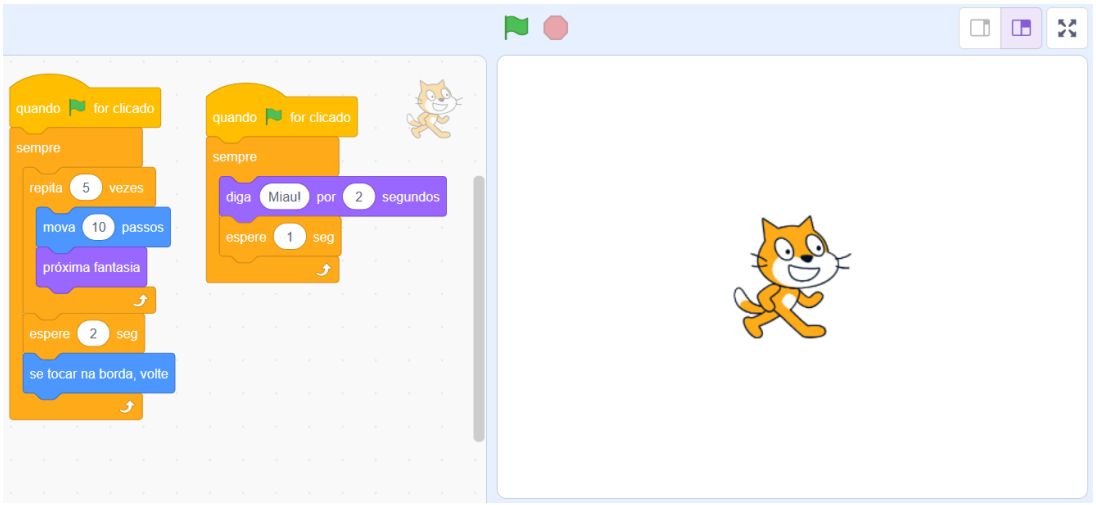
Fonte: Os autores.

Na construção apresentada na Figura 2.12, observam-se três eventos distintos. O primeiro, indicado em vermelho (1), dispara a sequência de blocos ao clicar na bandeira verde para iniciar o projeto. O segundo evento, indicado em azul (2), é ativado quando o ator é clicado pelo usuário, e as ações subsequentes são desencadeadas após esse evento. O terceiro evento, indicado em verde (3), é acionado quando a tecla espaço do teclado é pressionada.

O conceito de paralelismo envolve a execução simultânea de múltiplas sequências de instruções, permitindo a criação de programas dinâmicos e complexos, nos quais diferentes partes do projeto operam de maneira independente (Brennan; Resnick, 2012). Essa abordagem possibilita que ações e eventos ocorram ao mesmo tempo, como na criação de animações ou simulações em que vários elementos executam tarefas distintas de maneira coordenada. Nesse contexto, é possível, por exemplo, que personagens interajam entre si, realizem ações simultâneas ou respondam a estímulos variados.

Considere o exemplo das Figuras 2.10 e 2.11, em que um gato caminha na tela, para, diz “Miau!”, aguarda alguns segundos, volta a caminhar e repete o ciclo de ações. Suponha que se queira construir uma animação onde, ao mesmo tempo em que o gato caminha, ele diga “Miau!” periodicamente. A Figura 2.13 ilustra essa construção.

Figura 2.13 – Identificando paralelismo no Scratch

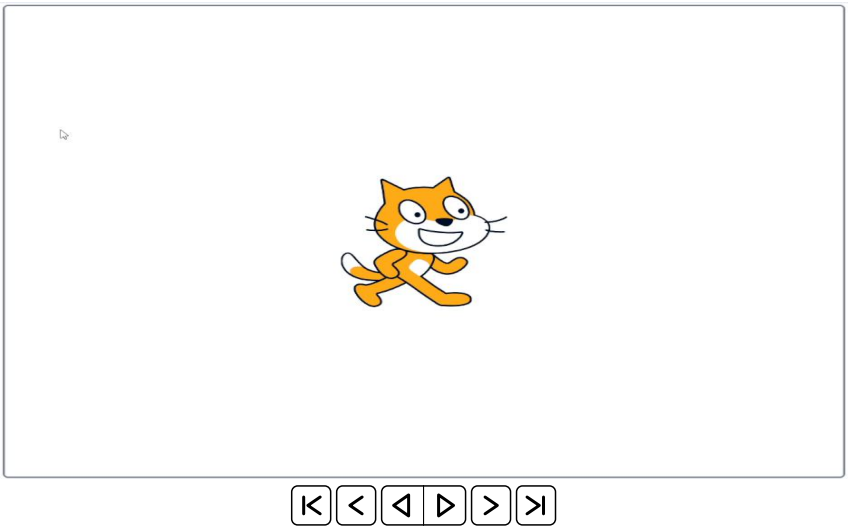


Fonte: Os autores.

Na Figura 2.13, observam-se duas sequências de blocos disparadas simultaneamente ao iniciar o código. Enquanto a primeira sequência movimenta o ator e verifica a colisão com a borda, a segunda faz com que, a cada intervalo de tempo, o ator diga “Miau!”. Essa implementação é possível por meio de comandos que executam simultaneamente os eventos.

De acordo com Brennan e Resnick (2012), as condicionais permitem a tomada de decisão com base em condições específicas, permitindo múltiplos resultados e respostas variadas dentro de um programa. Por exemplo, suponha que se queira construir um jogo em que o jogador deva fugir do gato com o mouse e, caso não consiga, uma mensagem “Te peguei!” seja exibida. A Figura 2.14 apresenta uma possível construção.

Figura 2.14 – Exemplo de utilização de condicionais no Scratch



Fonte: Os autores.

Para realizar a construção apresentada na Figura 2.14, foram utilizados os blocos apresentados na sequência, conforme ilustra a Figura 2.15.

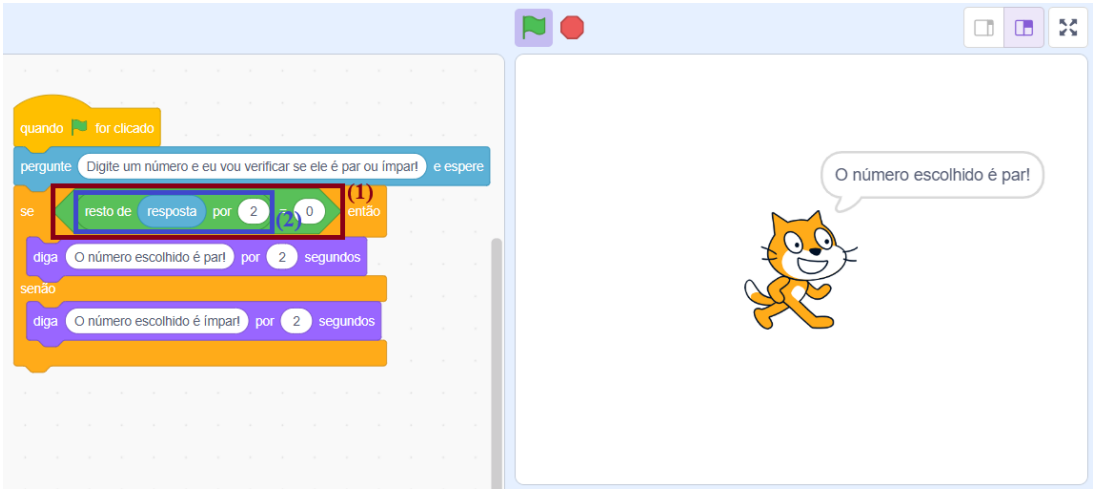
Figura 2.15 – Exemplo de blocos de condicionais no Scratch

Fonte: Os autores.

Na Figura 2.15, observa-se em vermelho (1) uma das estruturas condicionais presentes no Scratch, com uma condição indicada em azul (2), que determina quando os blocos alocados dentro dessa condicional serão executados. No exemplo, a construção verifica se o ator está tocando no ponteiro do mouse e, caso essa condição seja verificada, o ator diz “Te pegueil”.

Segundo Brennan e Resnick (2012), os operadores são conceitos computacionais que possibilitam construir expressões matemáticas, lógicas e de *string*[28], permitindo aos programadores realizar manipulações e cálculos dentro de seus projetos. Com os operadores, é possível, por exemplo, verificar se um número é maior que outro ou se uma frase contém um determinado caractere. É comum combinar os operadores com condicionais, conforme exemplificado na Figura 2.16.

Figura 2.16 – Exemplo de utilização de operadores no Scratch



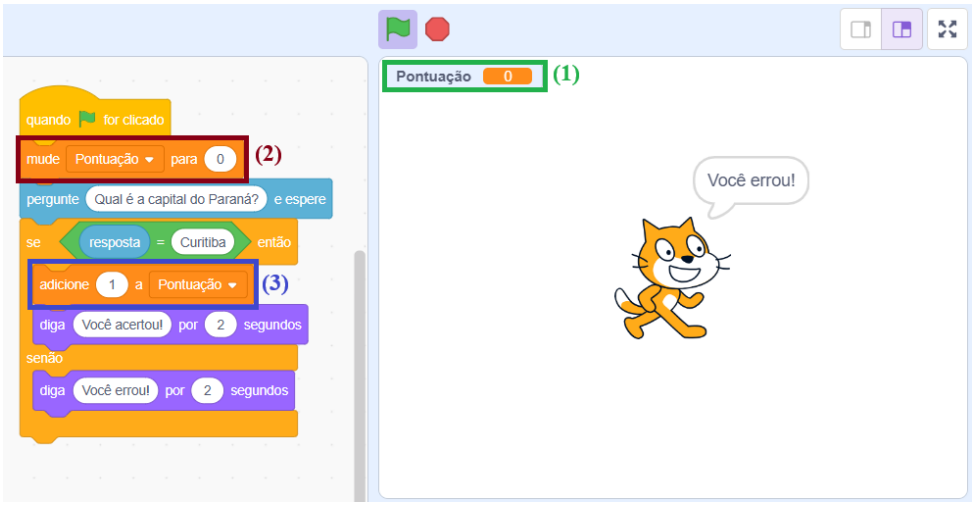
Fonte: Os autores.

Na Figura 2.16, a construção verifica se um número é par ou ímpar. Para tanto, combina-se o uso de condicionais com operadores. O bloco indicado em azul (2) calcula o resto da divisão do número escolhido por 2, enquanto o bloco indicado em vermelho (1) verifica se esse resto é igual a 0. Assim, caso o resto seja igual a 0, o ator afirma que o número é par; caso contrário, o ator diz que o número é ímpar.

O último conceito computacional elencado por Brennan e Resnick (2012) é o de dados, que abrange o armazenamento, recuperação e atualização de valores, elementos essenciais na programação para gerenciar informações dentro de um projeto. No Scratch, dados podem ser armazenados de duas maneiras: variáveis, que guardam um único número ou *string*, e listas, que armazenam uma coleção de números ou *strings*. A Figura 2.17 exemplifica o uso de dados.

[28] *Strings* são uma sequência de caracteres, como letras, números e símbolos.

Figura 2.17 – Exemplo de utilização de dados no Scratch



Fonte: Os autores.

A construção apresentada na Figura 2.17 ilustra a aplicação do uso de dados no Scratch por meio da elaboração de um placar. No canto superior esquerdo do palco, indicada em verde (1), há uma caixa que armazena a pontuação. Inicialmente, a pontuação é alterada para zero, conforme indicado pelo bloco em vermelho (2). Nesse exemplo, o ator realiza uma pergunta e, se a resposta estiver correta, a pontuação é acrescida em uma unidade, conforme indicado pelo bloco em azul (3), e a mensagem "Você acertou!" é exibida. Caso a resposta esteja errada, não se adicionam pontos e a mensagem "Você errou!" é exibida.

Embora os conceitos computacionais combinados forneçam uma base robusta para o desenvolvimento do Pensamento Computacional e promovam habilidades essenciais para a codificação e resolução de problemas no Scratch, Brennan e Resnick (2012) argumentam que esses conceitos não podem ser considerados isoladamente. Os autores destacam que a mera presença de blocos de código em um projeto não garante a compreensão profunda dos conceitos subjacentes. Assim, Brennan e Resnick (2012) propõem explorar o Pensamento Computacional em mais duas frentes: práticas e perspectivas computacionais.

As práticas computacionais, como apresentadas por Brennan e Resnick (2012), vão além dos conceitos básicos e examinam os processos pelos quais os designers constroem e aprendem durante a criação de projetos no Scratch. Segundo os autores, essas práticas enfatizam *como* os alunos aprendem, e não apenas *o que* eles aprendem. Brennan e Resnick

(2012) identificam quatro práticas computacionais principais: ser incremental e iterativo, testar e depurar, reutilizar e remixar, e abstrair e modularizar.

A prática de ser incremental e iterativo envolve ciclos repetidos de desenvolvimento, construção de soluções e testes, adaptando planos com base em *feedback* e novas ideias. Testar e depurar capacita os alunos a desenvolver métodos para testar projetos, identificar problemas e encontrar soluções, envolvendo a revisão de código, experimentação de soluções e busca de ajuda, se necessário. Reutilizar e remixar permite que os designers usem projetos existentes como ponto de partida, aprendendo com o código de outros e criando projetos mais complexos. Abstrair e modularizar envolve dividir problemas complexos em partes menores e mais gerenciáveis, organizando funcionalidades em blocos de código distintos.

A dimensão das perspectivas computacionais descreve as mudanças na forma como os jovens programadores do Scratch veem a si mesmos, suas relações com o outros e o mundo tecnológico ao seu redor (Brennan; Resnick, 2012). Os autores identificam três perspectivas principais: expressando, conectando e questionando.

A perspectiva de expressar permite que os alunos deixem de ser consumidores passivos e se vejam como criadores, usando a programação como meio de expressão pessoal. A perspectiva de conectar destaca a importância da colaboração, tanto on-line quanto presencialmente, para enriquecer o processo criativo. A perspectiva de questionar incentiva uma postura crítica em relação à tecnologia, motivando os alunos a explorar e usar a programação para investigar o mundo ao redor.

Em síntese, Brennan e Resnick (2012) argumentam que a avaliação do Pensamento Computacional deve abranger conceitos, práticas e perspectivas computacionais. A presença de um elemento de código em um projeto não indica necessariamente uma compreensão profunda do mesmo, destacando a importância de considerar o processo de desenvolvimento, e não apenas o produto final. Para concretizar essa análise, os autores exploram três abordagens principais.

Primeiramente, os autores defendem a análise de portfólio de projetos, empregando ferramentas para visualizar os blocos de programação usados nos projetos dos alunos na plataforma Scratch. A frequência de uso de certos blocos e a progressão na complexidade

dos projetos ao longo do tempo podem indicar a apropriação de conceitos. No entanto, essa abordagem se limita ao produto final, ignorando o processo de desenvolvimento e as dificuldades encontradas.

Além disso, Brennan e Resnick (2012) explanam sobre entrevistas baseadas em artefatos, em que os alunos escolhem projetos próprios para discutir, detalhando o processo de criação, os desafios e as soluções encontradas. Essa interação permite ao educador compreender a fluência do estudante com os conceitos, práticas e estratégias de resolução de problemas, além de revelar nuances que a análise de código não captura. Apesar de rica em detalhes, os autores afirmam que a entrevista depende da memória do aluno e da escolha dos projetos, que podem não representar totalmente o repertório de habilidades.

Por fim, os autores recorrem a cenários de design, propondo desafios de programação sem projetos predefinidos, exigindo que o aluno explique, estenda, corrija erros e modifique o código. Essa abordagem permite observar as práticas de Pensamento Computacional em tempo real, como a depuração e o raciocínio condicional. A desvantagem reside, segundo Brennan e Resnick (2012), na possível falta de engajamento do aluno com projetos externos e na dificuldade de mensurar o tempo e o nível de auxílio necessários durante a atividade.

Contudo, Brennan e Resnick (2012) afirmam que nenhuma dessas abordagens é perfeita ou completa. Os autores recomendam a combinação de múltiplos métodos para uma avaliação mais abrangente, reconhecendo as limitações de tempo e recursos. Eles também destacam a importância de envolver diferentes perspectivas (individual, em grupo, com o auxílio do professor) no processo avaliativo para obter uma visão holística do desenvolvimento do Pensamento Computacional.

Cabe salientar, corroborando as ideias de Raabe, Couto e Blikstein (2020), que a perspectiva de Pensamento Computacional de Brennan e Resnick (2012) se alinha à abordagem construcionista de Papert, enfatizando o aprendizado ativo e a criação de artefatos externos. Assim como na teoria de Papert, Brennan e Resnick (2012) argumentam que a aprendizagem é mais eficaz quando os alunos se envolvem ativamente na construção do conhecimento, em vez de simplesmente consumirem informações passivamente.

A pesquisa de Brennan e Resnick (2012) destaca como o desenvolvimento de projetos de mídia interativa no ambiente Scratch pode promover o Pensamento Computacional. Nesse contexto, os alunos não se limitam a aprender conceitos abstratos, mas os aplicam concretamente na criação de seus projetos. Essa ênfase na construção de artefatos externos encontra ressonância direta na abordagem construcionista, que valoriza a experiência prática e a materialização do conhecimento.

Na sequência, discorre-se sobre a perspectiva de Pensamento Computacional defendida por Dantas (2023), evidenciando como esta habilidade pode ser operacionalizada no processo de resolução de problemas com o software GeoGebra.

2.3.5 Pensamento Computacional na perspectiva de Dantas

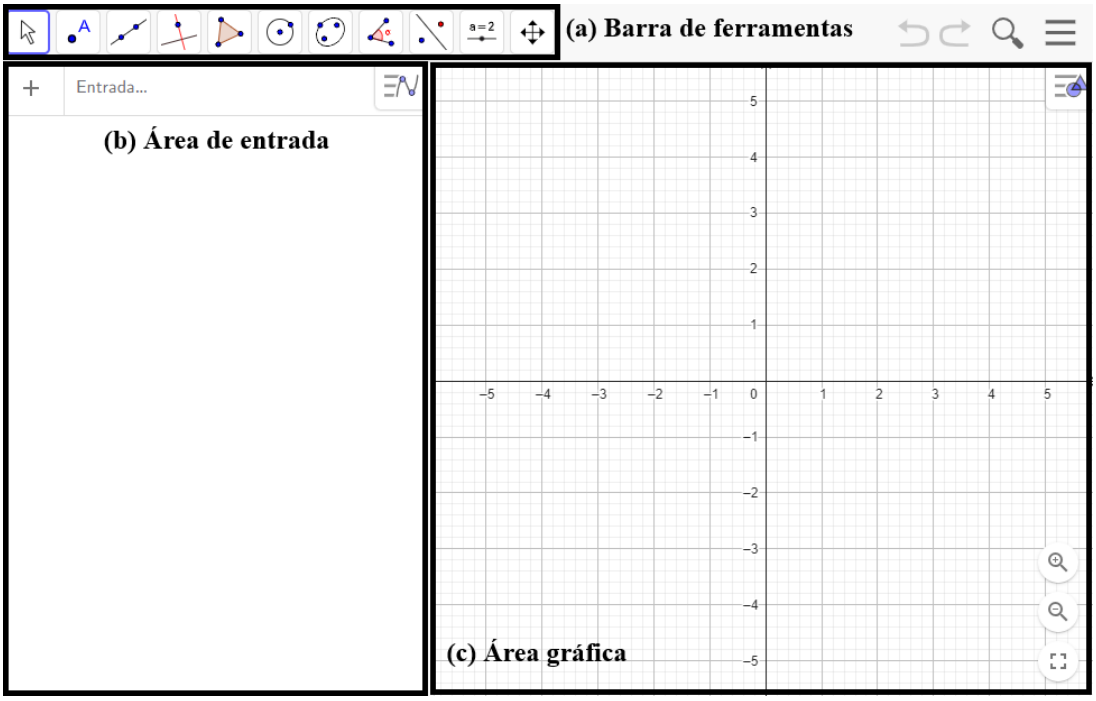
Com base nas investigações conduzidas pelo Grupo de Pesquisa Autômato[29], fundamentadas predominantemente em Dantas (2023), esta seção apresenta uma perspectiva de Pensamento Computacional aplicada à resolução de problemas com o uso do GeoGebra.

De acordo com o site oficial[30] do programa, o GeoGebra é um software matemático dinâmico, destinado a todos os níveis de ensino, que integra geometria, álgebra, planilhas, gráficos, estatísticas e cálculos em uma única plataforma. A Figura 2.18 ilustra o ambiente de trabalho do usuário ao acessar o GeoGebra.

[29] Grupo de Pesquisa e Estudos em Educação Matemática vinculado à Universidade Estadual do Paraná (Unespar), *Campus* de Apucarana, o qual o pesquisador autor deste trabalho faz parte e tem como líder o Prof. Dr. Sérgio Carrazedo Dantas, que orienta esta pesquisa.

[30] Disponível em: https://www.geogebra.org/about. Acesso em: 03 ago. 2024.

Figura 2.18 – Ambiente principal de trabalho no GeoGebra



Fonte: Os autores.

A Figura 2.18 apresenta as três principais áreas da interface do GeoGebra ao iniciar o programa: (a) a barra de ferramentas, localizada na parte superior da tela, que oferece os principais instrumentos para construção e manipulação de objetos geométricos e algébricos; (b) a área de entrada, onde é possível inserir comandos e expressões matemáticas para criar objetos diretamente por texto; e (c) a área gráfica, representada pelo plano cartesiano central, que permite a visualização e manipulação dos objetos em tempo real. Essas áreas formam o ambiente de trabalho principal do usuário no GeoGebra.

Além dessas áreas principais, o GeoGebra possui outras janelas que ampliam suas funcionalidades, como a janela CAS (*Computer Algebra System*) e a janela 3D. A janela CAS permite a realização de cálculos algébricos avançados, manipulação simbólica de expressões e resolução de equações, sendo particularmente útil em atividades que envolvem álgebra e cálculo. A janela 3D, por sua vez, possibilita a criação e manipulação de objetos tridimensionais, oferecendo uma representação visual de sólidos geométricos e permitindo a exploração de conceitos de geometria espacial.

No âmbito do Pensamento Computacional, Dantas (2022) caracteriza esse modo de pensar, com base na perspectiva de Brackmann (2017), como um conjunto de quatro

ações mentais não hierárquicas: decomposição, reconhecimento de padrões, abstração e algoritmos[31]. Segundo o autor, o Pensamento Computacional "[...] ocupa-se do tratamento de entes abstratos em interface com o pensamento matemático, na busca da resolução de problemas através de uma série de etapas que possam ser executadas por um agente humano ou por um dispositivo digital" (Dantas, 2022, p. 24).

Em um estudo posterior, Dantas (2023) incorpora o Ciclo de Ações e a Espiral de Aprendizagem, de Valente (2005), à sua perspectiva de Pensamento Computacional. Essa abordagem possibilita uma reflexão contínua sobre as próprias ações durante e após a resolução de problemas, compreendendo cada etapa como processos interativos e integrados. O ciclo é composto pelas etapas de formulação, descrição, execução, análise/reflexão e depuração (Valente, 2005), representando um processo iterativo de aprendizagem na resolução de problemas com o GeoGebra. A Espiral de Aprendizagem emerge dessa interação entre o aluno e a ferramenta digital, à medida que o aluno formula, descreve, executa, analisa e depura seu raciocínio (Dantas, 2023).

Ao contextualizar a Espiral de Aprendizagem de Valente (2005) nas discussões sobre Pensamento Computacional, Dantas (2023) questiona a adequação de sintetizar esse modo de pensar em apenas quatro processos mentais, como frequentemente descrito na literatura (Brackmann, 2017; Boucinha et al., 2019; Barros, 2020; Bertazini, 2022). O autor propõe uma abordagem ampliada, com seis processos mentais, incorporando a formulação do problema e a depuração como processos essenciais para operar com o Pensamento Computacional. Na sequência, discute-se sobre esses processos mentais.

A formulação do problema, segundo Dantas (2023), envolve a concepção e estruturação do problema em termos de necessidade, objetivo, questionamento e resolução, considerando as variáveis e ações necessárias. Esse processo abrange mental também o planejamento de ações, valendo-se de técnicas e repertórios específicos (Prado, 2024).

A depuração, por sua vez, foca na identificação e correção de erros, englobando as

[31] Como em Dantas (2022) esses processos são descritos na perspectiva de Brackmann (2017), a qual foi discutida em seções anteriores, opta-se por não discorrer sobre esses pilares nesta seção. Ao leitor interessado em compreender esses processos, recomenda-se voltar à seção anterior ou a leitura do trabalho de Brackmann (2017).

ações de testagem, verificação, refinamento e otimização da resolução apresentada (Dantas, 2023). Durante esse processo, os resultados, parciais ou finais, são avaliados, hipóteses são descartadas, e as ideias são refinadas, revisadas, reformuladas ou abandonadas. Em suma, a depuração constitui um movimento de autorreflexão sobre as próprias ações e processos mentais, visando garantir que esses processos estejam alinhados com o objetivo inicial (Teixeira; Juvanelli; Dantas, 2024).

Na perspectiva de Pensamento Computacional proposta por Dantas (2023), esse modo de pensar, associado ao uso do GeoGebra, revela-se uma ferramenta eficaz para a resolução de problemas matemáticos. O autor argumenta que o Pensamento Computacional não se limita ao uso de Tecnologias Digitais, manifestando-se também em processos de resolução manuscritos. No entanto, é através da integração com o GeoGebra que esse tipo de pensamento se torna especialmente potente.

Desse modo, para exemplificar como o Pensamento Computacional pode ser operacionalizado por meio do software GeoGebra, discute-se a seguir sobre um dos problemas apresentados em Dantas (2023). O autor aborda a construção e determinação de todas as diagonais de um polígono convexo no GeoGebra. Contudo, neste trabalho, apresenta-se um problema ligeiramente diferente: *construir e determinar a quantidade de diagonais de qualquer polígono convexo sem sobreposição de diagonais no GeoGebra.* A seguir, elucidam-se os passos necessários para a resolução do problema considerado.

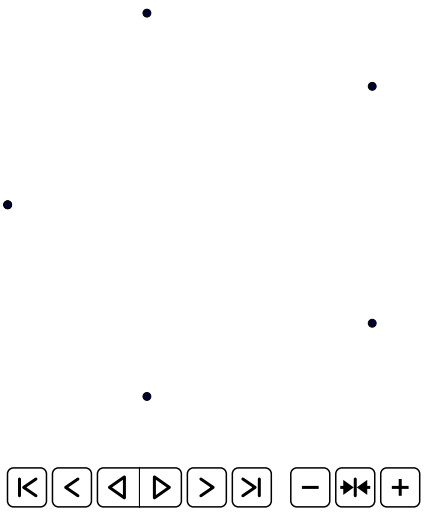
1º Passo: construir um controle deslizante n, com valor mínimo 3, valor máximo 20 e incremento 1.

2º Passo: utilizar o comando

```
L=Sequência(Girar((-1,0),i*2*π/n),i,0,n)
```

para construir uma sequência de pontos igualmente distribuídos (girados) em torno da origem do plano cartesiano. A Figura 2.19 ilustra alguns resultados obtidos no ambiente do GeoGebra.

Figura 2.19 – Exemplos de pontos igualmente distribuídos em torno da origem do plano cartesiano

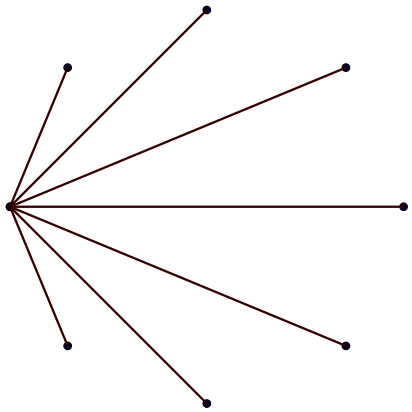


Fonte: Os autores.

Antes de prosseguir para o próximo passo, é conveniente elucidar o processo de generalização de geração dos polígonos, seus lados e diagonais.

Considere o exemplo de uma sequência com $n = 8$ pontos. Traçam-se todos os segmentos partindo primeiro elemento da lista, $L(1)$, até o último, $L(8)$. Pode-se imaginar que $L(1)$ estará ligado a $L(2)$, $L(3), ..., L(n)$, ou seja, a todos os outros $n - 1$ vértices do polígono. O resultado implementado no GeoGebra é apresentado na Figura 2.20.

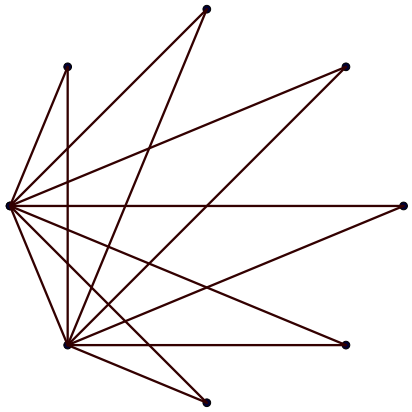
Figura 2.20 – Segmentos que ligam $L(1)$ aos demais pontos de L



Fonte: Os autores.

Na sequência, constroem-se todos os segmentos que partem de $L(2)$ para os demais vértices, desconsiderando a ligação desse com $L(1)$, pois esse segmento já foi construído no processo anterior. O resultado obtido é apresentado na Figura 2.21.

Figura 2.21 – Segmentos que ligam $L(1)$ e $L(2)$ aos demais pontos de L



Fonte: Os autores.

Esse mesmo processo deve ser realizado para o caso de $L(3)$, não ligando a $L(1)$ e nem a $L(2)$ e assim sucessivamente, para $L(4)$ até $L(n)$.

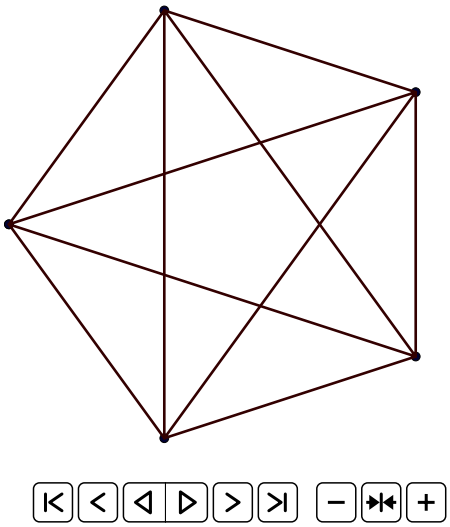
Esse processo de construção pode ser generalizado por meio da utilização de dois comandos Sequência aninhados. Dessa forma, o primeiro comando varia o vértice tomado como referência, enquanto o segundo comando varia os vértices aos quais o vértice de referência é conectado por segmentos. Essa abordagem permite organizar o próximo passo.

Passo 3: construir todos os vértices com o comando

```
D=Sequência(Sequência(Segmento(L(i),L(j)),j,i+1,n),i,1,n-1).
```

Esse comando permite obter todos os lados e diagonais de um polígono de n lados. A Figura 2.22 apresenta o resultado obtido, considerando polígonos de cinco até vinte lados, ou seja, $n = 5$ até $n = 20$.

Figura 2.22 – Lados e diagonais de polígonos de cinco até vinte lados



Fonte: Os autores.

A construção apresentada nos passos 1, 2 e 3 permite obter todos os lados e diagonais de um polígono. Contudo, para obter uma expressão generalizada o cálculo de diagonais de um polígono convexo, são necessárias algumas manipulações adicionais. Esse processo é descrito a seguir, nos passos 4 e 5.

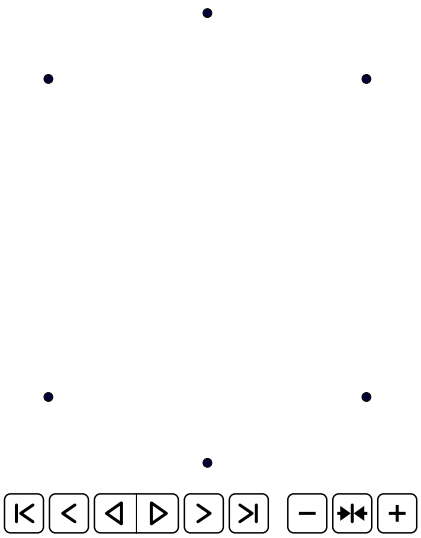
Passo 4: construir um controle deslizante m, com valor mínimo 0, valor máximo $n-1$ e incremento 1;

Passo 5: modificar o comando do *Passo 3* para

```
D=Sequência(Sequência(Segmento(L(i),L(j)),j,i+1,n),i,1,m).
```

Por meio desses incrementos na construção, é possível manipular o controle deslizante m, obtendo-se os segmentos que partem dos vértices $L(1)$ quando $m = 1$, $L(1)$ e $L(2)$ quando $m = 2$, e assim sucessivamente. A Figura 2.23 apresenta o resultado obtido, considerando um polígono de 8 lados ($n = 8$).

Figura 2.23 – Lados e diagonais de um octógono



Fonte: Os autores.

Observa-se, na Figura 2.23, que a quantidade de segmentos varia conforme os valores de m também varia. Desse modo, é possível estabelecer uma relação entre o valor de m e a quantidade de segmentos, conforme apresenta o Quadro 2.3.

Quadro 2.3 – Quantidade de segmentos para valores de m

m	**Segmentos**
1	7
2	$7+6$
3	$7+6+5.$
4	$7+6+5+4$
5	$7+6+5+4+3$
6	$7+6+5+4+3+2$
7	$7+6+5+4+3+2+1$

Fonte: Elaborado pelo autor (2025).

A Partir do exposto no Quadro 2.3, pode-se notar que, para um polígono de n lados, há $(n-1)+(n-2)+...+1$ segmentos. Como n segmentos são os lados do polígono, subtrai-se n da expressão anterior, obtendo-se, após algumas substituições e simplificações, a expressão $D = n(n-3)/2$, geralmente abordada em livros didáticos de Matemática.

A sequência de passos elucidada nos parágrafos anteriores apresenta o processo de obtenção de uma expressão que determina a quantidade de diagonais em um polígono convexo com o auxílio do software GeoGebra. Dantas (2023) argumenta que o processo de resolução de problemas como este é a operacionalização do Pensamento Computacional.

Desse modo, considerando as dimensões do Pensamento Computacional na perspectiva do Grupo de Pesquisa Autômato, a saber, formulação do problema, decomposição, reconhecimento de padrões, abstração, produção de algoritmos e depuração, identifica-se, na sequência, cada um desses processos mentais na resolução do problema considerado.

A formulação do problema ocorre no processo de idealizar uma solução geral e sistemática para o problema de construir e determinar a quantidade de diagonais de qualquer polígono convexo sem sobreposições de diagonais por meio do software GeoGebra. Esse processo de idealização envolveu a análise das ferramentas oferecidas pelo software, como a criação de controles deslizantes, sequências e segmentos, além de considerar a possibilidade de manipular variáveis visuais e geométricas de maneira dinâmica. Dessa forma, optou-se por um controle deslizante para ajustar o número de vértices do polígono e uma sequência para gerar pontos igualmente espaçados em torno da origem, facilitando a construção do polígono. Essas escolhas permitiram que o problema fosse formulado de modo a explorar plenamente os recursos do GeoGebra.

A decomposição é evidenciada no processo de dividir o problema, inicialmente complexo, em partes menores. Inicialmente, idealizou-se que o objetivo seria construir uma solução que permitisse obter a quantidade de diagonais de qualquer polígono convexo. Para isso, construiu-se um controle deslizante, que permitiria variar seus valores com maior facilidade. Na sequência, construiu-se uma sequência de pontos que deveriam corresponder aos vértices do polígono, dependendo do controle deslizante construído anteriormente. Por fim, para obter a solução final, construiu-se todos os lados e diagonais dos polígonos com base na sequência de pontos construída anteriormente.

O reconhecimento de padrões pode ser identificado em ao menos dois aspectos distintos. Primeiramente, destaca-se o reconhecimento de padrões ao fixar um dos vértices e variar os outros para obter os lados e diagonais dos polígonos. Assim, ao fixar o primeiro vértice, deve-se considerar o próximo e variar os demais vértices, desconsiderando os segmentos considerados nos processos anteriores. Outro padrão fundamental para a obtenção do modelo final foi a identificação dos padrões da quantidade de segmentos possíveis de se gerar a partir de um polígono, seguindo um padrão de soma.

A abstração é identificável no processo ao se desconsiderar as complexidades envolvidas em polígonos não-regulares, concentrando-se exclusivamente nas propriedades geométricas de polígonos regulares. Esse enfoque permitiu a utilização de técnicas geométricas implementadas no software para construir a sequência de pontos apresentada no *Passo 2*. Aspectos irrelevantes, como o tamanho dos lados dos polígonos e a cor dos objetos, não foram considerados durante a construção. Além disso, a maneira como os comandos foram plotados no software é um exemplo de abstração, pois os objetos matemáticos são criados com base em comandos que não necessariamente consideram todas as características destes objetos.

A organização do processo de obtenção da solução em cinco passos claros, precisos e sistemáticos explicita a importância da abordagem algorítmica para obter a solução final almejada.

O processo de depuração não é visível em um processo cristalizado, conforme apresentado ao longo dos parágrafos, pois, ao escrever a resolução conforme apresentada, foram eliminadas as dúvidas e possíveis erros cometidos em cálculos ou em suposições, e é exatamente nesse processo que a depuração acontece. Contudo, a reflexão e verificação contínua das implementações computacionais garantiram que o resultado estivesse alinhado com o projeto mental idealizado inicialmente. Durante o processo de construção houve alterações nos comandos implementados no software. Essas modificações foram realizadas após a verificação de que seria necessário incluir um segundo controle deslizante para esboçar e manipular, a cada incremento, os segmentos que partem de cada vértice do polígono.

Com base no exposto nesta seção, pode-se concluir que a abordagem do Pensamento Computacional delineada pelo Grupo de Pesquisa Autômato oferece uma perspectiva para a resolução de problemas matemáticos, utilizando Tecnologias Digitais como o software GeoGebra. A resolução do problema de construção e determinação das diagonais de polígonos convexos ilustra como os conceitos de Pensamento Computacional podem ser operacionalizados de maneira prática.

Na sequência, realizam-se algumas considerações sobre as perspectivas de Pensa-

mento Computacional apresentadas nesta seção.

2.3.6 Considerações sobre ascensão e perspectivas de Pensamento Computacional

Esta seção teve como objetivo apresentar e problematizar perspectivas de Pensamento Computacional a partir de Wing (2006), com ênfase nos impactos educacionais dessas perspectivas. A análise do contexto permitiu evidenciar as transformações que influenciaram o debate sobre o Pensamento Computacional na Educação.

Para fundamentar a argumentação, foram abordadas as perspectivas de pesquisadores de diferentes áreas, como Wing (2006, 2008, 2011, 2017), Brackmann (2017), Denning e Tedre (2019), Brennan e Resnick (2012) e Dantas (2023). A apresentação dessa pluralidade de visões possibilitou identificar pontos de convergência e divergência entre os autores. Em linhas gerais, todas as abordagens ressaltam a importância do Pensamento Computacional no século XXI, mas divergem quanto aos pressupostos epistemológicos e às aplicações educacionais.

Embora a perspectiva de Wing (2006, 2008, 2011, 2017) ganhe contornos específicos e pragmáticos em relação às habilidades abrangidas pelo Pensamento Computacional, observa-se que as discussões sobre a implementação desse ensino na Educação Básica ainda carecem de aprofundamento. Todavia, a contribuição de Wing foi essencial para consolidar o Pensamento Computacional como campo de estudo, organizando conhecimentos que fundamentaram o avanço das pesquisas e práticas educacionais na área.

No contexto brasileiro, Brackmann (2017) ampliou as discussões sobre Pensamento Computacional, especialmente em cenários com recursos tecnológicos limitados. O autor defende a abordagem desplugada como estratégia eficaz para o ensino do Pensamento Computacional, promovendo a resolução de problemas por meio da lógica, criatividade e atividades práticas.

Os trabalhos de Wing (2006, 2008, 2011, 2017) e Brackmann (2017) contribuíram para um movimento significativo entre pesquisadores interessados na integração do Pensamento Computacional na Educação. Entretanto, há um debate restrito sobre o

contexto político e econômico que motivou o ressurgimento do conceito após o ensaio de Wing em 2006. Valente et al. (2017) alertam para a influência de interesses comerciais, particularmente de grandes empresas do Vale do Silício, e defendem a necessidade de um debate mais amplo sobre os fatores que impulsionaram a rápida disseminação do Pensamento Computacional.

Denning e Tedre (2019) reforçam a importância de construir um entendimento sólido e abrangente do Pensamento Computacional na Educação, evitando simplificações. Os autores destacam pontos de controvérsia sobre a definição e os objetivos do ensino do Pensamento Computacional, como a relação entre algoritmos e abstração, a dependência do contexto e a transferibilidade das habilidades. Eles defendem uma abordagem pluralista que reconheça a diversidade de perspectivas dentro da área da Computação, sem perder de vista os rigores da disciplina.

Para Denning e Tedre (2019), é crucial distinguir entre Pensamento Computacional para iniciantes e para profissionais, evitando expectativas irreais quanto ao alcance dessas habilidades, especialmente no nível iniciante. Os autores argumentam que o domínio de ferramentas computacionais não implica automaticamente o desenvolvimento de habilidades avançadas de resolução de problemas.

Com o propósito de apresentar perspectivas operacionais sobre o Pensamento Computacional, destacaram-se duas abordagens que fundamentaram o uso de Tecnologias Digitais nesse contexto. Brennan e Resnick (2012) investigam como o desenvolvimento de projetos de mídia interativa no Scratch pode promover o Pensamento Computacional. No entanto, os autores enfatizam que não existe uma única forma de avaliar essa habilidade em sua totalidade. Assim, recomendam uma avaliação abrangente que considere conceitos, práticas e perspectivas computacionais, sugerindo a combinação de múltiplos métodos para obter uma análise mais completa.

Dantas (2023) propõe uma perspectiva operacional de Pensamento Computacional utilizando o software GeoGebra. O exemplo apresentado e desenvolvido ilustra como os conceitos de Pensamento Computacional podem ser operacionalizados de maneira prática, por meio da experimentação, visualização e manipulação de objetos matemáticos, o que

pode incentivar o desenvolvimento do Pensamento Computacional de forma prática.

Na sequência, realizam-se algumas considerações sobre a inserção do Pensamento Computacional no contexto educacional nacional.

2.4 A BNCC como catalisadora da ascensão do Pensamento Computacional

No contexto educacional brasileiro, os debates sobre a inserção do Pensamento Computacional ganharam destaque com a promulgação da Base Nacional Comum Curricular (BNCC) em 2017. Esse documento normativo estabelece um conjunto de competências e habilidades a serem desenvolvidas ao longo da Educação Básica, orientando práticas pedagógicas e curriculares em todo o país (Brasil, 2018).

Apesar de sua relevância, a BNCC tem sido alvo de diversas críticas desde suas primeiras versões, especialmente por parte de professores e pesquisadores da área da Educação. Entre os questionamentos mais recorrentes, destaca-se a adoção de uma perspectiva tecnocrata e instrumentalista da tecnologia, que enfatiza uma visão funcional e acrítica dessas habilidades, frequentemente alinhada a interesses mercadológicos e de instituições privadas (Valente et al., 2017; SBC, 2018). Essa abordagem, ao reduzir a tecnologia a uma ferramenta de eficiência, desconsidera suas implicações sociais, éticas e pedagógicas, além de negligenciar as desigualdades socioeconômicas que limitam o acesso às Tecnologias Digitais.

A relação entre tecnologia e Educação exige, portanto, uma análise crítica das concepções que orientam essa integração. Vargas, Vicente e Dantas (2022) argumentam que as políticas educacionais brasileiras frequentemente incorporam demandas econômicas e ideológicas de países desenvolvidos, respondendo a um modelo de economia neoliberal que subordina o papel social do indivíduo às forças produtivas do capital. Nesse contexto, a inserção de tecnologias na Educação tende a ser determinada por organizações internacionais e orientada por uma racionalidade instrumental, que privilegia a eficiência técnica em detrimento de uma reflexão crítica sobre seu impactos sociais e educacionais (Peixoto; Moraes, 2017).

Diante desse cenário, esta seção tem como objetivo discorrer sobre a inserção do

Pensamento Computacional no contexto educacional nacional, conforme estabelecido pela BNCC. Busca-se evidenciar e problematizar a perspectiva de Pensamento Computacional presente no documento, considerando que sua abordagem pode levar a uma implementação fragmentada e pouco reflexiva no currículo escolar.

Para sustentar essas discussões, apresentam-se as contribuições da Sociedade Brasileira de Computação (SBC), que teve participação ativa no processo de elaboração da BNCC, tecendo notas técnicas com propostas de habilidades e competências relacionadas à Computação. Além disso, são analisadas as perspectivas de pesquisadores da Educação Matemática (Dantas, 2022; Lirio; Prado, 2023; Plewka; Dantas, 2023), que apontam lacunas conceituais e metodológicas na abordagem do Pensamento Computacional e das Tecnologias Digitais no documento.

A estrutura desta seção segue a seguinte organização: inicialmente, apresenta-se o processo de elaboração da BNCC e a forma como o documento incorpora o Pensamento Computacional na área da Matemática e suas Tecnologias; em seguida, discutem-se as principais críticas direcionadas ao documento, com ênfase na abordagem das Tecnologias Digitais e do Pensamento Computacional.

2.4.1 O Pensamento Computacional na Base Nacional Comum Curricular

A BNCC é um documento de caráter normativo que

> [...] define o conjunto orgânico e progressivo de aprendizagens essenciais que todos os alunos devem desenvolver ao longo das etapas e modalidades da Educação Básica, de modo a que tenham assegurados seus direitos de aprendizagem e desenvolvimento, em conformidade com o que preceitua o Plano Nacional de Educação (PNE) (Brasil, 2018, p. 7).

O processo de elaboração da BNCC teve início em 2012 e foi marcado por um percurso longo e controverso. Prado (2024) aponta que o documento recebeu críticas relacionadas ao seu viés político e ideológico, sendo frequentemente associado a demandas mercadológicas e interesses de instituições privadas. Nesse sentido, Valente et al. (2017) destacam que essa característica reflete uma tendência histórica da Educação brasileira de alinhar-se a interesses externos ao campo educacional, priorizando a formação de capital

humano em detrimento de uma abordagem crítica e reflexiva.

Para compreender como a BNCC incorporou o Pensamento Computacional, é necessário, primeiramente, analisar a forma como as Tecnologias Digitais são apresentadas no documento. Essa análise permite contextualizar o papel dessas tecnologias na proposta curricular e a maneira como o Pensamento Computacional se insere na área de Matemática e suas Tecnologias (Reis; Barichello; Mathias, 2021).

Entre as dez competências[32] gerais estabelecidas para a Educação Básica, quatro mencionam explicitamente o uso de Tecnologias Digitais no processo de ensino e aprendizagem:

> 1. Valorizar e utilizar os conhecimentos historicamente construídos sobre o mundo físico, social, cultural e **digital** para entender e explicar a realidade; 2. Recorrer à abordagem própria das ciências [...] para investigar causas, elaborar e testar hipóteses, formular e resolver problemas e criar soluções (inclusive **tecnológicas**) com base nos conhecimentos das diferentes áreas. [...] 4. Utilizar diferentes linguagem – verbal, corporal, visual, sonora e **digital** [...] para se expressar e produzir sentidos que levem ao entendimento mútuo. 5. Compreender, utilizar e criar **tecnologias digitais** de informação e comunicação de forma crítica [...] (Brasil, 2018, p. 9, grifo nosso).

Essas competências refletem a crescente importância atribuída às Tecnologias Digitais na Educação Básica. Contudo, o destaque conferido a essas tecnologias não se limita às competências gerais, mas se estende a diversas habilidades[33] específicas no âmbito da Matemática e suas Tecnologias. O documento enfatiza o uso de ferramentas digitais em tópicos como álgebra, geometria, probabilidade, estatística e resolução de problemas. Um exemplo dessa abordagem pode ser encontrado na habilidade de "resolver e elaborar problemas, envolvendo o cálculo de porcentagens, incluindo o uso de **tecnologias digitais**" (Brasil, 2018, p. 313, grifo nosso), apresentada como parte essencial do desenvolvimento das competências matemáticas.

Reis, Barichello e Mathias (2021) apontam que a recorrência de referências a

[32] De acordo com a BNCC, uma competência é definida como a "mobilização de conhecimentos (conceitos e procedimentos), habilidades (práticas, cognitivas e socioemocionais), atitudes e valores para resolver demandas complexas da vida cotidiana, do pleno exercício da cidadania e do mundo do trabalho" (Brasil, 2018, p. 8).

[33] Segundo a BNCC, "as habilidades expressam as aprendizagem essenciais que devem ser asseguradas aos alunos nos diferentes contextos escolares" (Brasil, 2018, p. 29).

softwares, ferramentas e recursos digitais no documento sugerem um incentivo ao uso das Tecnologias Digitais como mediadoras do ensino e da aprendizagem em Matemática. Entretanto, essas referências nem sempre são acompanhadas de diretrizes detalhadas sobre como integrar essas tecnologias de forma crítica e reflexiva, o que pode reforçar uma abordagem funcionalista da Educação.

No que se refere especificamente ao Pensamento Computacional, a BNCC o apresenta como um conceito central na área de Matemática e suas Tecnologias, justificando que "[...] os processos matemáticos de resolução de problemas, de investigação, de desenvolvimento de projetos e da modelagem [...] são potencialmente ricos para o desenvolvimento do **pensamento computacional**" (Brasil, 2018, p. 266, grifo nosso).

Além disso, o documento define o Pensamento Computacional como uma habilidade que "[...] envolve as capacidades de compreender, analisar, definir, modelar, resolver, comparar e automatizar problemas e suas soluções, de forma metódica e sistemática, por meio do desenvolvimento de algoritmos" (Brasil, 2018, p. 474).

No Ensino Fundamental, a BNCC incentiva o uso de tecnologias como calculadoras e planilhas eletrônicas desde os anos iniciais, com o objetivo de introduzir o Pensamento Computacional por meio da interpretação e elaboração de algoritmos, frequentemente representados por fluxogramas. Essa abordagem, embora coerente com a perspectiva funcionalista do documento, tem sido criticada por sua superficialidade e falta de aprofundamento crítico, como será discutido na sequência.

No Ensino Médio, a BNCC amplia essa abordagem, apresentando as Tecnologias Digitais como ferramentas fundamentais para investigar, modelar e solucionar problemas complexos, utilizando raciocínio lógico e Pensamento Computacional. Todavia, conforme discutido por Reis, Barichello e Mathias (2021), a proposta ainda carece de uma fundamentação epistemológica mais robusta, o que pode limitar sua efetividade na prática docente e o impacto no desenvolvimento das habilidades propostas.

Embora a BNCC reconheça o Pensamento Computacional como uma habilidade essencial para a formação dos estudantes, especialmente na área de Matemática e suas Tecnologias, sua abordagem tem suscitado críticas de diversas frentes, incluindo pesquisadores

da Sociedade Brasileira de Computação (SBC) e educadores matemáticos. Essas críticas apontam para limitações e contradições na forma como o documento incorpora o Pensamento Computacional e as Tecnologias Digitais, evidenciando uma visão instrumentalista e uma carência de aprofundamento epistemológico.

A próxima seção discute essas perspectivas, analisando os desafios e as implicações dessa abordagem no contexto educacional brasileiro.

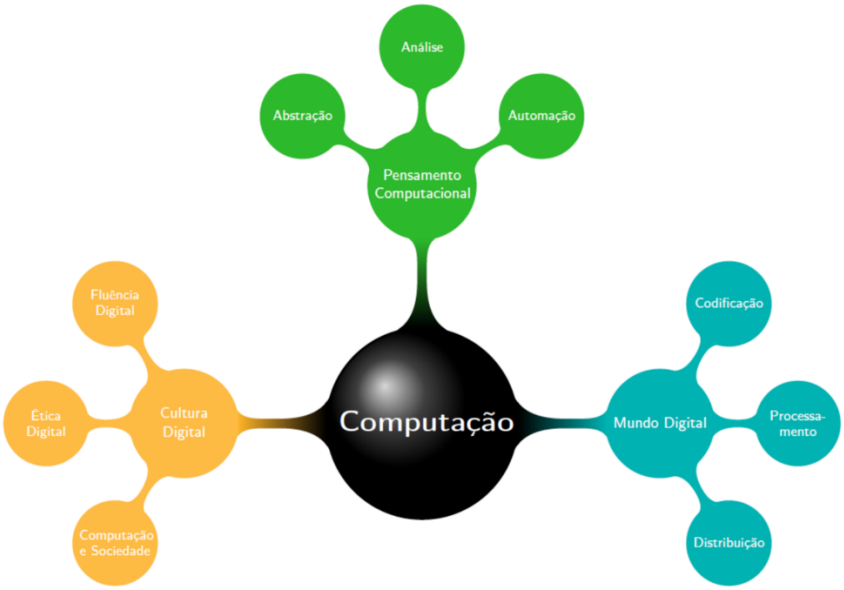
2.4.2 Críticas a Base Nacional Comum Curricular

A relação entre o Pensamento Computacional, conforme descrito na BNCC, e sua aplicação em resolução de problemas, questões algébricas e elaboração de fluxogramas sustenta a escolha da Matemática como a área responsável pelo desenvolvimento desse modo de pensar (Prado, 2024). No entanto, o documento concentra-se quase que exclusivamente na aplicação prática do Pensamento Computacional na resolução de problemas, negligenciando aspectos como abstração, decomposição e reconhecimento de padrões, frequentemente associados a uma definição mais abrangente de Pensamento Computacional, conforme discutido neste trabalho.

Desde sua concepção, a BNCC tem sido alvo de críticas em diversos aspectos. No escopo desta pesquisa, limitam-se as discussões às críticas relacionadas à perspectiva de Pensamento Computacional presente no documento. Durante o processo de elaboração da BNCC, a Sociedade Brasileira de Computação (SBC) desempenhou um papel ativo, promovendo diálogos com especialistas em Computação e publicando notas técnicas que propunham habilidades, competências e objetos de conhecimento a serem incluídos no documento (SBC, 2017, 2018).

Na primeira nota técnica enviada aos elaboradores da BNCC, a SBC apresentou uma proposta estruturada em três eixos fundamentais para a área da Computação: Pensamento Computacional, Mundo Digital e Cultura Digital. Esses eixos foram idealizados para englobar conhecimentos essenciais da Computação, situando o Pensamento Computacional como uma habilidade central. A Figura 2.24 ilustra esses eixos.

Figura 2.24 – Eixos da Computação na perspectiva da Sociedade Brasileira da Computação



Fonte: SBC (2017).

Segundo a SBC (2017), o Pensamento Computacional envolve a capacidade de resolver problemas sistematicamente, utilizando modelos e representações abstratas para criar soluções algorítmicas que possam ser executadas por máquinas. O Mundo Digital compreende a estrutura e o funcionamento do ambiente digital, incluindo conceitos como codificação de informações, armazenamento, transmissão e segurança. O eixo da Cultura Digital, por sua vez, abrange o impacto da revolução digital na sociedade e destaca o uso crítico e ético das ferramentas digitais, envolvendo aspectos como direitos autorais, segurança digital e interações humano-computador.

Apesar de apresentar essas propostas em audiências públicas e em documentos técnicos, a SBC não teve suas sugestões incorporadas na versão final da BNCC, homologada em 2017. Posteriormente, em 2018, a SBC publicou outra nota técnica, na qual manifestava discordância com a maneira como a BNCC abordava as habilidades relacionadas à Computação, tanto no Ensino Fundamental quanto no Ensino Médio (SBC, 2018). A nota argumenta que o documento apresenta uma visão fragmentada e limitada do Pensamento Computacional, além de falhar em reconhecer a Computação como uma área de conhecimento autônoma e relevante.

No Ensino Fundamental, as críticas da SBC incluem a ênfase inadequada no uso

de fluxogramas para representar algoritmos, a formulação excessivamente específica e mal estruturada de habilidades, a ausência de integração entre habilidades e objetos de conhecimento e a omissão de aspectos fundamentais do Mundo Digital. A SBC (2018) destaca que o uso de fluxogramas como linguagem de representação é inadequado, argumentando que se trata de uma abordagem ultrapassada que não segue paradigmas de programação estruturada nem estimula técnicas como decomposição, generalização e transformação.

No Ensino Médio, as críticas da SBC reforçam os problemas apontados para o Ensino Fundamental e acrescentam novas preocupações. Destacam-se a ênfase desproporcional em linguagens de programação em detrimento da criação de algoritmos, a ideia equivocada de que problemas podem ser "algorítmicos" – quando, na realidade, as soluções é que são –, e a dificuldade de determinar se um problema possui ou não uma solução algorítmica.

Pesquisas no âmbito da Educação Matemática, como as de Dantas (2022), Lirio e Prado (2023) e Plewka e Dantas (2023), destaca uma série de críticas à forma como o Pensamento Computacional foi integrado à BNCC. Entre as principais preocupações desses autores, estão a falta de clareza conceitual, a escassez de orientações pedagógicas e uma abordagem superficial que subestima o potencial do Pensamento Computacional e das Tecnologias Digitais na Educação.

Corroborando as críticas da SBC (2018), Dantas (2022) argumenta que, embora a BNCC mencione o uso de Tecnologias Digitais, sua abordagem é superficial e não demonstra um compromisso efetivo com a integração entre Matemática e tecnologia. O autor critica a ausência de discussões aprofundadas sobre o potencial pedagógico de ferramentas como softwares de geometria dinâmica e planilhas eletrônicas, que poderiam desempenhar um papel relevante na construção de conceitos matemáticos.

Lirio e Prado (2023) destacam a elaboração apressada da BNCC e a limitada participação democrática no processo de construção do documento. Segundo os autores, cartas, notas e críticas enviadas por professores, pesquisadores e especialistas da Educação foram desconsideradas pelos formuladores da base, evidenciando uma falta de diálogo que compromete sua legitimidade. Nesse sentido, interpretam a BNCC como um reflexo de uma

visão neoliberal da Educação, caracterizada pela ênfase em avaliações em larga escala e na preparação para o mercado de trabalho. Essa perspectiva, alertam os autores, é reforçada pela influência de instituições empresariais na elaboração do documento, priorizando interesses econômicos em detrimento de uma abordagem educativa mais ampla e inclusiva.

Além disso, Lirio e Prado (2023) e Plewka e Dantas (2023) criticam a falta de conexão entre as habilidades tecnológicas listadas na BNCC e a prática docente. Para esses autores, o documento estabelece uma relação mecânica entre Tecnologias Digitais e habilidades, desconsiderando a complexidade do processo de ensino e aprendizagem. Essa desconexão pode gerar incertezas na implementação do Pensamento Computacional, especialmente diante da ausência de diretrizes pedagógicas claras.

Na sequência, discutem-se as considerações sobre a inserção do Pensamento Computacional no contexto educacional nacional.

2.4.3 Considerações sobre a BNCC como catalisadora da ascenção do Pensamento Computacional

A BNCC desempenhou um papel central na ascensão do Pensamento Computacional no cenário educacional brasileiro. Ao incorporá-lo como uma habilidade essencial para o desenvolvimento dos estudantes, a BNCC consolidou o conceito como parte do currículo formal e ampliou sua relevância no campo da Matemática e em outras áreas do conhecimento. Contudo, a forma como essa inserção ocorreu levanta questionamentos sobre sua efetividade e os desafios envolvidos na implementação.

Um dos principais méritos da BNCC foi reconhecer a importância do Pensamento Computacional como competência transversal, enfatizando sua relação com a resolução de problemas e a automação de processos. Essa abordagem aproxima o ensino da Matemática e da Computação, potencializando o uso de Tecnologias Digitais como ferramentas para a construção do conhecimento. Não obstante, conforme apontam Dantas (2022), Lirio e Prado (2023) e Plewka e Dantas (2023), a falta de clareza conceitual e a ausência de orientações pedagógicas detalhadas dificultam a aplicação efetiva desse conceito na prática docente.

Além disso, a BNCC pode ser vista como um catalisador que impulsionou políticas públicas, programas de formação docente e o desenvolvimento de materiais didáticos voltados ao Pensamento Computacional. No entanto, a implementação dessas iniciativas ainda enfrenta entraves, como a desigualdade no acesso a recursos tecnológicos e a ênfase em uma abordagem instrumentalizada das Tecnologias Digitais, que pode esvaziar a riqueza conceitual subjacente ao Pensamento Computacional.

Dessa forma, a consolidação do Pensamento Computacional na Educação Básica requer um esforço contínuo para garantir sua abordagem de maneira crítica e significativa. Isso envolve ampliar o debate sobre as concepções que orientam sua inserção no currículo, assegurando que essa integração vá além da mera adoção de ferramentas tecnológicas e contribua, de fato, para a construção de conhecimentos e para o desenvolvimento do Pensamento Computacional nos estudantes.

Na sequência, realizam-se as considerações finais deste capítulo.

2.5 Considerações deste capítulo

A expressão Pensamento Computacional nunca foi definida de forma precisa. Segundo Raabe, Couto e Blikstein (2020), a literatura não esclarece se o Pensamento Computacional representa um novo tipo de pensamento, uma combinação de pensamentos existentes ou sequer se é adequado chamá-lo de "pensamento". Essa indefinição dificulta o debate sobre como desenvolver, mensurar e avaliar o Pensamento Computacional, e, consequentemente, limita sua disseminação em outras áreas, especialmente na Educação.

Apesar das dificuldades filosóficas e epistemológicas associadas ao conceito de Pensamento Computacional, este capítulo buscou explorar discussões e reflexões sobre o tema, organizando-se em três seções para evidenciar diferentes perspectivas ao longo da história.

Na primeira seção, realizou-se um levantamento histórico, abrangendo desde possíveis origens do Pensamento Computacional em métodos que remontam à Grécia antiga até a invenção dos computadores eletrônicos no século XX. Embora não se tenha explicitado quais práticas cognitivas constituem esse tipo de pensamento, o contexto evidenciou que

diversos conceitos, práticas e habilidades essenciais às perspectivas contemporâneas de Pensamento Computacional já estavam presentes em métodos e procedimentos históricos, assim como na elaboração de máquinas de computação anteriores à era moderna.

A segunda seção abordou as mudanças na percepção dos computadores por cientistas e pesquisadores entre as décadas de 1950 e 1990. Nesse ínterim, diversas tentativas foram feitas para inserir a computação e o Pensamento Computacional na Educação. Contudo, apesar dos avanços teóricos, não houve impacto significativo em termos metodológicos e epistemológicos. Até meados dos anos 2000, a principal perspectiva da Computação inserida na Educação tinha caráter mercadológico, visando a formação de mão de obra com foco em programação, sem uma preocupação expressiva com os aspectos educacionais.

Na terceira seção, foram apresentadas e problematizadas algumas perspectivas de Pensamento Computacional a partir de Wing (2006), com ênfase nos impactos educacionais dessas abordagens. Esse contexto permitiu evidenciar as transformações que influenciaram o debate sobre Pensamento Computacional na Educação. No entanto, embora o Pensamento Computacional seja frequentemente descrito como um modo de pensar, nenhuma das perspectivas discutidas abordou o conceito em termos de cognição.

Conforme debatido nas seções anteriores, o Pensamento Computacional é geralmente definido como uma habilidade baseada em ações objetivas, inferindo-se que um indivíduo possui Pensamento Computacional *a partir do que ele faz*. Por exemplo, ao realizar uma sequência específica de ações, infere-se que o aluno decompôs o problema ou depurou sua solução, sugerindo que ele possui Pensamento Computacional. Essa perspectiva, entretanto, não problematiza a natureza do pensamento subjacente a essas ações, restringindo a compreensão do Pensamento Computacional a um conjunto de procedimentos observáveis.

Ribeiro, Foss e Cavalheiro (2020) argumentam que, para compreender o Pensamento Computacional, é necessário entender o que é computação, sugerindo que a área visa "raciocinar sobre o raciocínio". Contudo, este trabalho propõe uma abordagem alternativa, defendendo que, para compreender o Pensamento Computacional, é essencial abordar o que se entende por *pensamento*. Essa necessidade torna-se ainda mais evidente quando se

analisa a incorporação do Pensamento Computacional na BNCC.

Conforme discutido na última seção deste capítulo, a BNCC reconhece o Pensamento Computacional como um elemento curricular, mas não o define de forma clara nem apresenta diretrizes pedagógicas consistentes para sua implementação. Essa indefinição não apenas reflete a falta de consenso na literatura, mas também evidencia a fragilidade conceitual do documento ao tratar do tema, reforçando a necessidade de investigações que busquem fundamentar teoricamente o Pensamento Computacional na Educação.

Dessarte, a literatura consultada não oferece uma construção teórica que permita definir, avaliar ou mensurar o Pensamento Computacional em termos cognitivos. Diante disso, este trabalho propõe uma perspectiva de Pensamento Computacional fundamentada no Modelo dos Campos Semânticos (MCS), no qual a cognição é vista como um processo de produção de *significados* (Ferreira, 2020). O MCS possibilita "leituras suficientemente finas de processos de produção de significados" (Lins, 2012, p. 18), permitindo caracterizar as ações cognitivas envolvidas em processos de Pensamento Computacional a partir das enunciações do sujeito em termos de produção de *significado*, distanciando-se de uma abordagem centrada em ações objetivas.

No próximo capítulo, serão apresentados os pressupostos epistemológicos que fundamentam a perspectiva teórica do MCS.

3 O MODELO DOS CAMPOS SEMÂNTICOS

O Modelo dos Campos Semânticos (MCS) é uma perspectiva teórico-epistemológica que busca compreender aspectos dos processos de produção de *conhecimento* e de *significado*, essencialmente no âmbito da Educação Matemática. Em vez de definir o que o *conhecimento deve ser*, o MCS se concentra em analisar como o *conhecimento está sendo produzido* em um contexto específico (Lins, 1999). Nesse sentido, o termo "modelo" em sua denominação não se refere a uma estrutura fixa ou prescritiva, mas a um processo contínuo de construção e reconstrução de *significados* por um sujeito epistêmico situado em um contexto sociocultural específico (Viola dos Santos; Paulo; Julio, 2023).

Propondo uma discussão inserida nesse contexto, este capítulo tem como objetivo elucidar a perspectiva epistemológica do MCS, destacando suas noções centrais[1] e como elas interagem para oferecer uma leitura suficientemente fina de processos de produção de *significados*. As noções não serão imediatamente explicadas à medida que surgem, mas serão desenvolvidas ao longo do texto. Essa escolha metodológica reflete o caráter interligado das noções do MCS, de modo que a compreensão de uma inevitavelmente conduz à consideração de outras[2].

Um dos principais movimentos que fundamenta este trabalho é a *leitura plausível*. No contexto do MCS, a leitura é compreendida como um processo de produção de *significados* a partir do que o *outro* enuncia, faz ou mostra (Viola dos Santos; Paulo; Julio, 2023). Com a *leitura plausível*, propõe-se um olhar que evita comparações com normas pré-estabelecidas ou julgamentos pela falta, focando em estabelecer a coerência interna a partir das enunciações do *outro*, sem imposição de juízos de valor (Paulo, 2020).

Assim, este capítulo organiza-se do seguinte modo: inicialmente, discutem-se as noções centrais do MCS; em seguida, elucida-se o movimento da *leitura plausível*, que orienta a postura epistemológica adotada na construção desta pesquisa; e, por fim,

[1] Neste trabalho, termos e expressões próprias do MCS são destacadas em itálico.

[2] Essa abordagem é inspirada no trabalho de Viola dos Santos, Paulo e Julio (2023), no qual os autores adotam a mesma estratégia de apresentar as noções ao longo do texto, sem a necessidade de defini-las imediatamente quando surgem pela primeira vez.

apresentam-se as considerações finais, que sintetizam os pontos centrais discutidos ao longo do capítulo.

3.1 Sobre as principais noções

O MCS coloca a relação entre o sujeito e seu meio social e cultural como um ponto central de investigação, destacando a influência da cultura na produção de *significados* e na constituição do sujeito epistêmico, assumindo que o sujeito não é um agente isolado, mas um produto de seu ambiente cultural e social. Suas produções de *significado*, portanto, só podem ser compreendidas à luz das legitimidades e práticas que circulam em seu contexto sociocultural (Lins, 1994). A cultura, entendida como um conjunto de práticas sociais e modos de produção de *significado*, molda a cognição do sujeito. Ao internalizar as *justificações* e formas de pensar validadas em sua comunidade, o sujeito se torna parte de uma rede de práticas e *significados* legitimados pela cultura em que está inserido.

Uma das principais contribuições do MCS é a análise do processo de internalização de legitimidades. Ao se inserir em um determinado contexto cultural, o sujeito internaliza as *justificações* e os modos de produção de *significado* que orientam o que ele pode ou não dizer, fazer ou acreditar. As *justificações* para as *crenças-afirmações* de um indivíduo, nesse sentido, não surgem de maneira autônoma, mas são moldadas pelas práticas culturais às quais ele está exposto e que ele internaliza ao longo de sua trajetória. O MCS, portanto, permite investigar como os *significados* são produzidos dentro de contextos específicos e como esses contextos influenciam as formas de cognição e enunciação do sujeito.

Fundamentado nesses pressupostos, Romulo Campos Lins (1955-2017) foi o principal responsável por desenvolver e aprimorar o MCS, construindo ao longo de sua trajetória acadêmica uma perspectiva epistemológica voltada à compreensão dos processos de produção de *significado*. Ao longo deste capítulo, serão abordadas as noções centrais que sustentam o MCS, fornecendo uma base para que o *um leitor*, instituído pelos *os autores* deste trabalho, compreenda as principais ideias e problematizações dessa abordagem teórica.

Desse modo, a primeira seção deste capítulo tem como objetivo apresentar e discutir

as principais noções do MCS, iniciando pela noção de *conhecimento*, sendo apresentada na sequência.

3.1.1 Conhecimento desde a perspectiva do MCS

No âmbito do MCS, as primeiras caracterizações de *conhecimento* foram introduzidas por Lins (1993), que utilizou a metáfora de um par ordenado para descrevê-lo: "[...] um par ordenado onde a primeira coordenada é *uma crença-afirmação*, e a segunda coordenada é uma *justificação* para esta *crença-afirmação*" (Lins, 1993, p. 86, grifos do autor). Assim, a *justificação* não é uma justificativa para a *crença*, mas um componente essencial do próprio *conhecimento*. É a *justificação*, construída e negociada dentro de um determinado contexto cultural e social, que confere legitimidade à *crença-afirmação*, tornando-a válida e aceitável naquele contexto.

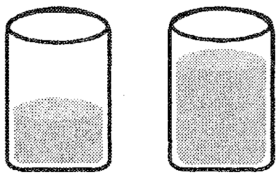
Em estudos subsequentes, como Lins (1994) e Lins e Gimenez (1997), essa estrutura conceitual se mantém semelhante. Nesses trabalhos, essa noção continua sendo metaforicamente representada como um par ordenado:

Conhecimento = ("*crença-afirmação*", "*justificação*").

Lins e Gimenez (1997) elucidam que uma mesma *crença-afirmação* pode estar associada a diferentes *justificações*, resultando em diferentes *conhecimentos*. Isso ocorre porque, desde a perspectiva do MCS, o *conhecimento* é sempre contextual, sendo negociado e produzido nas interações sociais. Assim, a *justificação* está sempre vinculada ao contexto em que o *conhecimento* é produzido e mobilizado.

Fundamentando-se em Lins e Gimenez (1997), apresenta-se, na sequência, a Atividade dos Tanques para ilustrar as noções de *conhecimento*, *crença-afirmação* e *justificação* no MCS. Na referida atividade, os alunos devem avaliar o nível de água em dois tanques, conforme ilustra a Figura 3.1.

Figura 3.1 – Atividade dos Tanques



Estes dois tanques são iguais.

Para encher o tanque da esquerda são precisos mais 9 baldes. Para encher o da direita, são precisos mais 5 baldes.

O que você pode falar sobre essa situação?

Fonte: Lins e Gimenez (1997)

Uma possível *crença-afirmação* enunciada pelos alunos pode ser: "No tanque à direita há mais água do que no da esquerda". O professor, então, pode questioná-los:

> "O que é que garante que vocês sabem que podem dizer isso?". Podemos imaginar, de imediato, pelo menos duas *justificações*:
> J1 – "Podemos *ver*, no desenho".
> J2 – "Porque falta mais para encher o tanque da esquerda (9 baldes) do que para encher o da direita (apenas 5 baldes)" (Lins; Gimenez, 1997, p. 124, grifos do autor).

Do ponto de vista do MCS, a explicitação dessas *justificações* revela a existência de dois *conhecimentos* distintos. No primeiro caso, o *conhecimento* pode ser representado da seguinte forma:

C_1 = ("No tanque à direita há mais água do que no da esquerda", "[pois] é possível ver no desenho").

No segundo caso, o *conhecimento* pode ser expresso como:

C_2 = ("No tanque à direita há mais água do que no da esquerda", "porque falta mais para encher o tanque da esquerda (9 baldes) do que para encher o da direita (apenas 5 baldes)").

Em C_1, é *plausível*[3] inferir que a *justificação* baseada na figura sugere que o aluno está operando em um *campo semântico* em que a percepção visual tem papel central na produção de *significado*. A centralidade na percepção visual pode dificultar a compreensão de situações abstratas, como no *pensamento algébrico*[4]. Informações adicionais

[3] O termo *plausível* é utilizado para indicar a aplicação do movimento de *leitura plausível*, que será detalhado posteriormente neste capítulo.

[4] A expressão *pensamento algébrico* refere-se a um modo específico de produzir *significados* para a álgebra, fundamentada nas ideias de Lins (1994). Essa expressão será retomada mais adiante, ao tratar das noções de *núcleo* e *campo semântico* no âmbito do MCS.

que expandam o contexto ou outras relações matemáticas, podem revelar as nuances do *campo semântico* que o aluno opera. A adição de dois baldes no tanque da esquerda, por exemplo, poderia evidenciar dificuldades para estimar diretamente o impacto dessa mudança no nível do tanque[5].

Em contrapartida, em C_2, a *justificação* baseada em relações matemáticas sugere que o aluno opera em um *campo semântico* em que a linguagem e o contexto matemático são centrais para o processo de produção de *significados*. Aqui, é *plausível* inferir que o aluno constitui um *objeto* que possui as relações matemáticas subjacentes ao desenho. O diagrama, nesse caso, deixa de ser central, podendo ser usado como ferramenta complementar na investigação.

A análise dessa atividade ressalta a importância da explicitação das *justificações* para uma mesma *crença-afirmação* enunciada. A princípio, poderia parecer que ambos os alunos possuem o mesmo *conhecimento*, baseado na *crença-afirmação* de que "No tanque à direita há mais água que no da esquerda". Contudo, ao explicitar as *justificações*, é possível identificar *conhecimentos* distintos, o que permite ao professor ajustar suas intervenções pedagógicas.

Antes de prosseguir com a apresentação das noções do MCS, é necessário esclarecer dois pontos sobre a *justificação*. Primeiro, a *justificação* não deve ser confundida com a justificativa de uma *crença-afirmação*, embora ela possa, em certos contextos, assumir essa função (Ferreira, 2020). Para ilustrar, considere o seguinte exemplo de *conhecimento*:

C_3 = ("Paris é a capital da França", "[pois] li em um livro de geografia").

Neste caso, a *justificação* não está diretamente relacionada à veracidade da *crença-afirmação* em si, mas sim àquilo que autoriza o sujeito a enunciá-la (Lins; Gimenez, 1997; Ferreira, 2020). O sujeito acredita que pode enunciar "Paris é a capital da França" porque considera legítimo fazê-lo com base no fato de que leu essa informação em um livro de geografia. A legitimidade de sua fala reside, portanto, na confiança social atribuída aos

[5] "É preciso que o leitor seja cuidadoso e não tente introduzir novos objetos de forma 'disfarçada': seria possível fazer essa estimativa dividindo a diferença entre os níveis em quatro partes, mas isso requer operar com o fato de que faltam 9 baldes de um lado e 5 do outro, introduzindo novos objetos" (Lins; Gimenez, 1997, p. 125).

livros. A *justificação*, nesse sentido, não se limita a complementar a *crença-afirmação*, mas a legitima dentro de um contexto social específico.

Em segundo lugar, é importante salientar que a *justificação* nem sempre é explicitada. Muitas vezes, o sujeito enuncia partindo do pressuposto de que sua fala é legítima naquele contexto, sem a necessidade de explicitar sua *justificação* (Ferreira, 2020). No entanto, Lins (1993) sugere que, em contextos educacionais, é fundamental incentivar a explicitação e análise de diferentes *justificações*, a fim de promover a reflexão crítica dos alunos sobre suas próprias falas, as dos colegas e as do professor. Ao explicitar as *justificações*, o professor pode identificar os *campos semânticos* em que os alunos estão operando, e, assim, compreender os processos de produção de *significado* que estão sendo mobilizados.

Ao longo de suas publicações (Lins, 1999, 2004, 2012), Lins gradualmente abandonou analogias com a Matemática, percebendo que as discussões sobre o MCS, na forma em que estavam sendo conduzidas, não resultaram nos avanços esperados (Paulo, 2020). Posteriormente, Lins (2012) sistematizou as noções do MCS em um glossário, oferecendo maior precisão conceitual. Nesse trabalho, o autor refina a noção de *conhecimento*, destacando sua natureza epistêmica:

> [...] uma crença-afirmação (o sujeito enuncia algo em que acredita) junto com uma justificação (aquilo que o sujeito entende como lhe autorizando a dizer o que diz). Um conhecimento não é nem mais, nem menos, que isto. Existe em uma enunciação e deixa de existir quando ela termina (Lins, 2012, p. 12).

Essa perspectiva sugere que o *conhecimento* não é algo universal e estático; é sempre situado e condicionado por fatores sociais e culturais específicos, e a *justificação* desempenha um papel constitutivo do próprio *conhecimento*. Não se trata de um complemento à *crença-afirmação*, mas sim do elemento que confere legitimidade ao que o sujeito enuncia, revelando os princípios que sustentam suas *crenças-afirmações*. Com isso, uma das problematizações propostas pelo MCS é compreender não apenas o que o sujeito sabe, mas também porque ele acredita no que afirma.

Até o momento, as argumentações tiveram o objetivo de evidenciar os diferentes

modos de produção de *significados* para um mesmo *resíduo de enunciação*[6], no caso, o enunciado da Atividade dos Tanques. Por exemplo, mesmo que dois indivíduos enunciem a mesma *crença-afirmação*, como "No tanque à direita há mais água do que no da esquerda", o *conhecimento* produzido por cada um deles podem ser substancialmente diferentes, dependendo das *justificações* que cada um oferece para sustentar suas *crenças*.

Desse modo, ao invés de considerar o *conhecimento* como um conjunto de proposições verdadeiras e neutras, o MCS o entende como um processo dinâmico e situado, que se concretiza no ato de enunciar. O *conhecimento* não reside em livros ou em enunciados dados, mas emerge na interação entre sujeito, *crença-afirmação*, *justificação* e contexto.

A enunciação de uma *crença-afirmação*, ou seja, a expressão daquilo que o sujeito acredita ser verdadeiro, juntamente com a *justificação* que a sustente, é o que constitui o *conhecimento*. Essa *justificação*, por sua vez, é o que autoriza o sujeito a enunciar, conferindo legitimidade à sua fala dentro de um contexto social específico[7]. Portanto, compreender o *conhecimento* como pertencente à ordem da enunciação implica reconhecer que ele é inseparável do sujeito que o enuncia e do contexto em que essa enunciação ocorre. Afinal, o sujeito enuncia a partir de suas experiências e legitimidades internalizadas em suas interações sociais.

Na sequência, discorre-se sobre as noções de *significado* e *objeto* no contexto do MCS.

3.1.2 Significado e objeto desde a perspectiva do MCS

No âmbito do MCS, o *significado*, assim como o *conhecimento*, é intimamente ligado ao contexto em que é produzido (Ferreira, 2020). Não se trata de um conceito abstrato ou universal, mas de um processo dinâmico e situado. Conforme (Lins; Gimenez,

[6] No âmbito do MCS, um *resíduo de enunciação* é compreendido como "[...] algo com que me deparo e que acredito ter sido dito por alguém [...] Sons, rabiscos de todo tipo, arranjos de coisas, gestos, imagens, construções. Mas também a borra de café ou chá no fundo da xícara, o resultado do lançamento de moedas ou varetas, a disposição dos planetas no céu, o fato de este carro ter a placa de uma cidade da qual nunca ouvi falar, a tempestade que devastou a cada de uma pessoa poucos dias depois de ela ter abandonado a religião que professava, e assim por diante" (Lins, 2012, p. 27).

[7] Convém salientar, mais uma vez, que a *justificação* nem sempre se apresenta de forma explícita. Muitas vezes, o sujeito a considera implícita, partindo do pressuposto de que sua fala é legítima no contexto em que se encontra.

1997, p. 145, grifos do autor), é "o conjunto de coisas que se diz a respeito de um objeto. Não o conjunto do que se *poderia* dizer, e, sim, *o que efetivamente se diz* no interior de uma atividade". Nesse processo, a *justificação* novamente desempenha o papel central, ao autorizar a enunciação do sujeito e conferir legitimidade à sua fala, delimitando o que é legítimo enunciar em um *campo semântico* específico.

Por sua vez, *objeto*, no MCS, não se limita à ideia da "coisa em si", mas é constituído a partir da produção de *significados* no interior de uma atividade[8] (Ferreira, 2020). O *objeto* é aquilo sobre o qual o sujeito enuncia (Lins, 2012), produzindo *significado* a partir de suas experiências e legitimidades internalizadas em suas interações sociais. Assim, a produção de *significados* e a constituição de *objetos* são processos indissociáveis, que ocorrem simultaneamente por meio de *resíduos de enunciação*. Ao se deparar com um *resíduo*, o sujeito, a partir de sua *direção de interlocução*, produz *significado* e, ao fazê-lo, constitui o *objeto* sobre o qual enuncia.

É importante ressaltar que o *significado* não preexiste no *objeto*, mas emerge na interação entre sujeito, objeto[9] e contexto. Não se trata de "descobrir" o *significado* do objeto, mas de construí-lo ativamente no processo de enunciação (Ferreira, 2020). O *significado* é sempre contextualizado, dependente de quem o produz, para quem e em qual contexto.

Essas considerações sobre *objeto* e *significado*, para Ferreira (2020), têm implicações pedagógicas. Um exemplo que ilustra essas implicações pode ser revisitado na análise da Atividade dos Tanques.

Na análise mencionada, observou-se que, embora ambos os alunos tivessem enunciado a mesma *crença-afirmação* – "No tanque à direita há mais água do que no da esquerda" – a explicitação das *justificações* demonstrou a existência de *conhecimentos* distintos.

[8] A noção de atividade, no âmbito do MCS e no contexto deste trabalho, é compreendida a partir da Teoria da Atividade de Leontiev (2004). Segundo o autor, uma atividade é composta por três elementos estruturais: necessidade, objeto e motivo. A necessidade é o princípio da atividade, é o que "dirige e regula a atividade do sujeito" (Asbahr, 2005, p. 29). O objeto, por sua vez, "é o objetivo, o fim, o propósito" (Dantas; Ferreira; Paulo, 2016, p. 224). Quando um objeto corresponde a uma necessidade, é possível afirmar que a atividade possui um motivo (Leontiev, 2004).

[9] Aqui, o termo "objeto" é empregado de forma mais ampla, desvinculado de seu uso técnico no MCS, referindo-se a qualquer elemento ao qual o sujeito atribui *significados* em sua enunciação.

A introdução da noção de *objeto* na perspectiva do MCS permite uma leitura significativamente mais precisa do processo de produção de *significados*. Em C_1, o *conhecimento* está fundamentado na percepção visual do diagrama, uma vez que a *justificação* se baseia no desenho. Em C_2, há uma construção de relações matemáticas embasada no texto da atividade. É *plausível*, portanto, inferir que os *significados* produzidos pelos alunos em relação ao *resíduo de enunciação* "Atividade dos tanques" são distintos.

Em C_1, é *plausível* inferir que o *objeto* constituído é o próprio desenho, ao qual o aluno produz *significado* com base na percepção visual. Se fossem adicionados dois baldes de água ao tanque da esquerda, por exemplo, o aluno que constitui esse *objeto* poderia enfrentar dificuldades para estimar o impacto dessa mudança. Isso ocorre porque o *objeto* constituído (o desenho) não inclui as relações matemáticas subjacentes.

Em C_2, o aluno constitui um *objeto* que articula o desenho e o enunciado textual. É *plausível* inferir que ele estabelece uma relação entre as quantidades de água e o número de baldes necessários para encher cada tanque. Nesse caso, a adição de dois baldes ao tanque da esquerda seria compreendida com base na análise dessas relações.

A análise dos *conhecimentos* dos alunos em C_1 e C_2 revela diferentes *objetos* constituídos, com implicações significativas para os processos de produção de *significados* para um mesmo *resíduo de enunciação*.

Desse modo, um sujeito produz *significados*, constitui *objetos* e realiza enunciações em uma *direção* que acredita ser legítima. Essa *direção* é denominada, no MCS, como *interlocutor* (Lins, 1999) ou *direção de interlocução* (Paulo, 2020), de forma que essa direção estabelece "um horizonte do possível, do que pode ser dito" (Paulo, 2020, p. 18).

A seguir, discutem-se essas noções em maior profundidade.

3.1.3 Espaço comunicativo e direção de interlocução desde a perspectiva do MCS

Para introduzir as discussões sobre *espaço comunicativo* e *direção de interlocução* na perspectiva do MCS, considere a seguinte situação: duas pessoas, Carlos e Sofia, estão montando um móvel recém-comprado. Carlos, experiente em tarefas manuais, começa a

separar as peças, enquanto Sofia tenta decifrar as orientações de montagem ilustradas em um manual de instruções. Nesse contexto, ocorre o seguinte diálogo:

Carlos: *(apontando para uma peça) Pega aquelas buchas ali, Sofia, por favor. Vamos começar a fixar essa parte.*

Sofia: *(franzindo a testa) Bucha? Que bucha? Não tô vendo nenhuma bucha aqui nas instruções...*

Carlos: *(olhando para as instruções) Ué, tá vendo aqui neste passo 3? Aquela peça cilíndrica pequena.*

Sofia: *(confusa) Mas aqui não está escrito "bucha"! Está chamando de "pino de fixação tipo A". E esta imagem parece mais um parafuso do que uma bucha...*

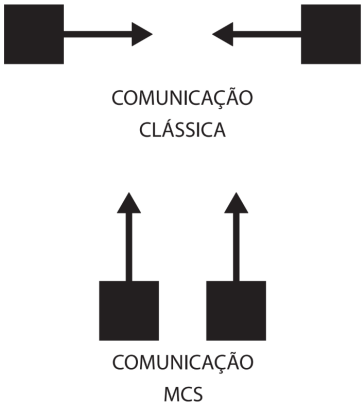
Carlos: *(rindo) Ah, Sofia, deixa de ser complicada! Você sabe que peça é essa. É a mesma coisa que uma bucha!*

A partir dessa situação, é possível inferir, corroborando as ideias de Dantas (2016), uma crença de que a palavra "bucha" possui uma função denotativa, referindo-se diretamente a um objeto da realidade. Segundo o autor, também prevalece a ideia de que, se algo é dito, deve ser compreendido pelo outro, a menos que ocorra algum problema de comunicação entre as partes, uma falha no processo emissor-mensagem-receptor, como sugere a situação apresentada.

Esse entendimento reflete uma visão de senso comum de uma realidade objetiva, uma linguagem capaz de descrever essa realidade, e a comunicação como um processo linear que permite a interação social, tornando compreensíveis as coisas para indivíduos de uma mesma cultura (Dantas, 2016).

No entanto, a perspectiva adotada neste trabalho diverge dessa visão. No MCS, a noção tradicional de comunicação é substituída pela de *espaço comunicativo*. Nessa perspectiva, a comunicação não se resume a "duas pessoas falando uma para outra", mas envolve dois sujeitos cognitivos enunciando em *direções de interlocução* instituídas por ambos (Lins, 2012). A Figura 3.2 ilustra a noção de *espaço comunicativo* no âmbito do MCS.

Figura 3.2 – *Espaço comunicativo* no âmbito do MCS



Fonte: Lins (2012)

Nesse contexto, quando duas pessoas conversam, estão assumindo os papéis de *autor* e *leitor*, conforme proposto por Lins (2012). Durante o processo comunicativo, uma das pessoas, ao falar, atua como *o autor*. Ao fazê-lo, imagina que há alguém que diria o mesmo que ela está dizendo com a mesma *justificação* que a autoriza a enunciar o que acredita. Em outras palavras, a fala do *autor* é dirigida a *um leitor* – não ao ser biológico presente, mas um sujeito cognitivo[10] que, supõe-se, produziria os mesmos *significados* a partir das enunciações realizadas. O outro participante, disposto a se comunicar, assume o papel de *o leitor*, reconhecendo o falante como *um autor* e, nessa *direção*, produz seus próprios *significados*. A Figura 3.3 representa essas noções de *autor* e *leitor* no MCS.

Figura 3.3 – Noções de *autor* e *leitor* no âmbito do MCS

Fonte: Lins (2012).

As *direções de interlocução* são, portanto, instituídas durante o processo comunicativo pelos participantes, que alternam nos papéis de *autor* e *leitor*. Desse modo, um sujeito sempre realiza enunciações, produz *significados* e constitui *objetos* em uma *direção* que acredita ser legítima, denominada *interlocutor* (Lins, 1999) ou *direção de interlocutor* (Paulo, 2020).

[10] No MCS, distingue-se entre sujeito biológico e sujeito cognitivo. O sujeito biológico refere-se ao corpo físico, enquanto o sujeito cognitivo é aquele que produz/enuncia *conhecimento* e produz *significados* (Paulo, 2020).

É de suma importância compreender que, na perspectiva do MCS, uma *direção de interlocução* não se refere a um ser biológico, mas a um ser cognitivo. Nesse sentido, Paulo (2020) aduz que a terminologia *interlocutor* pode ser substituída por *direção de interlocução* sem alteração nas concepções teóricas envolvidas. Segundo o autor, essa expressão desassocia qualquer corporeidade que o termo *interlocutor* possa, porventura, sugerir.

Portanto, ao utilizar a terminologia *direção de interlocução*, não é necessário referir-se especificamente ao *o autor* e o *um autor*, *o leitor* e o *um leitor*, uma vez que essa noção se refere ao sujeito cognitivo envolvido, seja ele o responsável por produzir um *resíduo de enunciação* ou o sujeito que produz *significados* a partir deste *resíduo*. Nesse sentido, não importa se a enunciação ocorre na *direção de interlocução* de quem produziu o *resíduo de enunciação* ou na *direção* para a qual se está produzindo *significados*.

Para ilustrar a noção de *direção de interlocução*, considere a seguinte situação[11]: Tiago e Helena conversam após assistirem à previsão do tempo, na qual a repórter mencionou a possibilidade de chuva no dia seguinte. A partir desse contexto, ocorre o seguinte diálogo:

Tiago: *Legal! Amanhã o tempo estará bom!*

Helena: *Como bom, Tiago? Não ouviu a repórter dizer que estará chovendo? Quando chove fica tudo mais difícil.*

Tiago: *Sim, mas você já parou para pensar há quanto tempo não chove em nossa cidade?*

Helena: *Ah! Verdade! Pelo menos assim teremos um alívio nessa seca, e talvez não fiquemos sem água. Você tem razão, o tempo vai estar bom mesmo.*

Tiago: *Exatamente! Era nisso que eu estava pensando.*

Esse diálogo exemplifica como Tiago e Helena produzem *significados* a partir de um mesmo *resíduo de enunciação*, no caso, a previsão do tempo transmitida pela televisão. Tiago considera que o tempo chuvoso será bom, uma conclusão que ele julga legítima, considerando o contexto da seca na cidade. Ao enunciar, Tiago o faz em uma *direção de*

[11] Exemplo extraído e adaptado de Dantas (2016).

interlocução que compreenderia e aceitaria essa afirmação.

No entanto, o *significado* que Helena produz a partir do *resíduo de enunciação* de Tiago causa-lhe *estranhamento*: "Como bom, Tiago? Não ouviu a repórter dizer que estará chovendo?". Esse *estranhamento* indica que Helena e Tiago estão enunciando em *direções* diferentes. O *estranhamento* de Helena revela que a afirmação de Tiago não seria legítima se ele estivesse operando com as mesmas legitimidades que ela.

Helena prossegue, explicitando sua *justificação* e *direção de interlocução*: "Quando chove fica tudo mais difícil". Para ela, a chuva complica suas atividades, tornando legítima a afirmação de que o tempo não estaria bom. Ao se deparar com essa *justificação*, Tiago sente a necessidade de apresentar a sua: "Sim, mas você já parou para pensar há quanto tempo que não chove em nossa cidade?". Ele considera legítima sua pergunta, baseada na observação da seca na cidade.

É *plausível* inferir que, ao continuar o diálogo, Helena passou a operar com base no *significado* que produziu a partir da *justificação* apresentada por Tiago. Nota-se uma mudança na *direção de interlocução* de Helena. Inicialmente, ela estranha o *significado* produzido por Tiago, mas, ao compreender a legitimidade de sua fala, muda sua *direção de interlocução* e passa a enunciar na mesma *direção*: "Ah! Verdade! Pelo menos assim teremos um alívio nessa seca, e talvez não fiquemos sem água".

É importante destacar que Helena também legitimou a *justificação* de Tiago. Caso contrário, eles não teriam compartilhado uma *direção de interlocução*. Segundo as noções do MCS, não é suficiente reconhecer que se está falando em uma *direção* diferente da do outro (Dantas, 2016). É necessário aceitar a outra *direção* como legítima para que se possa enunciar a partir dela.

Se Helena não tivesse alterado sua *direção de interlocução*, ambos continuariam produzindo *significados* distintos. Nesse caso, Tiago e Helena apresentariam suas *justificações*, mas, por estarem cientes de que enunciavam em *direções* diferentes, não aceitariam as falas um do outro e, consequentemente, não compartilhariam *direções de interlocução*. Sobre essa possibilidade, (Dantas; Lins, 2017, p. 7) evidenciam que "[...] duas pessoas podem ou não falar em uma mesma direção. Quando não falam em uma mesma direção,

não compartilham interlocutores. Elas não deixam de estar interagindo, mas podemos questionar sobre a possibilidade de estarem se comunicando".

Por fim, cabe salientar a última fala de Tiago no diálogo. O *descentramento*[12] por parte de Helena permitiu que ambos convergissem em uma mesma *direção de interlocução*. Assim, na perspectiva do MCS, estavam em um processo de *interação produtiva*, no qual os sujeitos envolvidos compartilham *direções de interlocução*, e, portanto, o que um fala não parece paradoxal ao outro (Dantas, 2016).

A importância da *direção de interlocução* no MCS reside no fato de que ela carrega em si as marcas das legitimidades, ou seja, as regras e normas internalizadas pelo sujeito que determinam o que é aceitável, compreensível e válido dentro de um determinado contexto social ou cultural. Corroborando as ideias de Ferreira (2020), é a internalização de *direções de interlocução* e de legitimidades, que torna possível a produção de *conhecimento*:

> [...] enunciamos, pois acreditamos que nossa enunciação será aceita como legítima no interior da prática cultural para a qual enunciamos – isto é antecipar a legitimidade da enunciação. Assim, o individual nunca antecede a legitimidade dos modos de produção de significado que internalizaram o sujeito do conhecimento (Ferreira, 2020, p. 122).

É pertinente destacar que um indivíduo pode ser o *interlocutor* de si mesmo. Ao internalizar certos repertórios culturais, ele passa a produzir *significados* em determinadas *direções de interlocução*. Assim, constrói-se certos repertórios de modos de dizer, considerados legítimos dentro de um contexto social ou cultural específico.

A *direção de interlocução*, portanto, refere-se ao espaço onde o sujeito enuncia (*crenças-afirmações*) e onde acredita que os outros enunciariam o mesmo que ele com a mesma *justificação* que o autoriza a fazê-lo (Lins, 2012). Dessa forma, o sujeito, na perspectiva do MCS, nunca precede o aspecto social, pois sempre antecipa legitimidades. Aprender, nesse contexto, é a capacidade que o sujeito tem de antecipar essas legitimidades, dispensando a necessidade de uma validação externa.

Neste momento, convém retomar a Atividade dos Tanques para evidenciar as *direções de interlocução* em que C_1 e C_2 enunciam.

[12] A noção de *descentramento* será abordada mais adiante, na seção que fundamenta o movimento da *leitura plausível*.

No caso de C_1, é *plausível* inferir que este aluno enuncia em uma *direção de interlocução* orientada para a validação visual, acreditando que o ato de ver é suficiente para justificar sua *crença-afirmação*. Para C_1, é legítimo enunciar com base na percepção imediata e visual, sem a necessidade de articulação com o enunciado textual da atividade. Por outro lado, é *plausível* inferir que C_2 enuncia em uma *direção de interlocução* que produz *significado* para as relações matemáticas presentes no enunciado da atividade.

Em síntese, a análise do *conhecimento* dos alunos em C_1 e C_2 revela *direções de interlocução* e *objetos* constituídos distintos. Contudo, é importante ressaltar que as discussões em torno da Atividade dos Tanques não pretendem afirmar que um *conhecimento* é correto e o outro não, mas sim de destacar que se trata de *conhecimentos* distintos. De acordo com Lins (1999), o *conhecimento* é relativo ao contexto em que é produzido e enunciado, sendo moldado pelas práticas sociais, culturais e históricas. Sob essa perspectiva, a validade de um *conhecimento* não pode ser determinada por critérios absolutos de "melhor" ou "pior", mas por sua legitimidade dentro do contexto social e cultural em que se manifesta.

Ao caracterizar a *justificação* como parte constitutiva do *conhecimento*, Ferreira (2020) argumenta que o MCS rompe com correntes epistemológicas que consideram a verdade como elemento central do *conhecimento*. Inicialmente, pode parecer que, no âmbito do MCS, todo *conhecimento* é verdadeiro e toda enunciação de uma *crença-afirmação* constitui um *conhecimento*.

> Essa afirmação é interessante por pelo menos dois motivos. O primeiro deles é que um conhecimento, especialmente o científico, precisa que alguma instituição/comunidade assegure que ele é verdadeiro. Uma instituição/comunidade que fiscalize os terrenos do que é válido e do que não é válido. Isso é um conhecimento, é válido. Aquilo não é conhecimento, não é válido. Mesmo que uma retórica seja quase sem fim, nesse processo, é preciso ter uma origem, um lugar para afirmar algo. O segundo motivo é que um conhecimento, especialmente, o científico, faz a manutenção do que pode ser dito e, como efeito, exclui produções que não atendam a interesses éticos, filosóficos, econômicos, sociais, políticos, etc. (Viola dos Santos; Paulo; Julio, 2023, p. 5-6).

Contudo, mesmo com a problematização proposta por Viola dos Santos, Paulo e Julio (2023), cabe salientar que, no contexto do MCS, não há um relativismo absoluto ou uma verdade objetiva. Segundo essa perspectiva, a questão da verdade é deslocada para

uma questão de legitimidades (Ferreira, 2020).

A perspectiva do MCS ainda considera o *conhecimento* uma noção não-trivial por dois motivos. Primeiro, porque nem tudo pode ser dito, uma vez que qualquer cultura aceita alguns, mas nunca todos os modos possíveis de produzir *significados* (Lins; Gimenez, 1997). A constituição de uma *direção de interlocução* define, para o sujeito do conhecimento, o que pode ou não ser enunciado, dependendo da legitimidade do que é dito. Assim, o sujeito do conhecimento não faz uma enunciação qualquer em um processo de produção de *significados*, como seria em um relativismo absoluto (Ferreira, 2020). Por exemplo:

> Se alguém, começando amanhã, jogasse uma moeda para o alto e dissesse 'vai chover amanhã', caso o resultado fosse cara, e 'não vai chover amanhã', caso o resultado fosse coroa, e seguisse acertando por mil dias, ainda assim a comunidade dos meteorologistas não aceitaria a previsão do dia 1001 como conhecimento. A enunciação,
>
> "Vai chover amanhã; (pois) eu joguei a moeda e deu cara."
>
> não seria legítima para esse interlocutor, a comunidade dos meteorologistas e, se a pessoa quisesse participar dela, não poderia enunciar "Vai chover amanhã", embora talvez houvesse um outro interlocutor que a legitimasse (Lins; Gimenez, 1997, p. 143).

Em segundo lugar, o próprio processo de produção de *significados* estabelece limites internos: não é possível, por exemplo, produzir *significado* para $x = -3$ no *campo semântico* da álgebra das balanças[13]. Essa impossibilidade é chamada de *limite epistemológico* (Lins; Gimenez, 1997).

Dessa forma, os dois aspectos abordados, a natureza social e cultural do *conhecimento* e os mecanismos de internalização de legitimidades a partir de práticas socioculturais, juntamente com a existência de *limites epistemológicos*, asseguram que a noção de *conhecimento* na perspectiva do MCS não se configure como um "vale-tudo" (Lins; Gimenez, 1997).

Na sequência, apresentam-se as noções de *núcleo* e *campo semântico* na perspectiva do MCS.

[13] A noção de *limite epistemológico*, *campo semântico* e álgebra das balanças será abordada com maior profundidade na seção seguinte.

3.1.4 Núcleo e campo semântico desde a perspectiva do MCS

Segundo o MCS, o conceito de *núcleo* refere-se a estipulações locais que funcionam como verdades absolutas dentro de uma determinada atividade, ou seja, *crenças-afirmações* que não requerem *justificações* no interior desse contexto específico (Lins, 2012). Essas verdades absolutas, aceitas temporariamente como "regras do jogo", constituem o *núcleo* e dispensam explicações adicionais no decorrer da atividade.

Um exemplo dessa noção pode ser observado na Atividade dos Tanques. O próprio desenho do tanque, incluindo sua forma geométrica, capacidade e os níveis de água representados, é compreendido como o *núcleo* da referida atividade. Essas características são tratadas como verdades temporárias e não exigem uma *justificação*, pois os participantes da atividade aceitam-nas como condições pré-estabelecidas. É essa aceitação que sustenta a continuidade da interação no interior da atividade.

Entretanto, o *núcleo* não é algo fixo ou imutável; ele é constituído e adaptado ao longo da atividade, à medida que os participantes produzem *significados* e interagem uns com os outros. Diferente de um "conhecimento universal" ou um conteúdo específico, como teoremas ou axiomas matemáticos, o *núcleo* depende das condições da atividade e do contexto em que está inserido (Lins; Gimenez, 1997).

Enquanto o *núcleo* refere-se às verdades aceitas momentaneamente, o *campo semântico*, por outro lado, diz respeito ao processo de produção de *significados* em relação a esse *núcleo*. Conforme os participantes de uma atividade interagem, eles produzem *significados*, constituindo um *campo semântico* que orienta suas ações e enunciações. Esse processo de produção de *significados* ocorre no interior da atividade e é constantemente transformado à medida que novos *significados* são produzidos (Lins, 2012).

Em outras palavras, o *campo semântico* não é um conjunto fixo de conceitos ou termos, mas um processo de produção de *significados* que emerge e se adapta ao contexto. Diferentes *campos semânticos* podem ser construídos para um mesmo objeto, dependendo do contexto e da atividade envolvida. Aqui, cabe salientar que a expressão "objeto" não se refere à noção do MCS. Não poderia ser diferente, pois, se os *campos semânticos* concebidos

são distintos, os *objetos* (noção do MCS) também são.

Por exemplo, no caso da Atividade dos Tanques, o diagrama do tanque pode disparar uma demanda por produção de *significados* distintos, dependendo da *direção de interlocução* e das *justificações* que cada participante enuncia ou utiliza. Em um *campo semântico* visual, um aluno pode focar nos aspectos perceptuais do diagrama, enquanto outro, operando em um *campo semântico* da Matemática, pode concentrar-se nas relações quantitativas e geométricas representadas pelo desenho.

Portanto, o *núcleo* e o *campo semântico*, embora distintos, estão interligados: o *núcleo* define as verdades momentâneas aceitas dentro de uma atividade, enquanto o *campo semântico* refere-se ao processo de produção de *significados* referentes a esse *núcleo*. Juntas, essas noções permitem compreender como os *significados* são produzidos e como os sujeitos interagem em atividades de aprendizagem, evidenciando que os *significados* e *objetos* constituídos dependem do contexto e das interações sociais.

Para elucidar as noções de *núcleo* e *campo semântico*, propõe-se uma análise da ideia de igualdade em diferentes *campos semânticos*, particularmente na álgebra algébrica e na álgebra da balança[14].

O *campo semântico* da álgebra algébrica é compreendido, de acordo com Lins (1994), como um modo específico de produzir *significados* para a álgebra, caracterizado pelo pensamento algébrico. Esse tipo de pensamento distingue-se por três características principais: aritmeticismo, internalismo e analiticidade.

Considere a expressão $2x + 6 = 0$ como um exemplo. O aritmeticismo refere-se à produção de *significados* para essa expressão como uma equação, em que operações como adição, subtração, multiplicação e divisão são aplicadas aos números (2, 6 e 0) e à incógnita x, que é tratada como um número desconhecido. O internalismo, por sua vez, refere-se à forma como a equação é manipulada internamente, ou seja, sem a necessidade de recorrer a modelos externos, baseando-se nas propriedades da igualdade e das operações. Por exemplo, ao subtrair 6 de ambos os lados da equação, preserva-se a igualdade sem

[14] Apresentam-se esses exemplos de *campos semânticos* pois são recorrentes nas obras de Lins (1993, 1999, 2012) e auxiliam na explanação das noções.

modificar o contexto em que a equação está inserida. Por fim, a analiticidade refere-se ao tratamento da incógnita x como se fosse um número conhecido, ainda que seu valor específico seja inicialmente desconhecido. Assim, ao dividir ambos os lados da equação por 2, chega-se à solução $x = -3$.

Nesse *campo semântico*, a equação $2x + 6 = 0$ é solucionada de forma abstrata, com a incógnita x assumindo o valor específico de -3. Esse *campo semântico* caracteriza-se por um pensamento formal e abstrato, em que as operações e as propriedades matemáticas são centralizadas na produção de *significados*, sem a necessidade de referenciais externos ou concretos.

Em contraste, o *campo semântico* da álgebra da balança, baseia-se na representação simbólica de uma balança de dois pratos, em que a igualdade é vista como o equilíbrio entre os lados da balança. Aqui, as equações são compreendidas como representações desse equilíbrio, e as operações algébricas são visualizadas como ações que afetam o equilíbrio da balança. Adicionar ou remover pesos de ambos os lados da balança corresponde a realizar operações que mantêm a igualdade da equação.

Nesse *campo semântico*, as incógnitas são compreendidas como pesos desconhecidos, e o objetivo das operações algébricas é descobrir o valor desses pesos para manter o equilíbrio da balança. No entanto, ao contrário do *campo semântico* da álgebra algébrica, a manipulação de equações do *campo semântico* da balança apresenta restrições físicas. Por exemplo, a solução da equação $2x + 6 = 0$ no *campo semântico* da álgebra algébrica, que resulta em $x = -3$, apresenta um *limite epistemológico* do *campo semântico* da balança, uma vez que a ideia de "peso negativo" não possui uma correspondência física imediata. Além disso, operações como a divisão por uma incógnita podem não encontrar uma representação clara no contexto da balança.

Essas diferenças ressaltam os *limites epistemológicos* de cada *campo semântico*. Enquanto o *campo semântico* da álgebra algébrica oferece uma abordagem abstrata e formal para resolver equações, o *campo semântico* da álgebra da balança é apresenta algumas restrições físicas pela impossibilidade de lidar com conceitos como números negativos.

As noções elucidadas até este momento permitem apresentar e problematizar a

postura de construção desta pesquisa, fundamentada essencialmente no movimento da *leitura plausível*, a qual é apresentada a seguir.

3.2 A leitura plausível

Sob a perspectiva do MCS, a leitura é compreendida como um processo de produção de *significados* a partir de *resíduos de enunciação*, ou seja, do que o *outro*[15] diz, faz ou demonstra (Viola dos Santos; Paulo; Julio, 2023). Esses *significados*, no entanto, não estão contidos nos *resíduos* em si, mas são produzidos pelo *o leitor*, com base em seus repertórios[16], suas legitimidades e no contexto em que a leitura ocorre.

No cotidiano, muitas vezes, a leitura do *outro* tende a ser orientada por uma perspectiva de falta: "Falta conhecimento, falta empenho, falta maturidade. [...] Falta de tudo para muitos" (Viola dos Santos; Paulo; Julio, 2023, p. 14). Em contrapartida, a *leitura plausível*, fundamentada nos pressupostos do MCS, propõe uma abordagem que visa "compreender o que está sendo dito sem fazer comparações, sem dizer que falta alguma coisa no texto que lemos para que ele tenha sentido, ou que quem o escreveu não havia compreendido bem a ideia sobre a qual escrevia" (Paulo, 2020, p. 10).

Esta seção tem como objetivo caracterizar o movimento da *leitura plausível* no âmbito do MCS. Para tanto, são abordadas as noções de *conhecimento* em primeira e terceira pessoa, a noção de *descentramento*, e os movimentos que orientam a *leitura plausível*. Além disso, discute-se sua aplicação como metodologia de pesquisa no campo da Educação Matemática.

3.2.1 Conhecimento em primeira e terceira pessoa desde a perspectiva do MCS

A *leitura plausível*, conforme proposta pelo MCS, não tem como objetivo atingir uma verdade absoluta ou objetiva. Em vez disso, busca compreender o *significado* que o *outro* atribui a algo. Esse *outro*, contudo, não é visto como um ente passivo, mas como

[15] Neste trabalho, o termo *outro* é empregado como um sinônimo da terminologia *um autor*.

[16] No âmbito do MCS, como o *conhecimento* é compreendido como da ordem da enunciação, a expressão "*conhecimentos* prévios" perde sentido. Portanto, adota-se a terminologia "repertórios" para se referir aos modos de produção de *significados* já internalizados por um sujeito.

um sujeito cognitivo que produz *significados* a partir de suas *justificações* e legitimidades. Nesse contexto, Lins (2002, *apud* Paulo, 2020) destaca a importância de distinguir entre o "*conhecimento* em primeira pessoa" e o "*conhecimento* em terceira pessoa", esclarecendo que:

> [...] é necessário distinguir operacionalmente "conhecimento primeira pessoa" (a crença-afirmação se refere ao que eu acredito) e "conhecimento terceira pessoa" (a crença-afirmação se refere ao que eu acredito que a pessoa acredita). Assim, fica claro que quando eu falo do outro estou na verdade falando de mim [...] (Lins, 2002, p. 61, *apud* Paulo, 2020, p. 13, grifos do autor).

Essa distinção é central para a análise de textos[17] no MCS, uma vez que, ao falar sobre o *outro*, o observador está, na verdade, produzindo *significados* a partir de suas próprias legitimidades, ou seja, está produzindo um *conhecimento* em terceira pessoa.

Um exemplo dessa dinâmica ocorre ao inferir que um aluno compreende o conceito de variáveis no Scratch ao utilizar o bloco `[mude (Pontuação) para 0]`. Na perspectiva do MCS, isso configura um *conhecimento* em terceira pessoa, pois, até que o aluno "explicite suas *justificações*, o que ele acredita autorizá-lo a fazer o que faz, o que existe é apenas a *crença* do observador de que ele faz aquilo por algum motivo; é o observador que está produzindo *justificações* que tornam coerente aquele *resíduo de enunciação*" (Paulo, 2020, p. 13, grifos do autor).

Desse modo, segundo Paulo (2020), o observador estabelece uma *direção de interlocução* em que considera legítima a produção de tais *justificações*, acreditando que elas sejam coerentes dentro da própria *direção de interlocução* instituída. No exemplo em questão, qualquer inferência sobre a compreensão do conceito de variáveis só se torna *plausível* quando o aluno explicita suas *justificações* para o uso do bloco `[mude (Pontuação) para 0]`. Essa *justificação* é construída dentro de um horizonte de enunciações possíveis, determinado pela *direção de interlocução* instituída pelo aluno – seja em uma comunidade de programadores, no contexto de uma aula de programação etc.

[17] Paulo (2020) realiza uma distinção entre texto e *resíduo de enunciação*, nos termos do MCS. Segundo o autor, "tudo que é posto no mundo como demanda de produção de *significado* pode vir a ser *resíduo de enunciação*. Vir a ser porque depende de que alguém produza *significado* a partir dele. Neste momento, a produção de *significado*, essas noções estão imbricadas: se alguém se depara com algo que ele acredita ter sido enunciado por outrem, esse algo torna-se *resíduo de enunciação* e, no momento em que este alguém produz *significado* a partir desse *resíduo*, ele se constitui em texto" (Paulo, 2020, p. 17).

Cabe à *leitura plausível* identificar essas *justificações* e reconhecer suas legitimidades, sem julgar sua "correção" ou "verdade", algo que pode ser discutido posteriormente, mas não faz parte do que é fundamental nesse movimento[18]. O objetivo é compreender as *justificações* dentro dos termos e do contexto em que foram produzidas, sem impor critérios externos de avaliação.

A *leitura plausível*, portanto, organiza-se em torno da constituição de *um autor* e da produção de *significados* a partir dos *resíduos de enunciação* desse *um autor*. Esse processo exige *do leitor* um esforço sistemático para reconstruir a lógica interna do discurso do *outro*, buscando compreendê-lo em seus próprios termos e identificando suas legitimidades, *justificações* e os *objetos* que sustentam seu modo de pensar.

No entanto, a *leitura plausível* vai além de uma descrição dos pensamentos do *um autor* constituído. Ao reconstruir o processo de produção de *significados* do *um autor*, *o leitor* inevitavelmente introduz suas próprias legitimidades e modos de produção de *significado*. Essa interação entre o *conhecimento* em terceira pessoa (construído sobre o *um autor*) e o *conhecimento* em primeira pessoa (produzido pelo *o leitor*) torna a *leitura plausível* um processo flexível e situado em um contexto específico.

É importante ressaltar que, mesmo após a explicitação da *justificação* para uma determinada *crença*, qualquer inferência decorrente do movimento da *leitura plausível* se caracteriza como *conhecimento* em terceira pessoa. Isso ocorre porque, no MCS, o processo de produção de *significado* a partir das *justificações* do *outro* é sempre mediado pelos próprios repertórios e legitimidades do observador. Ou seja, ao formular uma inferência, o observador não está acessando diretamente a *crença* do outro, mas sim reconstruindo-a a partir de sua própria perspectiva. O que se obtém é um *conhecimento* que, embora busque compreender o *outro*, permanece enraizado nas práticas cognitivas e legitimidades do próprio observador, mantendo-se, portanto, no domínio do *conhecimento* em terceira pessoa.

A seguir, discorre-se sobre a noção de *descentramento* na perspectiva do MCS.

[18] Essa questão será abordada com mais detalhes posteriormente.

3.2.2 Descentramento desde a perspectiva do MCS

O *descentramento* refere-se à capacidade de *o leitor* deslocar-se de sua própria perspectiva, suspendendo temporiamente suas *crenças* e convicções, a fim de compreender o outro a partir de seus próprios modos de produção de *significado* Paulo (2020). Nesse movimento, *o leitor* não julga o texto com base em suas referências pessoais, mas se esforça para entender o mundo pela perspectiva do *um autor* constituído, procurando identificar a lógica interna de seu discurso, bem como as legitimidades e *justificações* que sustentam suas afirmações.

Segundo Paulo (2020), o *descentramento* é frequentemente disparado pelo *estranhamento*, que ocorre quando *o leitor* se depara com algo (um *resíduo de enunciação*) que desafia suas próprias *crenças* e o leva a questionar suas certezas, conforme descrito por Luchetta (2017)[19]. Esse *estranhamento* permite que *o leitor* reconheça a existência de outras perspectivas, que, embora distintas das suas, são legítimas dentro de seus próprios contextos de produção. Assim, o *descentramento* não apenas facilita o reconhecimento de outras perspectivas, mas é crucial para que *o leitor* compreenda o *outro* em seus próprios termos.

Em síntese, o *descentramento* é fundamental para a *leitura plausível* na perspectiva do MCS, pois permite que *o leitor* supere a leitura pela falta – isto é, aquela que julga o discurso do outro pelo que ele "não tem" ou pelo que "não está certo" segundo seus próprios modos de produção de *significado*. Além disso, possibilita a construção da coerência do *um autor* constituído e favorece a internalização de novos modos de produção de *significado*, ampliando a compreensão e a elaboração de enunciações em diferentes *direções de interlocução*.

Na sequência, serão discutidos os movimentos da *leitura plausível*, aprofundando-se

[19] Em sua tese de doutorado, Luchetta (2017) traduziu e analisou a obra "Elementos de Álgebra" de Leonhard Euler. Nesse processo, a autora encontrou um obstáculo relacionado à sua própria formação matemática no IME/USP, que dificultava uma leitura fluida da obra. Suas crenças e modos de produção de *significado*, internalizados ao longo de sua trajetória acadêmica, a levavam a interpretar o texto pela "falta", concentrando-se nas diferenças entre a matemática de Euler e a matemática contemporânea (Paulo, 2022). No entanto, ao perceber essa dificuldade, Luchetta (2017) iniciou um processo de *descentramento*, esforçando-se para se distanciar de seus próprios referenciais e compreender a lógica interna e as *justificações* de Euler no contexto histórico em que sua obra foi produzida.

nos processos envolvidos nesse exercício.

3.2.3 Movimentos da leitura plausível desde a perspectiva do MCS

Após abordar as noções de *conhecimento* em primeira e terceira pessoa, bem como o *descentramento*, é possível delinear os movimentos que estruturam a *leitura plausível* à luz dos pressupostos do MCS. Paulo (2020) apresenta seis processos que organizam a *leitura plausível*, os quais podem ocorrer de maneira sobreposta ou simultânea. A seguir, detalham-se esses movimentos:

(i) Constituição de *um autor*: o primeiro movimento envolve o reconhecimento de *um autor*, um sujeito cognitivo, a partir de *resíduos de enunciação*. Em outras palavras, uma demanda por produção de *significados* é disparada a partir desses *resíduos* e, nesse processo, *o leitor* institui *um autor*, mesmo que este seja uma figura teórica ou um constructo elaborado pelo *o leitor*.

(ii) Constituição de *objetos*: a partir da demanda por produção de *significado* disparada pelo *um autor* constituído, *o leitor* identifica e delimita os *objetos* sobre os quais, supõe-se, o *um autor* do enunciado se refere. Esses *objetos*, concretos ou abstratos, precisam ser claramente delimitados para que a *leitura plausível* seja coerente e para que *o leitor* possa se orientar dentro do *campo semântico* supostamente proposto pelo *um autor*.

(iii) Constituição de um *núcleo*: no decorrer da *leitura plausível*, identifica-se um *núcleo* que aglutina as principais ideias do enunciado. Contudo, conforme destacado anteriormente, o *núcleo* não é fixo; ele se transforma à medida que *o leitor* avança no processo de produção de *significados*, ajustando sua compreensão conforme novas enunciações e *justificações* são introduzidas.

(iv) Realização de uma enunciação: com base na interação com os elementos anteriores, *o leitor* produz *significados*, expressando-os, por meio de uma enunciação. Esse movimento envolve a produção de novos *significados* que dialogam com o que foi constituído pelo *um autor*. A enunciação *do leitor*, assim, é fruto de uma reconstrução supostamente coerente e legítima do texto.

(v) Constituição de uma *direção de interlocução*: ao produzir sua leitura, *o leitor*

direciona sua enunciação a um *interlocutor*. É importante ressaltar que essa *direção de interlocução* pode ou não coincidir, *plausivelmente*, com a do *um autor* inicialmente instituído. Nesse contexto, estabelecer a *direção de interlocução* é essencial para delimitar as legitimidades do discurso e o horizonte de possibilidades da enunciação.

(vi) Antecipação de legitimidades: ao produzir a *leitura plausível*, *o leitor*, consciente ou inconscientemente, antecipa as possíveis respostas do *interlocutor*, com base nas legitimidades que acredita serem compartilhadas. Essa antecipação molda o processo de produção de *significados* e conduz o movimento da *leitura plausível*.

Esses seis processos evidenciam que o *conhecimento* em primeira pessoa – aquele que é baseado nas *crenças* e legitimidades do próprio leitor – desempenha um papel essencial na constituição *do leitor* como *interlocutor*. *O leitor* inicia o processo a partir de suas próprias legitimidades e repertórios. Contudo, para que a *leitura plausível* ocorra de forma efetiva, é necessário ir além da perspectiva individual e reconhecer a perspectiva do *um autor* instituído. Esse *descentramento* é fundamental para a realização de uma *leitura plausível* que respeite a coerência e lógica interna do texto.

Portanto, a *leitura plausível* é um movimento de produção de *significados*, que resulta do diálogo entre as perspectivas do *o leitor* e do *um autor*, mediado pelos contextos e *direções de interlocução* estabelecidas. A *leitura plausível*, nesse sentido, torna-se uma ferramenta possível para a produção de pesquisas acadêmicas, uma vez que promove uma interação dialógica e flexível entre diferentes modos de produção de *significado*.

Propondo uma discussão inserida nesse contexto, é importante ressaltar que a sistematização desses movimentos, conforme apresentada por Paulo (2020), não deve ser vista como uma tentativa de impor um caminho rígido ou prescritivo para a *leitura plausível*. Pelo contrário, a proposta do autor é um convite[20] para que pesquisadores explorem, compreendam e adaptem esses processos conforme as necessidades e contextos de suas próprias investigações. Essa flexibilidade será discutida na próxima seção, que aborda a *leitura plausível* como metodologia de pesquisa no campo da Educação Matemática.

[20] Termo oriundo de Viola dos Santos, Paulo e Julio (2023).

3.2.4 A leitura plausível como metodologia de pesquisa no contexto da Educação Matemática

A *leitura plausível*, fundamentada nos pressupostos do MCS, tem sido problematizada enquanto abordagem teórico-metodológica nas pesquisas em Educação Matemática. Paulo (2020, 2022) levanta reflexões acerca da complexidade dessa abordagem, questionando sua classificação enquanto método ou metodologia de pesquisa. Em particular, Paulo (2022) argumenta que a *leitura plausível* vai além de um conjunto fixo de procedimentos predeterminados, caracterizando-se como uma metodologia em virtude de sua base teórica e sua flexibilidade na adaptação ao contexto específico de investigação.

De maneira geral, Paulo (2022) afirma que um método pode ser entendido como um conjunto de técnicas e procedimentos rigidamente definidos, orientados por regras estabelecidas que conduzem o pesquisador de forma linear e controlada ao longo do processo investigativo. Por outro lado, segundo o autor, a metodologia envolve uma reflexão mais ampla e profunda sobre os caminhos e estratégias da investigação, estando intrinsecamente vinculada às teorias que fundamentam o trabalho do pesquisador.

Não obstante, a discussão sobre metodologia na Educação Matemática não se restringe à proposição de etapas para a realização de uma pesquisa, mas aborda a relação entre o objeto estudado, a concepção de conhecimento assumida pelo pesquisador e, a partir disso, as estratégias metodológicas coerentes com essa compreensão. Tendo isso em vista, a *leitura plausível* não deve ser compreendida como um método independente, pois sua aplicação não é possível sem a assunção das demais noções do MCS, em especial sua concepção de *conhecimento*. Ou seja, a *leitura plausível* não é uma técnica que pode ser utilizada em qualquer contexto investigativo de maneira neutra ou isolada, mas um movimento investigativo que emerge de uma postura epistemológica específica.

Dessa forma, a *leitura plausível* se distancia da rigidez das abordagens metodológicas tradicionais, que frequentemente seguem sequências fixas de etapas, e adota uma postura flexível, ajustando-se às particularidades e especificidades de cada contexto investigado. Conforme Paulo (2020) ressalta, a *leitura plausível* não se submete a um conjunto de procedimentos estanques, mas sim constitui um movimento investigativo

dinâmico, que se adapta às singularidades do fenômeno em análise.

A principal característica da *leitura plausível*, conforme discutido anteriormente, é sua busca por coerência nas enunciações dos sujeitos pesquisados, compreendendo seus modos de produção de *significado* com base em suas próprias *justificações* e legitimidades. Essa abordagem é especialmente adequada para a análise de fenômenos complexos, como os processos de ensino-aprendizagem, uma vez que reconhece e valoriza as perspectivas dos sujeitos dentro de seus contextos sociais e culturais. No entanto, um dos principais desafios enfrentados pela *leitura plausível* reside em sua validação em contextos acadêmicos que tendem a privilegiar métodos com procedimentos rígidos e predefinidos (Paulo, 2022). Apesar dessa tensão, a *leitura plausível*, como abordagem teórico-metodológica no âmbito do MCS, propõe uma alternativa mais flexível, que permite uma adaptação contínua às particularidades do contexto educacional.

Ao analisar quatro teses de doutorado que adotaram o MCS, Paulo (2022) observa que, mesmo sem seguir uma sequência metodológica rígida, esses trabalhos compartilham traços teóricos comuns, sugerindo a existência de uma consistência metodológica subjacente. Essa consistência, entretanto, respeita a construção de *conhecimento* dentro de contextos específicos, sem a imposição de um roteiro normativo ou prescritivo.

Além disso, Paulo (2020) propõe uma sistematização da *leitura plausível*, não com o intuito de criar um modelo metodológico fixo, mas como um conjunto de diretrizes que podem orientar outros pesquisadores no uso dessa abordagem. Essa sistematização, segundo o autor, visa fortalecer a legitimidade do MCS como referencial de pesquisa, oferecendo suporte para o desenvolvimento de investigações que preservem a flexibilidade característica da *leitura plausível*. Ao mesmo tempo, a sistematização proposta não deve ser vista como um conjunto de regras inflexíveis, mas como um apoio que permite a liberdade criativa e adaptativa do pesquisador.

Portanto, no contexto do MCS, a *leitura plausível* deve ser compreendida como uma abordagem teórico-metodológica intrinsecamente vinculada à concepção de conhecimento assumida pelo pesquisador. Não se trata de um método aplicável de maneira neutra e independente, mas de um movimento investigativo que emerge da articulação entre

pressupostos epistemológicos e estratégias de análise.

Ao proporcionar uma forma de sistematizar e tornar mais explícitos os processos envolvidos na *leitura plausível*, Paulo (2020, 2022) oferece uma perspectiva para que outros pesquisadores possam adotar e adaptar essa abordagem em suas investigações. Tal sistematização permite, ainda, que a *leitura plausível* seja validada e aplicada por outros pesquisadores no campo da Educação Matemática, sem comprometer a flexibilidade essencial que define essa abordagem.

Na sequência, realizam-se as considerações finais deste capítulo.

3.3 Considerações deste capítulo

Ao longo do primeiro capítulo, explorou-se o Pensamento Computacional sob diferentes perspectivas teóricas. Como argumentado, há uma lacuna teórica em relação à definição e mensuração do Pensamento Computacional em termos cognitivos. A proposta central deste estudo é justamente preencher essa lacuna ao adotar o MCS como uma perspectiva teórico-epistemológica que possibilita uma leitura suficientemente fina do processo de produção de *significados* envolvidos nesse tipo de pensamento.

A *leitura plausível*, fundamentada no MCS, emerge, portanto, como uma metodologia possível para a pesquisa, sensível à complexidade dos fenômenos cognitivos, inclusive os presentes no Pensamento Computacional. Conforme Paulo (2020), ao invés de priorizar a validação das respostas dos sujeitos em termos de certo ou errado, essa metodologia permite ao pesquisador compreender as *justificações* que sustentam tais respostas, revelando as *direções de interlocução* que os sujeitos instituem ao lidar com problemas diversos.

Na esfera educacional, essa abordagem se mostra particularmente relevante. Como discutido por Paulo (2022), o processo educativo pode ser compreendido como um ato de "colonização" do outro, na medida em que o aluno é guiado a adotar determinados modos de produção de *significados*. Após a *leitura plausível* das produções dos alunos, o educador, ao estabelecer objetivos educacionais, orienta o aluno a produzir *significados* específicos: "gostaria que você considerasse produzir *significados* para este objeto desta forma".

É crucial que esse processo não seja visto como uma imposição autoritária. Ao apresentar novos modos de produção de *significados*, o professor convida o aluno a ampliar seu repertório, abrindo caminhos para a internalização de novas legitimidades e navegação por diferentes *campos semânticos*. Cabe ao professor, nesse contexto, fomentar o desenvolvimento desses novos modos de produção de *significado*, atuando como um guia nesse processo.

Ao propor uma perspectiva de Pensamento Computacional à luz do MCS, este estudo não pretende afirmar que essa seja a única forma de se produzir *significados* para os objetos[21] e para as práticas computacionais. Ao contrário, busca-se oferecer uma alternativa que enriquece a compreensão do processo cognitivo envolvido no Pensamento Computacional no âmbito de educadores matemáticos.

Além disso, este trabalho propõe problematizar o próprio "pensamento" do Pensamento Computacional, destacando que esse aspecto constitui um ponto cego na pesquisa educacional. Conforme discutido anteriormente, Ribeiro, Foss e Cavalheiro (2020) argumentam que, para compreender o Pensamento Computacional, é necessário entender o que é computação. No entanto, este estudo adota uma abordagem alternativa, defendendo que, para compreender o Pensamento Computacional, é essencial abordar o que se entende por pensamento.

Dessa forma, busca-se evidenciar que, tal como frequentemente apresentado na literatura, a expressão "Pensamento Computacional" poderia ser mais bem descrita como "Procedimento Computacional", uma vez que sua ênfase recai mais sobre a atividade do sujeito do que sobre uma discussão aprofundada do pensamento em si. A partir dessa perspectiva, o MCS se apresenta como uma abordagem capaz de oferecer uma leitura significativamente precisa dos processos cognitivos subjacentes ao Pensamento Computacional, deslocando a atenção do fazer computacional para as formas como os sujeitos produzem *significados* nesse contexto.

No próximo capítulo, essa proposta será aprofundada, discutindo-se *uma* perspec-

[21] Aqui, o termo "objeto" é empregado de forma mais ampla, desvinculando de seu uso técnico no MCS, referindo-se a qualquer elemento ao qual o sujeito atribui *significados* em sua enunciação.

tiva de Pensamento Computacional fundamentada nos pressupostos do MCS.

4 O PENSAMENTO COMPUTACIONAL DESDE A PERSPECTIVA DO MODELO DOS CAMPOS SEMÂNTICOS

No primeiro capítulo deste trabalho, foram apresentadas e problematizadas perspectivas de Pensamento Computacional, sustentando a argumentação de que, de acordo com a literatura consultada, ainda não há uma construção teórica que permita definir, avaliar ou mensurar o Pensamento Computacional em termos efetivamente cognitivos.

Diante dessa lacuna teórica, propõe-se uma perspectiva de Pensamento Computacional a partir da perspectiva do MCS, fundamentada na possibilidade desse modelo oferecer "leituras suficientemente finas de processos de produção de significados" (Lins, 2012, p. 18). Essa escolha é justificada pela compreensão de que, sob essa perspectiva epistemológica, a cognição é entendida como um processo de produção de *significados* (Ferreira, 2020). Para viabilizar a proposta, o segundo capítulo apresentou e discutiu as noções centrais do MCS que sustentam a perspectiva adotada.

Este trabalho caracteriza-se, portanto, como uma pesquisa de desenvolvimento teórico, cuja problemática envolve a noção de Pensamento Computacional e a forma como seus processos se manifestam no âmbito educacional, especialmente na Educação Matemática. O objetivo deste capítulo é desenvolver uma construção teórica que possibilite compreender o Pensamento Computacional à luz do MCS. Nesse sentido, busca-se identificar quais são os processos de produção de *significado*, *núcleos* e *campos semânticos* que caracterizam cada um dos processos específicos do Pensamento Computacional, a saber: design, abstração, depuração, decomposição, reconhecimento de padrões e algoritmos.

Cabe salientar que a seleção desses seis processos fundamentais analisados neste trabalho não foi arbitrária, mas resultado de uma análise crítica da literatura sobre Pensamento Computacional, fruto de discussões sobre o tema no Grupo de Pesquisa Autômato e da produção de *significados dos autores* desta pesquisa. Embora, desde a perspectiva defendida neste trabalho, esses processos sejam interdependentes, cada um será discutido em seções específicas, de modo a evidenciar suas particularidades.

Caracterizar um tipo específico de pensamento – neste caso, o Pensamento Com-

putacional – sob a perspectiva do MCS envolve não apenas descrever suas manifestações, mas também identificar os processos cognitivos peculiares a essa forma de pensar (Lins; Gimenez, 1997). Assim, a construção teórica desenvolvida neste capítulo visa descrever o Pensamento Computacional em termos de ações e enunciações que emergem em diferentes atividades, configurando-se em enunciações que, *plausivelmente*, sugerem um processo de design, abstração ou reconhecimento de padrões, por exemplo. Com isso, a *leitura plausível* assume um papel fundamental na formulação de uma perspectiva de Pensamento Computacional no contexto do MCS, pois permite dialogar com autores que versam sobre o tema, bem como identificar, ao longo dos exemplos construídos, as enunciações que apontam para cada processo cognitivo.

> Desse modo, quando escrevemos, por exemplo, "Lins afirma que...", "Para Silva...", enfim, quando atribuímos qualquer autoria a Lins, Silva, ou qualquer um autor, em nossa escrita, compreenda, caro leitor, que estamos produzindo plausivelmente um mundo que acreditamos, baseados nas legitimidades que entendemos compartilhar com um autor que constituímos a partir da leitura de um conjunto de resíduos, aquele um autor também produziria (Paulo, 2020, p. 10).

O movimento da *leitura plausível*, portanto, é essencial para construir uma análise dos processos do Pensamento Computacional conforme compreendido no MCS, diferenciando quais enunciações caracterizam os processos de produção de *significado* do sujeito que indicam quando ele está "pensamento computacionalmente".

Nesse contexto, a análise de exemplos torna-se essencial, pois permite captar as particularidades de cada processo de produção de *significados* ao longo da atividade do sujeito, oferecendo indícios sobre a mobilização de um determinado processo do Pensamento Computacional. Ao investigar tais processos de produção de *significados*, é crucial examinar a relação entre os sujeitos e os objetos técnicos, uma vez que o Pensamento Computacional, conforme compreendido neste trabalho, não é produzido em um vácuo, mas emerge da interação entre sujeito, objetos técnicos e contexto.

Considerar essa interação é fundamental, visto que os processos do Pensamento Computacional se manifestam de modo singular conforme o contexto e o objeto técnico utilizado. Exemplos construídos com diferentes objetos técnicos – como Scratch, Python,

GeoGebra e Word – possibilitam observar como o Pensamento Computacional emerge em variadas atividades e como o processo de produção de significados depende das particularidades do objeto técnico, do contexto e das legitimidades envolvidas. Por essa razão, a perspectiva sociotécnica da tecnologia se mostra pertinente, ao integrar particularidades técnicas e culturais que permeiam o processo cognitivo em contextos específicos. A seguir, apresenta-se um breve panorama dessa perspectiva[1].

Na esfera educacional, abordagens como o instrumentalismo, o determinismo e a perspectiva sociotécnica orientam a compreensão da relação entre seres humanos e tecnologia, com implicações relevantes para a pesquisa acadêmica e a prática docente (Peixoto, 2015; Ferreira, 2020). A visão instrumentalista considera tecnologia como uma ferramenta neutra, subordinada à vontade humana (Ferreira, 2020). Em contrapartida, o determinismo tecnológico sugere que a tecnologia possui uma lógica própria que influencia a sociedade e gera efeitos independentes da ação humana (Peixoto, 2015), o que pode resultar em negligência quanto aos fatores sociais igualmente importantes.

Por sua vez, a perspectiva sociotécnica reconhece a interação entre aspectos técnicos e sociais na configuração dos objetos técnicos, concebendo-os como construções sociais que moldam e são moldadas pelas práticas e relações humanas (Peixoto, 2015). Esse enfoque permite uma análise mais abrangente sobre o impacto dos objetos técnicos como o GeoGebra, o Excel e o Scratch no Pensamento Computacional, considerando-os produtos da interação humana com a técnica que influenciam diretamente os processos de produção de *significados*.

Essas considerações sobre a perspectiva sociotécnica são centrais para fundamentar as teorizações sobre o Pensamento Computacional e os processos cognitivos que o constituem, além de orientar a construção dos exemplos analisados ao longo deste capítulo.

Portanto, para sustentar as discussões propostas, este capítulo organiza-se da seguinte maneira: inicialmente, apresenta-se a noção de design, fundamentada nas práticas computacionais de Brennan e Resnick (2012) e na formulação do problema de Dantas (2023);

[1] Para uma leitura mais aprofundada sobre a perspectiva sociotécnica da tecnologia, recomenda-se a consulta dos trabalhos de Peixoto (2015) e Ferreira (2020).

em seguida, discute-se sobre os processos de decomposição de produção de algoritmos; na sequência, elucidam-se os processos de abstração e reconhecimento de padrões; por fim, discorre-se sobre os erros de sintaxe e de semântica no contexto da depuração, com base nas considerações de Teixeira, Juvanelli e Dantas (2024).

4.1 Design desde a perspectiva do MCS

Conforme argumentado anteriormente, a perspectiva de Pensamento Computacional adotada neste trabalho defende que todos os processos envolvidos estão conectados em maior ou menor grau. No entanto, ao longo da construção desta investigação, observou-se que um dos processos – a formulação do problema – requer um aprofundamento maior, pois esse processo, tal como proposto, reflete de forma intrínseca a consideração de outros processos interligados.

Propõe-se, então, a noção de design em substituição à formulação do problema. Essa substituição não implica que a formulação do problema seja irrelevante para o Pensamento Computacional, mas destaca que, ao formular um problema, o sujeito simultaneamente traça estratégias de decisão que envolvem escolhas sobre objetos técnicos, contexto e modos de produção de *significado* – ações que extrapolam o escopo restrito da formulação. Assim, o conceito de design permite abordar a resolução de problemas não como uma sequência de etapas predeterminadas, mas como um conjunto de ações cognitivas interativas e dinâmicas, de modo que a formulação e o design se entrelaçam continuamente no processo.

A escolha de iniciar as discussões sobre os processos do Pensamento Computacional abordando o design possibilita, na perspectiva defendida neste trabalho, uma argumentação mais abrangente sobre os processos subsequentes, evidenciando como esses estruturam o pensamento do sujeito durante a resolução de problemas. Considera-se que o design não é um processo independente ou subsequente à formulação do problema, mas um processo que a acompanha e a complementa diretamente – o design perpassa a formulação do problema.

O objetivo desta pesquisa é apresentar *uma* perspectiva de Pensamento Computacional fundamentada no MCS. Para isso, discutir sobre o design no contexto desta

investigação permite teorizar sobre esse processo à luz das noções do MCS, sustentando que sua inclusão como um dos processos do Pensamento Computacional propicia uma compreensão aprofundada e fundamentada dos processos cognitivos – ou, na terminologia do MCS, dos processos de produção de *significado* – que embasam a resolução de problemas pelo Pensamento Computacional.

Dessa forma, o objetivo desta seção é sustentar que a formulação do problema e o processo de design são processos indissociáveis e essenciais para a resolução de problemas no Pensamento Computacional. Para tanto, a seção organiza-se da seguinte maneira: inicialmente, discorre-se sobre a formulação do problema no âmbito do Pensamento Computacional; em seguida, expande-se essa formulação ao integrar o processo de design; na sequência, teoriza-se sobre o design à luz das noções do MCS, proporcionando uma compreensão mais aprofundada e articulada desse processo; e por fim, apresentam-se as considerações finais desta seção.

4.1.1 A formulação do problema no Pensamento Computacional

A formulação do problema, no contexto do Pensamento Computacional, vai além da simples apresentação de um enunciado a ser resolvido. Como ressalta Dantas (2023), esse processo envolve identificar variáveis, necessidades e ações, bem como planejar e organizar as etapas para a resolução.

Segundo Dantas (2023), ao formular um problema, o sujeito não apenas define o que precisa ser resolvido, mas também estabelece uma sequência de ações cognitivas e operacionais que orientam a busca pela solução. Assim, a formulação do problema desempenha uma função essencial de planejamento, organizando as etapas e selecionando as ferramentas ou métodos a serem utilizados, estabelecendo as bases para as ações subsequentes.

É importante, contudo, distinguir entre o enunciado e a formulação de um problema. Enquanto o enunciado pode ser uma descrição textual, a formulação refere-se à forma como o sujeito produz *significados*, organiza e define essa situação, estabelecendo uma *direção de interlocução* que ele considera legítima. Dessa forma, diferentes indivíduos podem formular

problemas distintos a partir de um mesmo enunciado, dependendo das legitimidades que operam, dos objetos técnicos envolvidos e do contexto.

Para ilustrar, considere o enunciado: "Determine o ponto de interseção entre as retas $r : y = 2x + 1$ e $s : y = -x + 4$". Dependendo da abordagem adotada, podem surgir diferentes formulações e estratégias de resolução. Por exemplo:

Resolução manual (algébrica): Um sujeito que opta pela resolução manual, utilizando apenas lápis e papel, pode formular o problema com foco na manipulação de equações algébricas para encontrar o ponto de interseção. Utilizando técnicas como substituição e eliminação de variáveis, ele pode determinar os valores de x e y que satisfaçam ambas as equações. Nesse caso, a formulação do problema poderia ser: "Como manipular as duas equações algébricas para encontrar os valores de x e de y que satisfaçam ambas?"

Resolução com GeoGebra (visualização gráfica): Outro sujeito, ao utilizar o GeoGebra, pode inserir as equações no ambiente gráfico e visualizar imediatamente o ponto de interseção. Aqui, a formulação do problema poderia ser expressa como: "Como inserir e visualizar graficamente as retas e identificar o ponto de interseção?"

Resolução com Excel (abordagem tabular): Um terceiro sujeito pode utilizar o Excel para criar uma tabela que compare os valores de y para diferentes valores de x, identificando o ponto em que as coordenadas das retas coincidem. A formulação do problema, nesse caso, poderia ser: "Como usar o Excel para criar uma tabela que compare os valores das equações e identifique o ponto em que x possui y_1 e y_2 iguais?"

Esses exemplos mostram como a escolha[2] do meio de resolução influencia diretamente a formulação e o processo de resolução. Diferentes objetos técnicos – como lápis e papel, GeoGebra e Excel – podem desempenhar papéis distintos na maneira como os problemas são formulados e solucionados e, por conseguinte, influenciar os processos envolvidos no Pensamento Computacional. A escolha do objeto técnico não é apenas uma questão de conveniência ou preferência; ela é também um fator determinante na forma como o sujeito estrutura o problema e encontra soluções.

[2] A própria ação de escolher o meio de resolução, se este é "adequado" ou não, corresponde a um processo de abstração e será problematizado com maior profundidade em uma seção posterior.

Essa escolha, no entanto, não é neutra; ela influencia a forma como o sujeito articula seu entendimento do problema, constrói suas estratégias e supera obstáculos ao longo do processo. Ferreira (2020) argumenta que os objetos técnicos podem transformar tanto o processo de resolução quanto o próprio problema. Por exemplo, o uso de objetos técnicos como o GeoGebra ou o Excel implica diferentes formas de abstrair e estruturar o problema, alterando a natureza das ações subsequentes.

No caso da resolução manual das equações de reta, supõe-se que o aluno possa utilizar um método algébrico, como a substituição de variáveis, para encontrar o ponto de interseção. No entanto, a resolução manual pode assumir várias formas, e restringir a análise à álgebra simbólica não representa a diversidade de métodos que podem ser empregados. Resolver manualmente um problema como o da interseção de duas retas não implica, necessariamente, o uso exclusivo da álgebra. Um aluno pode optar por uma abordagem geométrica, desenhando as retas em um gráfico e determinando visualmente o ponto de interseção. Nesse caso, o foco estaria nas propriedades visuais e geométricas das retas, substituindo a manipulação simbólica por uma análise gráfica.

Reconhecer essas diferentes abordagens problematiza a ideia de que a resolução manual implica, necessariamente, uma única estratégia. A escolha da álgebra como a primeira opção não é inevitável, mas reflete tanto as práticas de ensino quanto as preferências e experiências dos sujeitos.

Nesse contexto, observa-se que a formulação do problema envolve também fatores sociais e culturais. O contexto educacional, as práticas pedagógicas e as interações entre alunos e professores influenciam como um problema será formulado e, consequentemente, resolvido. Determinadas práticas de ensino podem levar os alunos a adotar abordagens específicas ou utilizar determinados objetos técnicos.

Essas práticas moldam a maneira como os alunos produzem *significado* para os problemas que encontram. Em um contexto de ensino tradicional, por exemplo, no qual a álgebra simbólica é fortemente enfatizada, os alunos podem ser incentivados a formular problemas exclusivamente com base em métodos algébricos. No entanto, essa ênfase em abordagens específicas restringe o espectro de formulações possíveis. Quando

expostos a um conjunto limitado de técnicas, como o método algébrico, os alunos tendem a formular problemas de acordo com o que já conhecem, o que limita a exploração de outras formas de resolução, como representações visuais ou numéricas. Esse foco em determinados procedimentos pode levar à cristalização de certas práticas, fazendo com que os alunos acreditem que existe apenas um "caminho correto" para formular e resolver um problema, o que inibe a criatividade e a experimentação.

Portanto, ao se referir às práticas de ensino que levam os alunos a adotarem abordagens específicas, observa-se que essas práticas moldam a maneira como os alunos abordam e formulam os problemas, impondo um conjunto limitado de ferramentas e métodos. A formulação de problemas, assim, não é um ato isolado, mas um processo influenciado pelo contexto sociocultural.

Na sequência, discute-se a escolha em adotar a terminologia design em vez de formulação do problema.

4.1.2 O processo de design no Pensamento Computacional

Apesar da importância da formulação do problema, sua análise, conforme a perspectiva adotada neste trabalho, implica uma articulação com processos subsequentes, intrinsecamente relacionados à resolução. A própria natureza da formulação envolve ajustes, refinamentos e explorações que se desdobram ao longo do desenvolvimento da solução. Como afirma Dantas (2023):

> Durante o processo de formulação do problema podem ser feitas as seguintes perguntas:
> – O que considerar?
> – Quais são as variáveis?
> – Que ações executar? (Dantas, 2023, p. 153).

Acredita-se que as respostas a essas perguntas impliquem processos subsequentes à formulação em si, envolvendo, por exemplo, abstração, decomposição e depuração. Pode-se inferir que, ao responder "O que considerar?", o sujeito utiliza a abstração para identificar os elementos mais relevantes do problema, deixando de lado os detalhes não essenciais. Ao definir "Quais são as variáveis?", ele recorre à decomposição, fragmentando o problema em

partes ou componentes menores que podem ser explorados separadamente, o que facilita a compreensão e o manejo de cada aspecto. Ao questionar "Que ações executar?", ele pode mobilizar os processos de produção de algoritmos e depuração, pois, ao decidir quais ações serão tomadas, o sujeito antecipa passos que poderão ser ajustados conforme os resultados parciais, prevendo a necessidade de revisar e corrigir etapas ao longo do processo.

Desse modo, as respostas a essas perguntas não apenas delineiam a formulação inicial do problema, mas abrem espaço para processos como a abstração, decomposição, produção de algoritmos e depuração, durante o desenvolvimento da resolução. Propõe-se, então, a noção de design como uma abordagem que permite integrar a formulação inicial com os processos subsequentes que emergem ao longo da resolução.

Neste trabalho, a noção de design é fundamentada nas práticas computacionais descritas por Brennan e Resnick (2012) e nos processos do Pensamento Computacional delineados por Dantas (2023). Embora Brennan e Resnick (2012) utilizem a terminologia "práticas computacionais" para se referir aos modos de pensar e agir no desenvolvimento de projetos computacionais, aqui opta-se pela expressão "processos do Pensamento Computacional". Essa escolha visa alinhar a terminologia ao foco central da pesquisa, que considera o Pensamento Computacional como um processo cognitivo amplo, envolvendo não apenas a criação de soluções computacionais, mas também a maneira como o sujeito formula, refina e implementa estratégias de resolução de problemas em contextos variados. O uso de "processos" reforça a ideia de que o Pensamento Computacional não se limita a habilidades técnicas, mas abrange ações cognitivas e reflexivas em constante interação com os objetos técnicos utilizados e o contexto.

Propõe-se, então, a seguinte definição de design[3]: um processo de resolução de problemas no âmbito do Pensamento Computacional, no qual o sujeito formula, reformula, revisa, ajusta e aprimora continuamente sua abordagem à medida que interage com o problema, com os objetos técnicos, com o contexto e mobiliza seus repertórios.

[3] A definição de design proposta neste trabalho é inspirada nas discussões apresentadas em Denning e Tedre (2019). Contudo, enquanto esses autores discutem o design no contexto do Pensamento Computacional aplicado a cientistas da computação e engenheiros de software, este trabalho adapta o conceito para uma perspectiva educacional voltada a educadores e professores de matemática.

Adotar a perspectiva do design permite reconhecer que o processo de resolução raramente segue uma sequência linear. À medida que novas informações surgem e o entendimento do problema se aprofunda, a formulação inicial pode exigir ajustes ou revisões. Por exemplo, ao formular um problema no Scratch, o sujeito pode perceber que as ferramentas disponíveis não são adequadas para alcançar certos objetivos, o que pode demandar a reestruturação do problema ou até mesmo uma mudança de ferramenta, como ao migrar do Scratch para o GeoGebra. Assim, a interação entre formulação e design é continua, com ambos os processos influenciando-se mutuamente ao longo do desenvolvimento da solução.

Neste trabalho, os processos do Pensamento Computacional – abstração, decomposição, reconhecimento de padrões e depuração – estão intrinsecamente ligados ao design, pois envolvem a construção e adaptação da solução ao longo da resolução do problema. Em seguida, cada um desses processos é discutido brevemente, uma vez que, embora integrados ao design, possuem nuances que serão pormenorizadas em seções específicas.

A abstração é o processo mental em que o sujeito foca nos elementos essenciais e suficientes para a resolução de um problema, desconsiderando dados ou variáveis irrelevantes (Dantas, 2023). Esse processo permite ao sujeito construir uma representação simplificada do problema, isolando as características fundamentais que orientam a resolução. No contexto do design, a abstração também implica a habilidade de alternar entre diferentes níveis de complexidade, ajustando a solução conforme as necessidades do problema e do ambiente em que será implementado (Brennan; Resnick, 2012; Dantas, 2023).

A decomposição consiste em dividir um problema maior e mais complexo em partes menores e mais manejáveis, possibilitando ao sujeito focar na resolução de cada componente individualmente antes de integrá-los na solução final. Ao fragmentar o problema em subproblemas, o processo de decomposição facilita a análise, o planejamento e a execução da resolução, tornando o problema menos denso e mais acessível (Dantas, 2023). Dessa forma, o sujeito pode reunir os componentes resolvidos para construir uma solução robusta e bem estruturada.

O reconhecimento de padrões é o processo que permite ao sujeito identificar

elementos recorrentes, comportamentos previsíveis ou estruturas familiares dentro do problema. Esse processo se manifesta tanto na identificação de similaridades entre problemas anteriormente resolvidos quanto na percepção de elementos constantes ou variáveis em um conjunto de dados ou estruturas associadas ao problema (Dantas, 2023). Essa habilidade possibilita ao sujeito reaproveitar soluções anteriores, adaptando-as ao contexto atual e promovendo soluções mais eficientes e ágeis. Assim, o reconhecimento de padrões está intimamente ligado à prática de reutilização, favorecendo a adaptação de estratégias previamente bem-sucedidas, o que maximiza a eficácia do design (Brennan; Resnick, 2012).

Por fim, a depuração é o processo de identificação e correção de erros, indo além da busca por uma solução tecnicamente correta e abrangendo a adaptação da solução ao contexto e aos recursos disponíveis. Este processo reflete a flexibilidade e adaptabilidade do design, envolvendo as ações de testagem, verificação, refinamento e otimização da resolução apresentada. Dessa forma, os resultados, sejam eles parciais ou finais, são continuamente avaliados, com hipóteses descartadas e ideias reformuladas, revisadas ou, até mesmo, abandonadas, quando necessário (Teixeira; Juvanelli; Dantas, 2024).

Assim, o design amplia a visão sobre o problema, promovendo a adaptação contínua das estratégias, a exploração de novas ferramentas e a reorganização dos caminhos para a solução à medida que o sujeito interage com o problema. Mudar o foco das discussões da formulação do problema para o design permite, portanto, uma abordagem mais flexível dos processos do Pensamento Computacional.

Considerando a complexidade dos processos de formulação do problema e de design, bem como a influência dos meios de resolução, propõe-se, na sequência, uma análise mais aprofundada desses processos sob a perspectiva do MCS. Ao integrar as noções do MCS, busca-se compreender como os aspectos cognitivos e sociais envolvidos na resolução de problemas podem ser mais bem compreendidos e explorados em diferentes contextos.

4.1.3 O processo de design desde a perspectiva do MCS

Considerando que o MCS permite uma leitura suficientemente fina de processos cognitivos, ou seja, da produção de *significados* (Lins, 2012; Ferreira, 2020), é pertinente

aprofundar e problematizar a formulação do problema e o processo de design com base nos pressupostos dessa perspectiva epistemológica. Isso possibilita deslocar o foco da simples transposição de um enunciado para sua resolução, conforme será discutido a seguir.

No contexto do MCS, a formulação do problema pode ser compreendida como um processo em que o enunciado, ao disparar uma demanda por produção de *significado* pelo sujeito em uma atividade, torna-se um *resíduo de enunciação*, desencadeando o processo de produção de *significados*. O processo de design, por sua vez, refere-se à produção e manutenção de *significados* ao longo da resolução do problema, sendo moldado pela interação entre o sujeito, o contexto, os objetos técnicos disponíveis e seus repertórios. Esse processo orienta-se de acordo com o *núcleo* e o *campo semântico* no qual o sujeito opera.

Conforme destacado anteriormente, diferentes sujeitos podem formular problemas distintos a partir de um mesmo enunciado. No âmbito do MCS, isso se explica porque as legitimidades, as *direções de interlocução*, os objetos técnicos disponíveis e o contexto influenciam os *significados* que o sujeito produz e, consequentemente, a formulação do problema. A formulação – e, portanto, o design – é contextual e moldada pela interação entre o sujeito, os objetos técnicos, a atividade e as legitimidades envolvidas.

No MCS, o *núcleo* refere-se a estipulações locais que funcionam como verdades momentâneas dentro de uma determinada atividade (Lins, 2012). Esse *núcleo* emerge da própria atividade e não corresponde a um conjunto pré-definido de elementos ou *crenças*, podendo ser alterado ao longo da atividade. O *campo semântico*, por sua vez, é o próprio processo de produção de *significados* em relação a esse *núcleo*. No processo de design, o *núcleo* se constitui a partir da interação do sujeito com o enunciado, os objetos técnicos e o contexto, atuando como um conjunto de *crenças-afirmações* que orientam suas ações e escolhas de resolução.

Embora os objetos técnicos e o contexto participem desse processo de formulação, eles não são determinantes em si mesmos; as legitimidades envolvidas e os repertórios que o indivíduo possui têm papel central nesse processo. Dessa forma, a interação entre o sujeito, os objetos técnicos, o contexto e as legitimidades molda o processo de design.

O MCS fornece, assim, um quadro de referência para compreender como os

objetos técnicos, aliados ao contexto e às legitimidades envolvidas, influenciam tanto o processo de produção de *significados* quanto o design. A resolução manual, por exemplo, tende a direcionar o aluno a uma formulação centrada nas regras simbólicas que lhe são familiares, enquanto o uso de objetos técnicos como o GeoGebra ou o Excel pode favorecer uma formulação mais aberta à experimentação com representações gráficas e numéricas. Contudo, a utilização dos termos "tende" e "pode" em relação aos meios de resolução é intencional, pois essa questão será posteriormente problematizada.

É essencial, portanto, compreender que as legitimidades e as *direções de interlocução* do sujeito moldam sua interação com o problema. Ao produzir *significados* para o enunciado e mobilizar ferramentas, o sujeito formula o problema e idealiza os caminhos de solução que ele vislumbra. O *núcleo*, composto pelas *crenças-afirmações* e *justificações* tomadas como "verdades momentâneas" no interior da atividade (Lins, 2012), orienta o processo de design na busca por soluções coerentes com as legitimidades envolvidas.

Para ilustrar o processo de design à luz do MCS, considere a seguinte situação: o professor solicita aos alunos que utilizem o Scratch para generalizar o processo de construção de qualquer polígono regular com n lados, registrando o processo de desenvolvimento do projeto. O enunciado apresentado foi: "Crie um programa no Scratch que permita ao usuário escolher a quantidade de lados de um polígono regular, e que o desenhe automaticamente na tela. Registre os passos que você percorreu no processo de construção e elabore uma descrição de como o Scratch foi utilizado na resolução"[4].

Nesse contexto, suponha que Ana[5] tenha elaborado o seguinte registro[6], conforme

[4] O enunciado dessa situação, bem como das demais que serão posteriormente apresentadas, são inspirados nas tarefas propostas no curso do GeoGebra, nas quais os alunos eram instruídos da seguinte maneira: "Ao resolver o problema, poste o arquivo que construiu acompanhado de uma descrição de como o GeoGebra foi utilizado por você em sua resolução" (Ferreira, 2020, p. 131). A escolha por um enunciado análogo permite explorar e problematizar as discussões acerca do processo de construção das soluções, destacando como a explicitação do uso de um determinado objeto técnico pode evidenciar o desenvolvimento do processo de design sob a ótica do MCS. Informações sobre o curso do GeoGebra podem ser encontradas por meio do link: https://ogeogebra.com.br/site/. Acesso em: 27 out. 2024.

[5] Todos os exemplos apresentados ao longo deste capítulo foram elaborados pelos autores deste trabalho, com o propósito de modelar situações pedagógicas específicas que facilitam a análise e compreensão dos processos do Pensamento Computacional no âmbito do MCS.

[6] Ao longo dos quadros apresentados neste capítulo, há menção a algumas figuras, destacadas entre colchetes. Opta-se por apresentá-las no decorrer do texto, e não de imediato, para possibilitar uma contextualização detalhada de cada imagem, favorecendo a análise sequencial e fundamentada das ações representadas.

ilustra o Quadro 4.1[7].

Quadro 4.1 – Registro de Ana

Direção de interlocução	**Enunciação**
1ª direção	Comecei o meu projeto desenhando um triângulo, pois é o polígono mais simples para começar [Figura 1].
2ª direção	No entanto, ao rodar o código, fiquei surpresa ao ver que o resultado não correspondia às minhas expectativas [Figura 2]. Então, decidi tentar desenhar um quadrado. A ideia era testar outro polígono para comparar os resultados e identificar onde estava o erro [Figura 3].
3ª direção	Ao investigar, percebi que o problema estava no ângulo de giro [Figura 4]. O Scratch considera o ângulo externo do polígono, então tive que ajustar para o ângulo suplementar. Com essa correção, consegui resolver o problema do triângulo [Figura 5]. A partir daí, passei a desenhar mais alguns polígonos, como o pentágono e o hexágono.
4ª direção	Para deixar meu projeto ainda mais interativo, pensei em uma maneira de permitir que o usuário escolhesse qual polígono gostaria de gerar, utilizando o bloco `[se, então]` [Figura 6]. Estou animada para ver como isso vai funcionar!

Fonte: Elaborado pelo autor (2025).

Desde a perspectiva do MCS, os *resíduos de enunciação* produzidos por Ana sugerem que ela enuncia em diferentes *direções de interlocução*, conforme destacado no Quadro 4.1. A análise do processo de design de Ana será dividida em partes: inicialmente, examinando cada *direção de interlocução* separadamente e, posteriormente, finalizando com considerações gerais sobre o processo de design.

Ana inicia com a enunciação "Comecei o meu projeto desenhando um triângulo, pois é o polígono mais simples para começar [Figura 1]" (Registro de Ana). No âmbito do MCS, essa enunciação pode ser estruturada como um *conhecimento*:

C_1 = ("Desenhar um triângulo [no Scratch]", "pois é o polígono mais simples para começar").

Essa *crença-afirmação* reflete, possivelmente, a formulação inicial do problema por

[7] A organização dos registros no formato de quadro, estruturada de modo que a coluna da direita apresenta a enunciação e a da esquerda expõe a *direção de interlocução* corresponde, é inspirada no trabalho de Ferreira (2020).

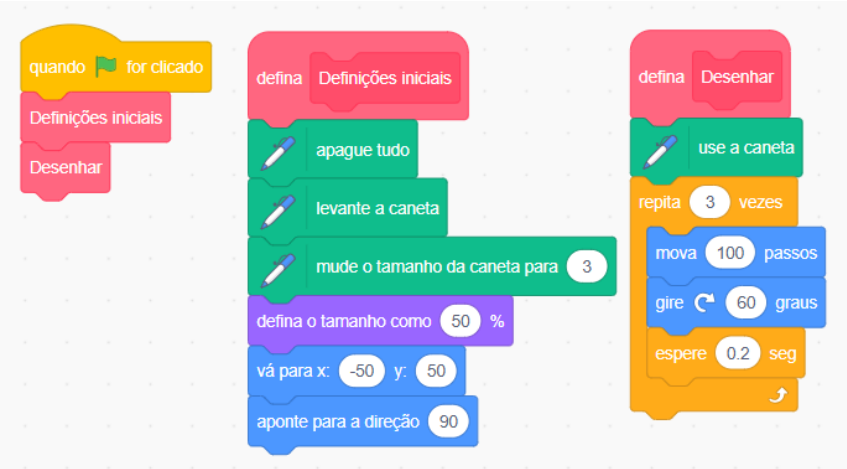
Ana. A explicitação da *justificação* revela a *direção de interlocução* para a qual Ana enuncia, legitimando a escolha de começar a construção do projeto pelo triângulo, considerando por ela como o polígono "mais simples". Ao enunciar nessa *direção de interlocução*, é legítima a escolha do triângulo como um ponto de partida adequado.

A escolha do triângulo sugere um processo de abstração no contexto do Pensamento Computacional, na medida em que Ana utiliza um polígono de menor complexidade como base para construir os demais. Nesse caso, a simplicidade do triângulo, caracterizada por seu menor número de lados, parece justificar sua escolha. Assim, o triângulo, com suas propriedades geométricas, é o *objeto* constituído por Ana, sobre o qual ela produz *significado*.

A *crença* de que "o triângulo é o polígono mais simples para começar" se revela como uma estipulação local, ou seja, uma "verdade absoluta" dentro do contexto da atividade, que não necessita de justificação adicional. A partir desse *núcleo*, Ana inicia a produção de *significado* para o triângulo, influenciando suas ações subsequentes. O *campo semântico* constituído aqui envolve as propriedades geométricas dos polígonos.

Essas considerações sobre a primeira *direção de interlocução* são essenciais para entender como Ana organiza sua ação e raciocínio. Na sequência, apresenta-se a Figura 4.1, correspondente à Figura 1 citada por Ana e exibe a primeira versão do código desenvolvido por ela para desenhar o triângulo.

Figura 4.1 – Primeira versão do código desenvolvido por Ana

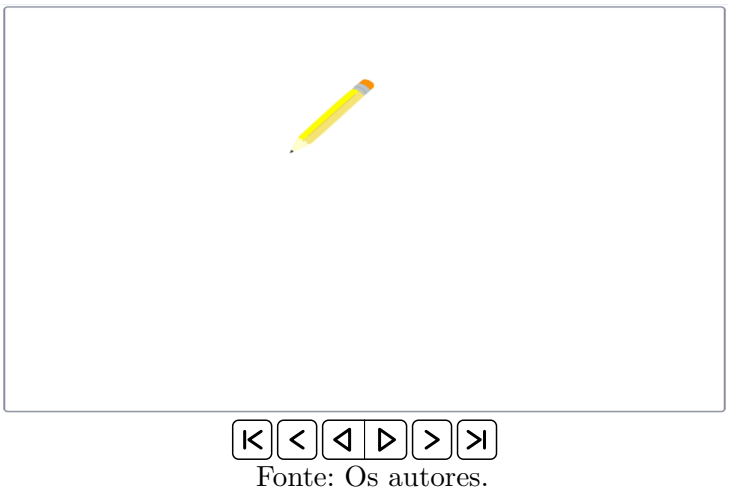


Fonte: Os autores.

Na Figura 4.1, observa-se que Ana utiliza a ferramenta Caneta, uma funcionalidade do Scratch que permite desenhar na tela usando um ator como "caneta". O código contém três sequências de blocos: a primeira define as configurações do projeto, ajustando a espessura do traço, além de definir o tamanho, a posição e a direção do ator para iniciar o desenho; a segunda contém os blocos que executam o desenho do triângulo, utilizando o comando `[repita (3) vezes]`, uma vez que o triângulo possui três lados, e o comando `[gire (60) graus]` para formar os ângulos internos; a terceira parte conecta esses dois grupos de comando, organizando o código. Ana ainda inclui um pequeno atraso por meio do bloco `[espere (0.2) segundos]`, permitindo que a construção do triângulo seja visualmente clara durante a execução.

Após executar o código e visualizar o resultado apresentado na tela, Ana se depara com uma discrepância entre o triângulo idealizado e o resultado visual obtido, que não corresponde ao esperado, conforme ilustra a Figura 4.2, representando a Figura 2 elencada por Ana.

Figura 4.2 – Resultado da primeira tentativa de desenhar um triângulo



Fonte: Os autores.

Esse momento é crucial na atividade de Ana, marcado por uma ruptura em suas expectativas e gerando um *estranhamento*. A enunciação seguinte – "No entanto, ao rodar o código, fiquei surpresa ao ver que o resultado não correspondia às minhas expectativas [Figura 2]" (Registro de Ana) – evidencia uma mudança na *direção de interlocução*. A falha em produzir um triângulo a partir do código gerado leva Ana a reavaliar a validade de seus pressupostos iniciais.

Para Ana, o código parecia estar correto, sugerindo que a discrepância reside na relação entre o código e a figura gerada, e não na escrita do código em si. Esse *estranhamento* coloca em xeque as legitimidades que sustentavam sua atividade até então, levando-a a reconsiderar os *significados* produzidos sobre triângulos e sobre a própria linguagem de programação.

A simplicidade do triângulo, que inicialmente norteava suas ações, é confrontada com a complexidade inesperada do código e o funcionamento do Scratch. Diante da impossibilidade de produzir a figura idealizada, Ana reconhece que a simplicidade do triângulo não garante o sucesso da programação, levando-a a reavaliar sua abordagem e buscar novos *significados*.

Esse movimento de busca por compreensão se manifesta na enunciação: "Então, decidi tentar desenhar um quadrado. A ideia era testar outro polígono para comparar os resultados e identificar onde estava o erro [Figura 3]" (Registro de Ana). Essa nova estratégia reflete a mudança na *direção de interlocução* e na forma como Ana produz *significado* no contexto da atividade.

O novo *conhecimento* de Ana pode ser representado da seguinte forma:

C_2 = ("Desenhar um quadrado [no Scratch]", "[pois] se eu testar outro polígono, posso comparar os resultados e identificar onde estava o erro").

A *crença-afirmação* "Desenhar um quadrado", acompanhada de sua *justificação*, sugere a reformulação do problema inicial, característica do processo de design. Ana constitui um novo *objeto* na atividade: o quadrado como ferramenta de depuração. A *justificação* explícita – "testar outro polígono para comparar os resultados e identificar o erro" – revela que ela enuncia para uma *direção de interlocução* que reconhece a experimentação e a comparação como modos legítimos de produção de *significado*.

Nesse momento, Ana, enuncia em uma *direção de interlocução* que validaria a escolha do quadrado como estratégia para depuração. Nessa *direção de interlocução*, é legítimo compreender que a regularidade do quadrado, com lados e ângulos iguais, facilita a comparação com o triângulo e auxilia na identificação do erro.

Comparando a primeira enunciação de Ana, na qual ela afirma que “o polígono mais simples para começar” é o triângulo, com a segunda, em que “decide desenhar um quadrado” para “comparar os resultados e identificar onde estava o erro”, nota-se uma mudança na forma como Ana produz *significado* ao longo da atividade.

Na segunda enunciação, Ana demonstra ter internalizado modos de produção de *significado* mais alinhados ao que pode ser feito no Scratch, como “rodar o código” e “identificar o erro”. Essa internalização se reflete em sua atenção aos detalhes do código e no uso da comparação como ferramenta de depuração. Ana passa a explorar hipóteses e soluções para o problema.

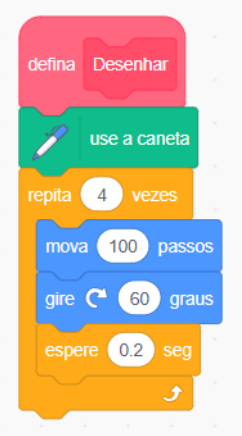
Essa mudança na *direção de interlocução* também implica uma modificação no *núcleo* de sua atividade. A simplicidade geométrica do triângulo, que antes orientava suas ações, é relativizada diante da complexidade da programação. É *plausível* inferir que Ana percebe que a simplicidade de uma forma geométrica não garante a viabilidade de seu código, levando-a a considerar novos fatores, como a lógica do código e a funcionalidade do Scratch.

A *crença* de que “o triângulo é o polígono mais simples para começar” se altera para incluir a *crença* de que a experimentação e a comparação são ferramentas eficazes para resolver problemas de programação. A nova estratégia de design de Ana não necessita de *justificações* adicionais, refletindo uma internalização das práticas legítimas que guiam suas ações. A postura de Ana passa a ser de alguém que utiliza a experimentação como parte essencial no processo de depuração.

O *campo semântico* da atividade também se amplia, incorporando conceitos relacionados à depuração de código – como testar, comparar e identificar erros –, ao mesmo tempo em que mantém os polígonos como elementos centrais da atividade. Assim, a atividade de Ana articula conceitos geométricos e a lógica de programação no Scratch.

Na sequência, a Figura 4.3 apresenta o código ajustado para desenhar o quadrado, correspondente à Figura 3 mencionada por Ana.

Figura 4.3 – Código ajustado para desenhar um quadrado

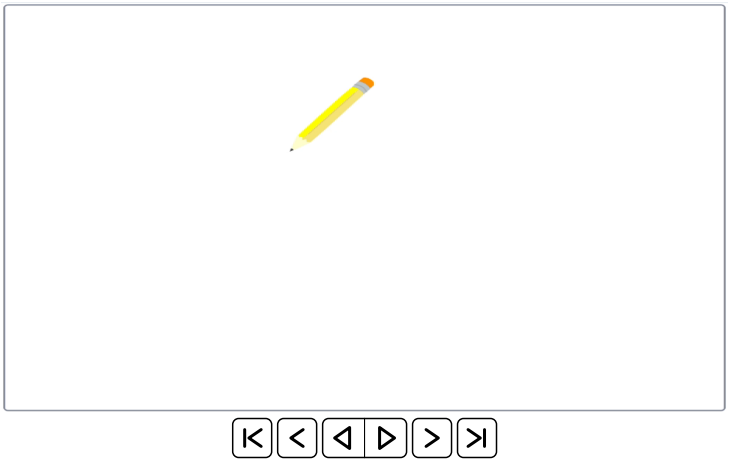


Fonte: Os autores.

Na Figura 4.3, é possível observar como Ana adaptou seu código para desenhar um quadrado, ajustando a quantidade de lados no bloco `[repita]` de 3 para 4 e alterando o ângulo de giro de 60° para 90°. Esses ajustes ilustram o processo de depuração que Ana utiliza ao modificar o código para diferentes polígonos, mantendo sua estratégia de comparação e experimentação como eixo central do processo de design.

Após modificar o código e obter o resultado esperado, conforme ilustrado na Figura 4.4 (correspondente à Figura 4 citada por Ana), Ana prossegue com suas enunciações, revelando uma nova mudança em seu processo de produção de *significados*.

Figura 4.4 – Quadrado desenhado corretamente no Scratch



Fonte: Os autores.

Na enunciação seguinte, Ana expõe:

> Ao investigar, percebi que o problema estava no ângulo de giro [Figura 4]. O Scratch considera o ângulo externo do polígono, então tive que ajustar para

> o ângulo suplementar. Com essa correção, consegui resolver o problema do triângulo [Figura 5]. A partir daí, passei a desenhar mais alguns polígonos, como o pentágono e o hexágono (Registro de Ana).

A partir dessa enunciação, é *plausível* inferir que Ana demonstra ter compreendido a lógica específica do Scratch na construção de polígonos, especialmente no que se refere aos ângulos.

Esse novo *conhecimento* de Ana pode ser representado da seguinte forma:

C_3 = ("O Scratch considera o ângulo externo do polígono", "então tive que ajustar para o ângulo suplementar para resolver o problema do triângulo").

A enunciação "ajustar para o ângulo suplementar", justificada pela compreensão de que "o Scratch considera o ângulo externo do polígono", revela que Ana internalizou a lógica particular do Scratch para a construção de polígonos. Ela reconhece que a medição e a utilização dos ângulos no Scratch diferem de sua compreensão anterior dos ângulos na geometria tradicional, o que a leva a adaptar seu código para essa nova lógica.

Observa-se novamente uma mudança na *direção de interlocução*. Ana passa a enunciar em uma *direção de interlocução* que compreende a lógica de programação do Scratch e possui familiaridade com conceitos geométricos, como ângulos externos e ângulos suplementares. Essa compreensão compartilhada minimiza a necessidade de Ana explicitar tais conceitos, pois a *direção de interlocução* pressupõe um repertório matemático e de programação comum que legitima suas enunciações.

A estrutura da frase de Ana reflete essa pressuposição. Ao enunciar "O Scratch considera o ângulo externo do polígono, então tive que ajustar para o ângulo suplementar", ela estabelece uma relação de causa e consequência que pressupõe a compreensão da lógica subjacente ao funcionamento do Scratch e à relação entre os conceitos de ângulos externos e suplementares. O uso do conectivo "então" sugere que a ação de ajustar o ângulo para o suplementar é vista como uma consequência natural da forma como o Scratch processa os ângulos. Ana demonstra, assim, ter internalizado a lógica específica do programa e a utiliza como base para justificar suas ações.

A *direção de interlocução* para qual Ana enuncia pressupõe um entendimento

acerca da matemática envolvida na construção, permite que ela concentre sua explicação na lógica da programação no Scratch, dispensando uma exploração mais detalhada dos conceitos geométricos. Seu discurso é legitimado pela *direção de interlocução* que valida os conceitos matemáticos sem a necessidade de explicações extensivas. Ana não precisa justificar suas escolhas em termos de conceitos geométricos "básicos", pois assume que nessa *direção de interlocução* é legítimo possuir esse repertório. Isso possibilita que ela se concentre em aspectos mais específicos da programação, como a forma como o Scratch lida com ângulos e a necessidade de adaptar o código a essa lógica.

Com a internalização da lógica do Scratch, o *objeto* da atividade – a construção de polígonos – torna-se mais complexo. Ana passa a considerar não apenas as propriedades geométricas dos polígonos, mas também a forma como essas propriedades são traduzidas para a linguagem de programação do Scratch. A relação entre a geometria e a lógica de programação se torna um elemento central em seu processo de design.

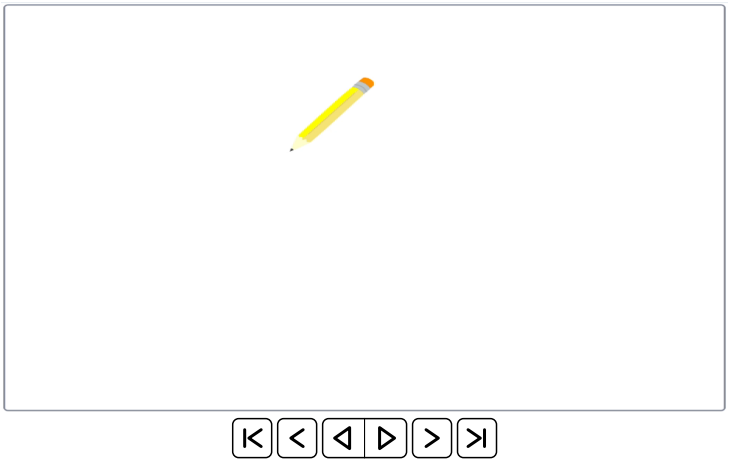
Ao analisar o resultado da interação com o Scratch, conforme ilustrado na Figura 4.4, Ana experimenta uma modificação no *núcleo* que orienta sua atividade. A *crença* implícita de que "o ângulo relevante para a construção de polígonos é o ângulo interno" é questionada e substituída por uma nova estipulação local: "para construir um polígono no Scratch, é preciso considerar o ângulo suplementar ao ângulo interno".

Essa mudança no *núcleo* da atividade sugere que Ana internalizou a lógica específica do Scratch para a construção de polígonos. A compreensão de que o programa utiliza o ângulo externo como referência para o comando `[gire]` leva Ana a adotar o ângulo suplementar como ferramenta para a construção de polígonos. Essa nova prática se torna legítima no contexto da programação, influenciando diretamente como Ana produz *significado* e resolve problemas nesse ambiente.

Por conseguinte, o *campo semântico* também se modifica, para acomodar esse nova modo de produção de *significados*. Conceitos relacionados tanto à matemática quanto à programação no Scratch – como ângulo externo, ângulo suplementar e o comando `[gire]` – passam a ser articulados com noções geométricas, como ângulo interno e propriedades dos polígonos.

Na sequência, a Figura 4.5 apresenta o resultado obtido por Ana após os ajustes realizados, correspondente à Figura 5 mencionada por ela.

Figura 4.5 – Triângulo desenhado no Scratch após ajustes



Fonte: Os autores.

Após o êxito na construção do triângulo e do quadrado, Ana realiza sua última enunciação no interior da referida atividade, revelando a última *direção de interlocução* para a qual enuncia: "Para deixar meu projeto ainda mais interativo, pensei em uma maneira de permitir que o usuário escolhesse qual polígono gostaria de gerar, utilizando o bloco `[se, então]` [Figura 6]. Estou animada para ver como isso vai funcionar!" (Registro de Ana).

Essa enunciação é, *plausivelmente*, direcionada ao problema proposto pelo professor, que solicitava um programa permitindo ao usuário escolher a quantidade de lados do polígono. Dessa forma, Ana busca atender à demanda do problema e obter a validação de sua solução por meio do cumprimento das exigências formais da tarefa. Essa atitude indica uma mudança em seu processo de produção de *significados*, priorizando agora a interação com o professor e a adequação de sua solução aos critérios estabelecidos por ele.

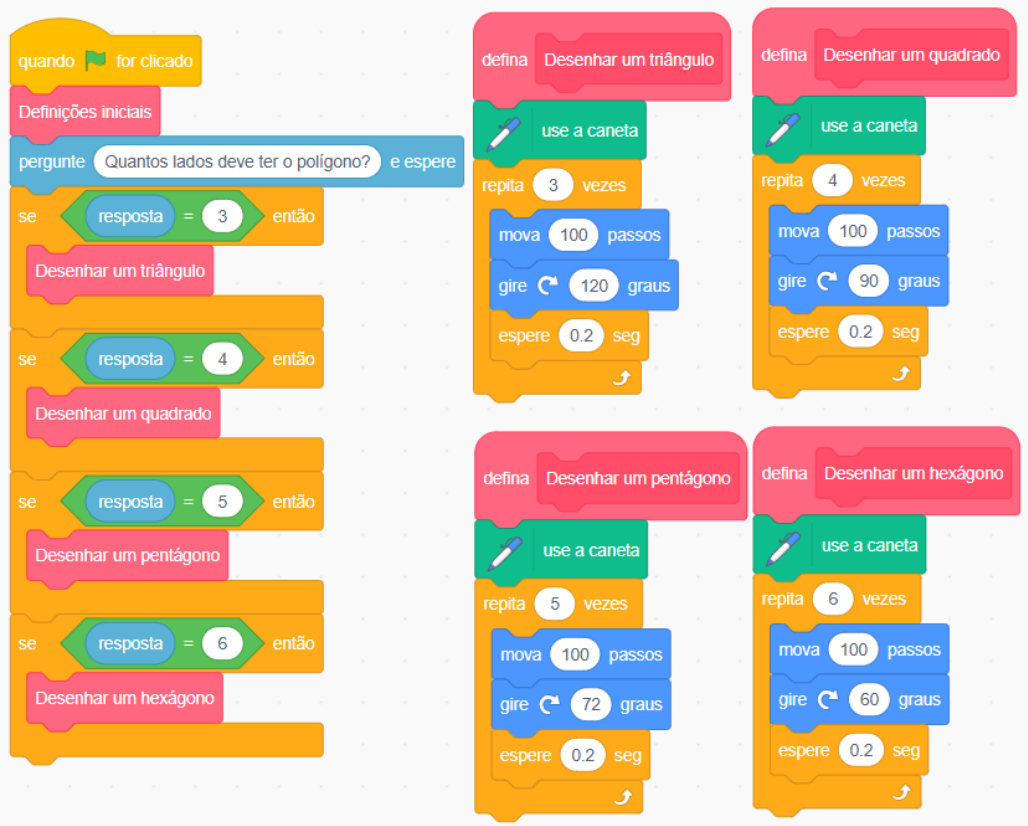
Ao enunciar nessa nova *direção*, Ana deixa de problematizar a solução apresentada pelo Scratch, não mencionando mais possíveis erros ou hipóteses de solução. Isso sugere que ela internalizou as demandas da tarefa e acredita que sua solução, ao permitir a escolha do polígono, atende plenamente ao enunciado do professor. Ana confia na correção de sua abordagem e busca apenas confirmar essa *crença* com o professor, legitimando sua solução.

O *objeto* constituído por Ana também se modifica, passando de polígonos específicos para a categoria mais abstrata de "polígono regular no Scratch". Essa abstração reflete a incorporação de novos elementos ao seu processo de design, englobando tanto as propriedades geométricas dos polígonos quanto as particularidades de sua implementação no Scratch. A transição para uma abordagem mais geral sinaliza que Ana internalizou a necessidade de criar uma solução flexível e adaptável, que atenda às diferentes escolhas de usuários.

O *núcleo* da atividade passa a ser orientado pela busca pela validação do professor e pelo cumprimento das exigências da tarefa. A "verdade absoluta" local se desloca da lógica interna do Scratch para a lógica externa da interação com o professor, visando a obtenção de aprovação. O *campo semântico* também se modifica, incorporando conceitos relacionados à interação com o usuário, como interatividade e escolha, além dos conceitos de programação e geometria que já estavam presentes. Essa modificação do *campo semântico* evidencia um aspecto relevante do processo de design: a experiência do usuário e a usabilidade do programa se tornam centrais, refletindo a atenção de Ana à funcionalidade do programa para terceiros.

Desse modo, a versão final do código é apresentada a seguir, na Figura 4.6, que corresponde à Figura 6 mencionada por Ana.

Figura 4.6 – Código final desenvolvido por Ana para desenhar qualquer polígono regular



Fonte: Os autores.

Ana reutiliza o código que havia desenvolvido para o triângulo e o quadrado, expandindo-o para outros polígonos ao duplicar e adaptar a lógica de desenho para diferentes números de lados. Esse movimento reflete, possivelmente, o processo de reconhecimento de padrões no âmbito do Pensamento Computacional. Ao reaproveitar o código de polígonos anteriores, Ana economiza tempo e esforço, demostrando como soluções pré-existentes podem ser adaptadas para novos contextos.

Conforme destaca Ferreira (2020), entender o *conhecimento* como da ordem da enunciação implica que a mudança de *direção de interlocução* resulta, necessariamente, em uma alteração no *conhecimento* mobilizado. A análise do processo de design de Ana, à luz do MCS, revela como a constituição de *objetos* e a produção de *significados* foram influenciadas pela mudança de *direção* ao longo da atividade.

Na primeira *direção de interlocução*, Ana constitui o triângulo como *objeto* central de sua atividade, com foco em suas propriedades geométricas. A *crença* de que o triângulo é “o polígono mais simples para começar” não requer *justificação* explícita, sendo tomada

como uma estipulação local nesse contexto. No entanto, ao ser confrontada com o resultado inesperado apresentado em tela, Ana percebe uma discrepância entre o triângulo idealizado e o visualizado no Scratch. Esse *estranhamento* gera uma mudança no *núcleo*, levando Ana a enunciar em uma nova *direção de interlocução*.

Na segunda *direção de interlocução*, Ana reformula o problema, passando a constituir o quadrado como *objeto* de sua atividade, utilizando-o como ferramenta de comparação e depuração. A simplicidade do quadrado, em termos de regularidade geométrica, é legitimada como uma estratégia válida para identificar o erro no código do triângulo. Nesse momento, o *objeto* do triângulo já não se limita às suas propriedades geométricas, mas é constituído dentro de um *campo semântico* que integra tantos conceitos geométricos quanto a lógica do Scratch. O triângulo, então, deixa de ser apenas um polígono com três lados e ângulos internos de 60° e se transforma em um *objeto* cuja construção está atrelada às particularidades do ambiente de programação, como a consideração do ângulo externo pelo comando `[gire]`.

Na terceira *direção de interlocução*, a constituição do *objeto* continua a se transformar. Após ajustar o código para considerar o ângulo suplementar, Ana opera com um triângulo diferente daquele inicialmente considerado, agora com propriedades que englobam a lógica de funcionamento do Scratch. Este novo triângulo, cujo ângulo interno é obtido pela consideração do ângulo externo no Scratch, constitui-se como um *objeto* que integra tanto aspectos da geometria tradicional quanto as regras da programação no Scratch. O *núcleo* da atividade, anteriormente orientado pelas propriedades geométricas dos polígonos, amplia-se para incluir a compreensão da linguagem de programação, evidenciando a dependência entre os *significados* geométricos e computacionais na constituição de *objetos*.

Finalmente, na quarta *direção de interlocução*, Ana generaliza sua solução, constituindo como *objeto* não apenas um polígono específico, mas uma categoria mais abstrata de polígono regular no Scratch. Ao propor a interatividade por meio do bloco `[se, então]`, ela expande seu foco para incluir a experiência do usuário, evidenciando uma mudança no *núcleo* da atividade, que agora integra questões de usabilidade e interação. A escolha

do polígono deixa de ser realizada exclusivamente por Ana, sendo delegada ao usuário, o que modifica o *campo semântico* da atividade para abarcar não apenas a geometria e a programação, mas também a interação e a experiência do usuário no ambiente de programação.

Nesse processo, o design revela-se como um elemento central para a produção de *significados* e constituição de *objetos*. A depuração, evidenciada nas tentativas de ajustar o código para resolver o problema do triângulo, ilustra o caráter cíclico e interativo do processo de design, no qual hipóteses são testadas e ajustadas continuamente à medida que o sujeito interage com o problema e com o *feedback* visual do código.

Além disso, o processo de decomposição é evidente na forma como Ana separa o código em diferentes blocos para lidar com polígonos específicos. Essa prática facilita a organização do pensamento e permite que Ana aborde cada parte da solução de maneira modular, mantendo a clareza e a flexibilidade necessárias para a generalização posterior. Por fim, a reutilização de código, demonstrada quando Ana aproveita as soluções desenvolvidas para o triângulo e o quadrado na criação de outros polígonos, sugere o processo de reconhecimento de padrões e demonstra a eficiência de seu design.

Desse modo, o exemplo de Ana evidencia como o processo de design influencia diretamente a constituição de *objetos* e a produção de *significados*, caracterizando um tipo de Pensamento Computacional. Ao longo de sua atividade, Ana enuncia para diferentes *direções de interlocução*, ajustando seus *objetos* e *significados* conforme sua interação com o problema, o Scratch, e as demandas impostas pela tarefa. Esse movimento cíclico entre formulação do problema, construção de soluções e depuração reflete a complexidade do processo de design no contexto do Pensamento Computacional.

Após as considerações sobre o processo de design de Ana, passa-se, agora, à análise de outra construção, agora desenvolvida por Pedro.

Suponha que Pedro tenha produzido o seguinte registro, apresentado no Quadro 4.2:

Quadro 4.2 – Registro de Pedro

Direção de interlocução	**Enunciação**
1ª direção	Comecei tentando desenhar um polígono regular com n lados, já que sabia que a fórmula para a soma dos ângulos internos é $(n-2) \times 180°$. Para encontrar a medida de cada ângulo, bastava dividir o resultado por n.
2ª direção	Implementei o código [Figura 1] e testei para $n = 3$, 4, 5 e 6. No entanto, só funcionou corretamente para $n = 4$ [Figura 2]. Observando os resultados estranhos para os outros valores, percebi que o erro estava no ângulo de giro. O Scratch usa o ângulo externo do polígono, então precisei ajustar para o ângulo suplementar do que eu havia programado inicialmente. Com isso, o problema foi resolvido [Figura 3]. Depois, testei com valores maiores de n e encontrei outro desafio: quando o número de lados aumentava, o ator não terminava de desenhar o polígono corretamente, porque batia na parede [Figura 4]. Para corrigir isso, fiz um último ajuste, diminuindo o tamanho dos segmentos à medida que a quantidade de lados aumentava [Figura 5]. Agora o programa está funcionando bem!

Fonte: Elaborado pelo autor (2025).

A primeira enunciação de Pedro, "Comecei tentando desenhar um polígono regular com n lados, já que sabia que a fórmula para a soma dos ângulos internos é $(n-2) \times 180°$. Para encontrar a medida de cada ângulo, bastava dividir o resultado por n" (Registro de Pedro), constitui o primeiro *conhecimento* produzido por ele no contexto do projeto, que pode ser expresso como:

C_1 = ("Desenhar um polígono regular com n lados no Scratch", "[pois] a soma dos ângulos internos é dada por $(n-2) \times 180°$, e a medida de cada ângulo é o resultado dessa fórmula dividido por n").

Essa *crença-afirmação* sugere, possivelmente, como Pedro formula inicialmente seu problema, utilizando como base modos de produção de significado internalizados, especificamente geométricos, para justificar sua *crença-afirmação*: a fórmula da soma dos ângulos internos de um polígono regular. Essa *justificação* parece legítima para ele naquele momento, e sua aplicação no Scratch é uma mobilização dessa legitimidade em um novo

contexto.

O *objeto* com qual Pedro opera em sua enunciação é o polígono regular com n lados, pois a enunciação inicial se concentra na determinação dos ângulos internos utilizando uma fórmula matemática. Nesse ponto, Pedro produz *significado* para o processo de programação, mobilizando legitimidades da geometria, de modo que a fórmula geométrica é diretamente implementada em seu código para desenhar o polígono.

Pedro enuncia em uma *direção de interlocução* que, ele acredita, autorizaria o uso de conceitos geométricos e matemáticos como ferramenta para resolver problemas de programação no Scratch. Ou seja, ele supõe que essa *direção de interlocução* validaria a aplicação direta de uma fórmula matemática no contexto da programação. O fato de Pedro não explicar a fórmula, apenas aplicá-la, indica que ele acredita que essa *justificação* seria considerada legítima pelo sua *direção de interlocução*.

Nesse momento, Pedro opera no *campo semântico* da geometria, tomando a fórmula dos ângulos internos do polígono como *núcleo*. Essa fórmula, dentro desse *campo semântico*, funciona como uma estipulação local, que Pedro utiliza para justificar suas *crenças-afirmações* sobre a construção do polígono. A partir desse *núcleo*, Pedro busca produzir *significados* para o código no Scratch, mobilizando legitimidades da geometria e da lógica de programação. O *campo semântico* de Pedro, portanto, envolve a mobilização de conceitos como ângulos internos e soma dos ângulos de um polígono, aplicando-os diretamente ao contexto de programação.

Desse modo, Pedro elabora a primeira versão de seu código, apresentada na Figura 4.7, correspondente à Figura 1 mencionada por ele.

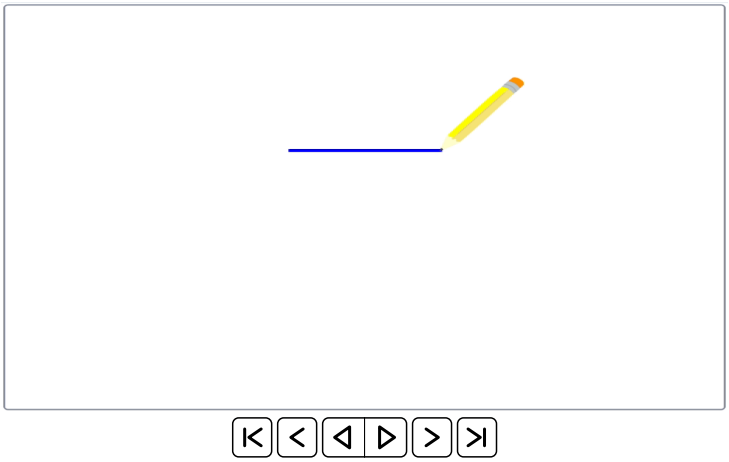
Figura 4.7 – Primeira versão do código desenvolvido por Pedro

Fonte: Os autores.

Na Figura 4.7, observa-se uma construção semelhante à de Ana, em que o código solicita ao usuário a quantidade de lados do polígono e utiliza essa resposta em dois momentos. Primeiro, dentro do laço de repetição, definindo a quantidade de lados do polígono. Por exemplo, se o usuário inserir o valor 4, o laço de repetição `[repita (4) vezes]` é acionado para desenhar um polígono de 4 lados (um quadrado), e assim por diante. A segunda aplicação da resposta ocorre no comando `[gire]`, em que o ator deve rotacionar em relação ao ângulo calculado para o polígono regular de n lados. Aqui, a variável `[resposta]` é usada no cálculo do ângulo de giro, assumindo o valor de n. Pedro aplica no comando `[gire]` a fórmula da soma dos ângulos internos, dividindo o resultado pelo número de lados do polígono.

Pedro, ao testar o código com diferentes valores de n (3, 4, 5 e 6), observa que ele funciona corretamente apenas para $n = 4$, conforme explicita em sua enunciação: "Implementei o código [Figura 1] e testei para $n = 3$, 4, 5 e 6. No entanto, só funcionou corretamente para $n = 4$ [Figura 2]" (Registro de Pedro). A partir desse teste, Pedro identifica uma discrepância entre o resultado obtido para $n = 4$ e os demais, conforme ilustra a Figura 4.8, correspondente à Figura 2 mencionada por ele.

Figura 4.8 – Resultados inesperados para valores específicos de n



Fonte: Os autores.

Pedro, operando no *campo semântico* da geometria, esperava que a fórmula fosse válida para qualquer valor de n. A diferença observada gera um *estranhamento*, levando-o a questionar as *crenças-afirmações* que guiaram a construção do código. Esse *estranhamento* desencadeia um processo de depuração, no qual Pedro busca produzir *significado* para o *resíduo de enunciação* apresentado em tela.

Em seguida, Pedro enuncia:

> Observando os resultados estranhos para os outros valores, percebi que o erro estava no ângulo de giro. O Scratch usa o ângulo externo do polígono, então precisei ajustar para o ângulo suplementar do que eu havia programado inicialmente. Com isso, o problema foi resolvido [Figura 3] (Registro de Pedro).

Dessa forma, Pedro, a partir da diferença observada, mobiliza o seguinte *conhecimento*:

C_2 = ("O Scratch usa o ângulo externo do polígono", "[pois] precisei ajustar para o ângulo suplementar do que havia programado inicialmente").

Ao buscar produzir *significado* para o *estranhamento*, Pedro reconhece que o comando `[gire]` do Scratch opera com o ângulo externo do polígono, e não com o ângulo interno, como ele assumira inicialmente. Nesse processo de depuração, Pedro abandona a *crença* de que a simples aplicação da fórmula geométrica seria suficiente, passando a considerar a lógica específica do Scratch.

Ao se deparar com a necessidade de ajustar o ângulo de giro, Pedro passa a operar

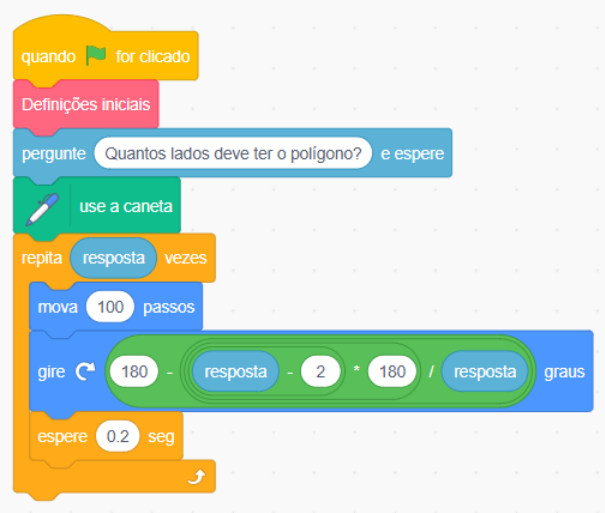
com um novo *objeto*: a relação entre o ângulo externo do polígono e o funcionamento do Scratch. Ele modifica seu *campo semântico* ao incorporar elementos da lógica do Scratch à sua compreensão geométrica.

Além disso, é possível inferir uma mudança na *direção de interlocução* para qual Pedro enuncia. Ele já não em uma *direção de interlocução* que opera apenas no *campo semântico* da geometria tradicional, mas a uma que compreende a lógica e as particularidades do Scratch. Ao justificar sua solução com base no funcionamento do Scratch, Pedro sugere uma *crença* de que a *direção de interlocução* para qual enuncia compartilha as mesmas legitimidades sobre como o software opera com ângulos.

O *núcleo* da atividade de Pedro deixa de ser a aplicação da fórmula para o cálculo dos ângulos internos de um polígono. Ele passa a operar com a lógica do ângulo externo, que, em sua perspectiva, orienta a construção de figuras geométricas no Scratch. Assim, o *campo semântico* de Pedro se expande para além dos conceitos da geometria tradicional, incorporando os comandos do Scratch e a lógica de programação utilizada pelo software para interpretar os ângulos. Pedro se encontra, portanto, em um *campo semântico* que considera tanto a geometria quanto a programação, demandando a transposição de conceitos matemáticos para o ambiente do Scratch.

Com o ajuste no valor do ângulo de giro, conforme explicita em sua enunciação, o problema é solucionado, e o polígono passa a ser desenhado corretamente. A Figura 4.9 apresenta o código ajustado por Pedro, que representa a Figura 3 mencionada por ele.

Figura 4.9 – Ajuste no ângulo de giro para correção dos polígonos



Fonte: Os autores.

Ao reconhecer e corrigir o erro no ângulo de giro, Pedro finaliza essa etapa de seu processo de design, solucionando o problema que identificado em sua primeira enunciação.

Passa-se agora à próxima enunciação de Pedro: “Depois, testei com valores maiores de n e encontrei outro desafio: quando o número de lados aumentava, o ator não terminava de desenhar o polígono corretamente, principalmente quando batia na parede [Figura 4]” (Registro de Pedro).

Essa enunciação pode ser expressa como um novo *conhecimento*:

C_3 = (“O ator não terminava de desenhar o polígono corretamente”, “[pois] quando o número de lados aumentava, ele batia na parede”).

Nesta enunciação, Pedro descreve um novo problema identificado durante a execução do código. Após ajustar o ângulo de giro, ele testa o código com valores maiores de n (número de lados) e percebe um comportamento inesperado no Scratch: o polígono gerado perde suas propriedades geométricas desejadas, já que o ator, ao se movimentar, “bate na parede” da área de desenho.

O *objeto* constituído por Pedro se modifica novamente. Anteriormente, o *objeto* eram os polígonos regulares no Scratch, com foco em suas propriedades angulares. Agora, o *objeto* passa a considerar aspectos espaciais da construção, especificamente o tamanho dos segmentos, algo que até então, *plausivelmente*, não fazia parte do horizonte de enunciações

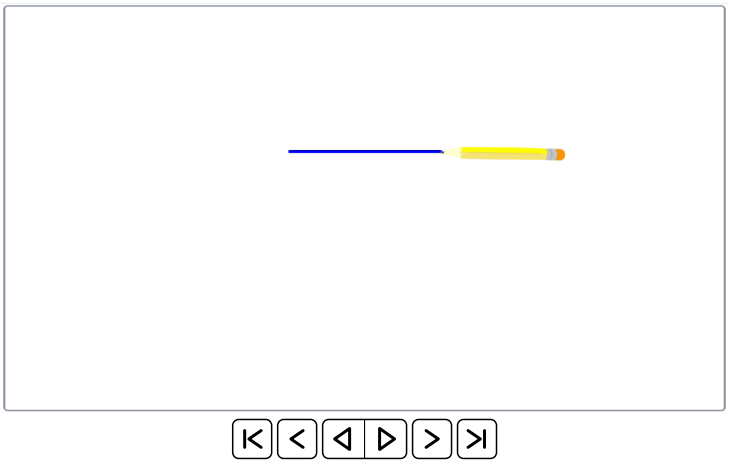
possíveis para Pedro.

O *núcleo* da atividade também sofre uma modificação, integrando o gerenciamento do espaço no ambiente do Scratch. Inicialmente, o *núcleo* era sustentado por *crenças* centradas na geometria dos polígonos e no ajuste de ângulos; agora, há uma *crença* implícita de que "o espaço disponível para o desenho deve ser considerado na construção do polígono". A partir dessa *crença*, o *núcleo* passa a incluir a necessidade de ajustar o tamanho dos segmentos a fim de evitar o contato do ator com as "paredes" do cenário, assegurando que o polígono seja desenhado corretamente.

O *campo semântico*, por sua vez, se expande. Além dos conceitos geométricos e computacionais que já estavam presentes, ele passa a incluir a noção de espaço, movimentação e limites do cenário. Nesse *campo semântico*, conceitos como o alcance do movimento do ator e o ajuste das dimensões do polígono tornam-se elementos fundamentais para que Pedro produza *significados* dentro do ambiente do Scratch.

A Figura 4.10 apresenta alguns dos resultados obtidos por Pedro ao se deparar com essa nova questão, correspondente à Figura 4 mencionada por ele.

Figura 4.10 – Resultados obtidos para valores maiores de n



Fonte: Os autores.

Pedro, então, elabora sua última enunciação no interior da referida atividade: "Para corrigir isso, fiz um último ajuste, diminuindo o tamanho dos segmentos à medida que a quantidade de lados aumentava [Figura 5]. Agora o programa está funcionando bem!". Sob a perspectiva do MCS, essa enunciação pode ser organizada como um novo

conhecimento:

C_4 = ("Diminuir o tamanho dos segmentos à medida que a quantidade de lados aumenta corrige o problema", "[pois] isso impede que o ator bata na parede e garanta que o polígono seja desenhado corretamente").

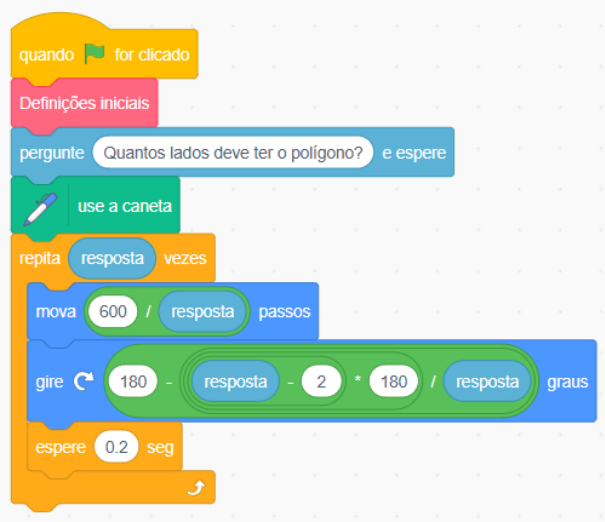
Nesta enunciação, Pedro descreve a solução final para o problema de movimentação do ator dentro do cenário do Scratch. Sua *crença-afirmação* reflete a estratégia utilizada para corrigir o erro identificado na enunciação anterior, ajustando o tamanho dos segmentos de acordo com o aumento do número de lados do polígono. A *justificação* apresentada explicita a *crença* de que diminuir o tamanho dos segmentos permitirá que o ator complete o polígono sem ultrapassar os limites da tela.

O *objeto* constituído nessa última enunciação continua a ser o polígono, mas também envolve a relação entre o tamanho dos segmentos e o número de lados. Ao ajustar o tamanho dos segmentos, Pedro reconstitui o *objeto* polígono, estabelecendo uma nova correspondência entre a quantidade de lados e o espaço ocupado pelo desenho no Scratch. Isso indica uma mudança de foco, passando da construção geométrica do polígono para o controle do espaço e da movimentação do ator dentro do ambiente de programação.

O *núcleo* da atividade de Pedro, antes centrado nas questões de ângulos e movimentação, agora se estabiliza em torno do ajuste dos parâmetros do programa para garantir que o polígono seja desenhado de forma correta e proporcional ao espaço disponível. O *campo semântico*, por sua vez, passa a abranger tanto conceitos geométricos quanto a lógica de programação no Scratch, especialmente no que se refere ao controle de movimento e ao gerenciamento de espaço. Esse *campo semântico* também passa a incluir a noção de escalabilidade do código. O ajuste no tamanho dos segmentos sugere que Pedro desenvolve uma solução flexível que se adapta ao ser desenhando na área delimitada, independentemente do valor de n, garantindo que o polígono sempre seja desenhado corretamente dentro dos limites da tela.

A Figura 4.11 apresenta o resultado final da solução implementada por Pedro, representando a Figura 5 mencionada por ele.

Figura 4.11 – Solução final com ajuste no tamanho do segmento



Fonte: Os autores.

Com esse ajuste, Pedro estabeleceu que o ator não se movimenta mais um número fixo de passos. Após a modificação, o ator se desloca $600/n$ passos; ou seja, quanto maior for o número de lados do polígono, menor será o deslocamento em cada repetição, permitindo a construção adequada dos polígonos dentro da área delimitada.

Ao construir seu projeto no Scratch, Pedro demonstra um processo de design marcado por constantes reformulações e pela produção de novos *significados*, impulsionado por sua interação com o software. Os desafios encontrados durante o desenvolvimento levam Pedro a internalizar novas legitimidades, adaptando o código para atender tanto às demandas geométricas quanto às especificidades do Scratch. Esse processo é guiado pelas *direções de interlocução* para as quais Pedro enuncia, as quais orientam suas estratégias de design e determinam os objetos que ele constitui ao longo do processo.

Inicialmente, Pedro enuncia em uma *direção de interlocução* que, em sua perspectiva, compartilha as mesmas legitimidades geométricas e matemáticas. Ao aplicar diretamente a fórmula para calcular os ângulos internos de um polígono regular, Pedro busca mobilizar uma legitimidade que considera correta e aplicável ao contexto da programação no Scratch. A *crença* de que o uso dessa fórmula seria suficiente para desenhar polígonos com n lados demonstra um processo de abstração com base na geometria. Nesse caso, o *objeto* constituído é o próprio polígono regular, visto como uma figura geomé-

trica estável, cujas propriedades podem ser transferidas diretamente para o ambiente de programação.

A complexidade do ambiente do Scratch confronta o *objeto* inicial constituído por Pedro, o polígono regular com base em uma fórmula geométrica. Essa confrontação leva Pedro a enunciar em uma nova *direção de interlocução*, buscando uma que legitime não apenas a geometria, mas também a lógica específica do Scratch.

Ao se deparar com a discrepância entre o código e o resultado esperado – polígonos não se formam corretamente para alguns valores de n –, Pedro reformula seu problema inicial. O ângulo externo, antes não considerado, passa a ser visto como essencial para a construção de polígonos no Scratch. Esse processo sugere a constituição de um novo *objeto*, que integra as propriedades geométricas com as particularidades da programação no Scratch. O *descentramento* de Pedro é crucial nesse processo: ele abandona sua *crença* inicial – na suficiência da fórmula geométrica – e adota a lógica do Scratch, compreendendo que o polígono, nesse ambiente, precisa ser ajustado de acordo com essa nova perspectiva.

O avanço no projeto apresenta a Pedro outro desafio, relacionado ao movimento do ator no ambiente de desenho. Ao perceber que o ator não completa o polígono para valores mais altos de n devido às limitações espaciais do Scratch, Pedro começa a produzir *significados* para essa restrição.

Pedro, então, ajusta o código para controlar o deslocamento do ator com base no número de lados do polígono. O *objeto* polígono, em sua concepção, modifica-se, passando a incluir a relação entre o tamanho dos segmentos e o número de lados. Assim, Pedro incorpora ao processo de design tanto as características geométricas do polígono quanto o controle de espaço no ambiente de programação.

O processo de design de Pedro se caracteriza pela prática de depuração, na qual ele busca resolver os problemas que surgem durante a execução do projeto, ajustando o código continuamente. Esse ciclo iterativo e interativo reflete um processo de produção de *significados* em constante transformação, impulsionado pelo surgimento de novas dificuldades e necessidades de adaptação. Pedro também mobiliza práticas de decomposição e abstração, abordando os componentes do projeto de forma modular e ajustando os

parâmetros do código à medida que explora as propriedades geométricas e computacionais dos polígonos.

As legitimidades matemáticas e computacionais internalizadas por Pedro – como a fórmula geométrica inicial e o conceito de ângulo suplementar – são mobilizadas durante o processo de design. Cada ajuste no código incorpora elementos do repertório geométrico ao contexto computacional, ao mesmo tempo que a lógica do Scratch é integrada ao *objeto* que Pedro constitui. O resultado é uma solução que é flexível e escalável para diferentes valores de n.

Na sequência, apresentam-se as considerações sobre o processo de design.

4.1.4 Considerações sobre o design desde a perspectiva do MCS

A análise dos casos de Ana e Pedro evidencia como o processo de design se entrelaça com os outros processos do Pensamento Computacional, destacando sua natureza cíclica, iterativa, interativa e adaptativa. Em ambos os casos, o design vai além da formulação inicial do problema, incorporando adaptações e refinamentos contínuos das soluções à medida que novos desafios e informações emergem. Esse caráter dinâmico se reflete nas ações de ajuste, reconfiguração e experimentação realizadas pelos alunos, ilustrando a flexibilidade do design em responder às demandas específicas de cada situação.

O MCS, quando adotado como lente analítica, permite evidenciar como os alunos produzem *significados* para os resultados exibidos em tela e redirecionam suas ações por meio de mudanças nas *direções de interlocução*, nos *núcleos* e nos *campos semânticos*. No caso de Ana, o *núcleo* inicial de sua atividade foi construído na *crença* de que a simplicidade geométrica do triângulo facilitaria a resolução, guiando suas primeiras ações. No entanto, ao ser confrontada com resultados inesperados, Ana redirecionou seu foco, produzindo novos *significados* e integrando elementos do funcionamento específico do Scratch. O MCS permitiu, assim, identificar como essas reformulações e redirecionamentos foram necessários para adaptar a abordagem ao ambiente de programação, resultando em uma contínua constituição e reconstituição dos *objetos* e das legitimidades envolvidas.

Contudo, é necessário problematizar essas análises para reconhecer as limitações

que permeiam as representações dos casos. Embora os exemplos de Ana e Pedro ilustrem o processo de design, é importante enfatizar que, em contextos reais de sala de aula, esse processo raramente segue um caminho estruturado ou ordenado como os apresentados. Os exemplos foram organizados com base na visão dos autores deste trabalho, o que implica uma narrativa que, por vezes, pode sugerir uma trajetória pré-determinada para o processo de design. Essa cristalização, quando apresentada de forma escrita, pode omitir a complexidade real e os desvios naturais que ocorrem no design. Em ambientes de ensino, os alunos geralmente enfrentam dificuldades imprevistas, passam por múltiplas tentativas e erros, e podem omitir etapas devido a restrições de tempo, desafios de compreensão ou influência dos objetos técnicos que estão utilizando.

Ademais, as decisões de design observadas nem sempre são fruto de um planejamento deliberado ou de uma reflexão estruturada. Muitas escolhas emergem diretamente da interação contínua com o problema e os objetos técnicos. No caso de Pedro, o ajuste do tamanho do segmento para acomodar o polígono no espaço disponível surgiu de uma necessidade prática, não planejada inicialmente. Em situações reais, soluções desse tipo poderiam ocorrer somente após múltiplas tentativas ou até pela intervenção de um professor, o que evidencia o caráter contingente e imprevisível do design.

As abordagens seguidas por Ana e Pedro representam apenas duas possibilidades para o problema proposto. Outros alunos, com diferentes repertórios, crenças, experiências e familiaridade com os objetos técnicos, poderiam formular problemas distintos e adotar estratégias completamente diferentes. Isso reflete o caráter multifacetado do design no Pensamento Computacional, no qual o processo de resolução é profundamente influenciado pelos contextos socioculturais e repertórios de cada indivíduo.

Corroborando as ideias de Ferreira (2020), os objetos técnicos, como o Scratch, desempenham um papel crucial no processo de design. As interações com esses objetos técnicos não são determinadas apenas por suas funcionalidades, mas também pelos agenciamentos sociotécnicos e as práticas culturais internalizadas pelos alunos. Assim, o design no Pensamento Computacional não ocorre em um vácuo, mas é mediado tanto pelas características dos objetos técnicos quanto por fatores contextuais e culturais que impõem

suas próprias limitações e possibilidades.

Considerando a análise detalhada e a perspectiva do MCS, é possível aprofundar a compreensão do design no Pensamento Computacional, reconhecendo seu papel não apenas como um processo de formulação e ajuste contínuo, mas como um metaprocesso[8]. Isso implica que o design atua em um nível de orientador em relação aos demais processos do Pensamento Computacional, permeando e moldando as ações e decisões do sujeito ao longo de toda a atividade de resolução de problemas.

Nessa perspectiva, o design, compreendido como o processo dinâmico de formulação, revisão e aprimoramento contínuo da abordagem para resolver um problema, é onde o sujeito inicialmente se depara com o *resíduo de enunciação* do problema e começa a produzir *significados* em interação com os objetos técnicos, o contexto e seus repertórios. Essa produção inicial de *significados*, que caracteriza a formulação do problema sob a ótica do MCS, já é, em si, um ato de design. É nesse momento que se inicia a constituição dos *objetos*, a definição das *direções de interlocução* consideradas legítimas e o estabelecimento do *núcleo* inicial que guiará a busca por uma solução.

A ideia de design como um metaprocesso no âmbito do Pensamento Computacional, conforme a lente do MCS, ganha força ao se observar como ela orienta os demais processos cognitivos, como a decomposição, a abstração, o reconhecimento de padrões e a depuração. A maneira como o sujeito formula e estrutura o problema no design abre espaço para a aplicação desses outros processos. Por exemplo, a decisão de como abordar um problema complexo (parte do design) leva à sua decomposição em partes menores. A identificação dos elementos essenciais para a solução (parte do design) implica processos de abstração. O reconhecimento de que certas estruturas se repetem (parte do design que busca eficiência) depende do reconhecimento de padrões. E, crucialmente, a depuração – o ajuste e refinamento contínuo – está intrinsecamente ligada ao processo de design, que é inerentemente cíclico e iterativo.

Assim, ao considerar o design como um metaprocesso de produção de *significados*

[8] A ideia de metaprocesso refere-se, neste trabalho, como um processo que engloba e integra outros processos.

no Pensamento Computacional, reconhece-se que é em relação a ele que todos os outros processos se manifestam e são contextualizados. É o design que impõe o objeto da necessidade da atividade (em termos da Teoria da Atividade de Leontiev (2004), que fundamenta o MCS). As mudanças nos *objetos* constituídos, nos *núcleos* estabelecidos, nos *campos semânticos* explorados e nas *direções de interlocução* adotadas ao longo da resolução do problema, conforme evidenciado nos exemplos de Ana e Pedro, são todos movimentos que ocorrem dentro e são moldados pelo metaprocesso de design. Essa perspectiva reforça o caráter dinâmico, adaptativo e multifacetado do design, posicionando-o como principal processo mental em que o Pensamento Computacional se desdobra por meio da produção de *significados*.

Na próxima seção, discorre-se sobre os processos de decomposição e produção de algoritmos à luz do MCS.

4.2 Decomposição e produção de algoritmos desde a perspectiva do MCS

Conforme discutido anteriormente, a perspectiva de Pensamento Computacional adotada neste trabalho considera que todos os processos cognitivos que o compõem estão conectados, interagindo em diferentes níveis. Todavia, ao aprofundar a análise desses processos, observou-se que a decomposição e a produção de algoritmos apresentam uma relação particularmente estreita, o que dificulta abordá-los separadamente sem incorrer em redundâncias.

A decomposição envolve a capacidade de dividir um problema complexo em partes menores e manejáveis (Espadeiro, 2021). Ao segmentar um problema em subproblemas, torna-se possível analisá-los de maneia mais aprofundada, possibilitando um foco direcionado em aspectos específicos da questão e, consequentemente, facilitando sua compreensão e resolução (Brackmann, 2017).

Além disso, segundo Brackmann (2017), a decomposição não se limita apenas à divisão do problema, mas também à identificação de elementos familiares e desconhecidos. Esse processo pode incluir a busca por problemas similares que já foram resolvidos, bem

como a adaptação de estratégias previamente conhecidas[9]. No contexto da programação, por exemplo, a decomposição possibilita a criação de sistemas complexos de maneira estruturada, por meio da utilização de funções, procedimentos e objetos computacionais.

A produção de algoritmos, por sua vez, refere-se à elaboração de um plano estruturado, uma estratégia ou um conjunto de instruções organizadas para a resolução de um problema (Brackmann, 2017). Um algoritmo descreve as etapas necessárias para atingir um objetivo e pode ser expresso em diferentes representações, como diagramas, códigos em língua materna ou linguagens de programação (Brackmann, 2017; Dantas, 2023). O processo de criação de algoritmos envolve não apenas envolve a definição de regras e procedimentos, mas também a generalização de soluções, possibilitando sua aplicação em problemas análogos.

Para que um algoritmo seja efetivo para o que foi criado, seus passos devem ser claros e bem definidos, permitindo que o processo ou o cálculo que ele realiza não produza ambiguidades; o que não se restringe à computação: desde a elaboração de uma receita culinária até a programação de um software, o desenvolvimento de algoritmos implica a organização lógica e sequencial de um conjunto de ações que podem ser executadas por um agente humano ou por um sistema computacional (Dantas, 2023).

A proximidade entre os processos de decomposição e produção de algoritmos torna desafiadora a delimitação precisa entre ambos. A fragmentação de um problema em partes menores frequentemente já implica as primeiras etapas de construção de um algoritmo. Da mesma forma, a produção de algoritmos exige, muitas vezes, a reorganização dos elementos identificados na decomposição, tornando-os interdependentes.

Diante do exposto, optou-se por abordar a decomposição e a produção de algoritmos em uma única seção, utilizando um exemplo elaborado que permite evidenciar as nuances específicas de cada processo cognitivo. Essa abordagem possibilita uma análise mais precisa das enunciações que apontam para cada um deles, sem perder de vista suas particularidades.

[9] A identificação de elementos familiares e a adaptação de estratégias previamente conhecidas já remetem ao processo de reconhecimento de padrões, que será discutido em maior detalhe em seção posterior.

Nesta seção, as noções de decomposição e produção de algoritmos serão discutidas à luz do MCS. Em vez de apresentar definições teóricas isoladas, suas conceituações serão integradas diretamente à análise da situação proposta, permitindo que suas características e diferenças sejam exploradas à medida que emergem no contexto analisado. Essa escolha possibilita uma construção teórica contextualizada, evitando repetições desnecessárias e favorecendo uma argumentação precisa dos processos cognitivos envolvidos.

O exemplo será apresentado inicialmente, seguido de uma análise das enunciações produzidas na resolução do problema, intercalando-as com as discussões. Essa estrutura visa facilitar a identificação dos aspectos que distinguem e conectam os dois processos, reforçando a abordagem interconectada do Pensamento Computacional defendida neste trabalho.

4.2.1 Um exemplo para a decomposição e produção de algoritmos

Para ilustrar as noções de decomposição e produção de algoritmos na perspectiva do MCS, considere a seguinte situação: em uma aula de geometria, o professor propõe que os alunos utilizem o GeoGebra para inscrever um quadrado em um triângulo qualquer. O enunciado do problema é: “Inscreva um quadrado em um triângulo qualquer de modo que dois vértices do quadrado estejam na base do triângulo e os outros dois vértices estejam sobre os lados inclinados do triângulo, um em cada lado. Registre os passos percorridos no processo de construção e descreva os ajustes realizados”.

A partir desse contexto, suponha que Mariana desenvolva o seguinte registro ao longo de sua atividade, conforme apresentado nos Quadros 4.3 e 4.4.

Quadro 4.3 – Registro de Mariana

Direção de interlocução	**Enunciação**
1ª direção	Para resolver o problema, decidi inicialmente explorar uma solução visual no GeoGebra, ajustando os pontos manualmente para atender às condições do problema. Desenhei o triângulo no GeoGebra a partir de três pontos (A, B e C), conectados por segmentos para formar os lados. Posicionei dois pontos móveis (D e E) sobre a base AB, que seriam os vértices inferiores do quadrado. A partir desses pontos, tracei linhas perpendiculares à base para encontrar os vértices superiores (F e G) nos lados inclinados do triângulo (AC e BC) [Figura 1]. Ajustei manualmente a posição de D e E até que as condições do quadrado fossem satisfeitas, verificando se os lados eram iguais e os ângulos internos eram retos. Embora tenha conseguido um resultado visualmente satisfatório [Figura 2], [...].

Fonte: Elaborado pelo autor (2025).

Quadro 4.4 – Registro de Mariana (Continuação)

Direção de interlocução	**Enunciação**
2ª direção	[...] Refleti que essa abordagem dependia de ajustes empíricos, o que poderia não ser aceito pelo professor por não oferecer uma solução geral ou rigorosa para o problema. Com receio de que o professor não considerasse válida minha tentativa inicial, procurei uma solução mais formal e rigorosa. Após pesquisar, descobri que o conceito de homotetia poderia ser utilizado para resolver o problema de forma geral, sem depender de ajustes manuais. Segui os passos abaixo para implementar a solução rigorosa: **1. Construção do triângulo e do quadrado inicial:** – Desenhei o triângulo ABCcom os lados AB, AC e BC, assumindo que AC era o maior lado. – Construí um quadrado com dois vértices sobre AC e outro vértice sobre AB. Para isso, posicionei um ponto móvel D sobre AC e utilizei a ferramenta *Polígono Regular* para desenhar o quadrado com base em D. O ponto G foi identificado como o quarto vértice do quadrado [Figura 3]. **2. Traçado de reta auxiliar e interseções** – Tracei a reta AG e identifiquei sua interseção P com o lado BC. – A partir de P, desenhei uma linha paralela a AC e uma perpendicular a AC, criando os pontos S e Q, respectivamente [Figura 4]. **3. Construção do quadrado final** – Por fim, tracei uma perpendicular a AC passando por S, determinando o ponto R, a interseção dessa reta com AC. – Conectei os pontos P, Q, R e S com a ferramenta *Polígono Regular* [Figura 5]. Após verificações, confirmei que o quadrilátero $PQRS$ era um quadrado. Comparando as duas construções, percebi que minha primeira abordagem, embora válida para situações específicas, não era geral e dependia de ajustes manuais. A solução rigorosa com homotetia, por outro lado, oferecia um método aplicável a qualquer triângulo, independentemente de sua forma ou proporções [Figura 6].

Fonte: Elaborado pelo autor (2025).

Ao longo de seu registro, Mariana enuncia para duas *direções de interlocução* distintas. Inicialmente, sua abordagem se baseia em uma solução visual, na qual ajusta manualmente os pontos no GeoGebra, sem recorrer a justificativas geométricas. Essa estratégia sugere uma *direção de interlocução* que legitima a verificação visual como critério suficiente para validar a construção geométrica.

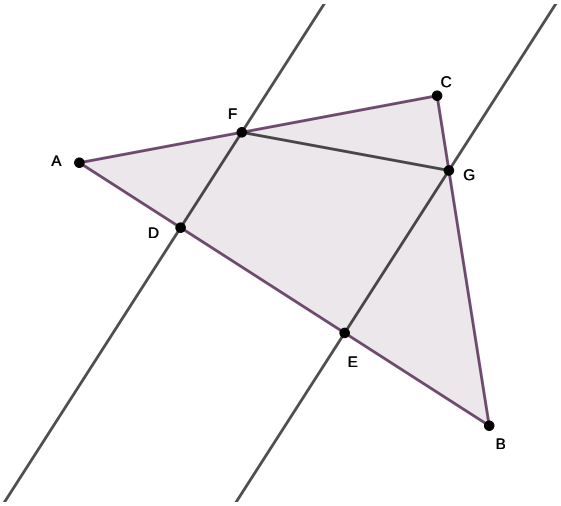
Contudo, ao refletir sobre os resultados obtidos com essa abordagem, Mariana reconsidera sua estratégia e passa a buscar uma solução geral, fundamentada nas noções de homotetia. Nesse momento, ela enuncia em uma *direção de interlocução* que legitima justificativas matemáticas rigorosas[10] e soluções generalizáveis. Essa mudança reflete uma nova relação com o problema, na qual a solução não deve apenas ser funcional em um caso específico, mas sim garantir validade para qualquer triângulo.

A análise a seguir elucida essas duas *direções de interlocução* e como cada enunciação sugere os processos de decomposição e produção de algoritmos no contexto da atividade de Mariana.

Na etapa inicial de sua resolução, Mariana adota uma estratégia visual para obter a construção geométrica desejada. O resultado parcial dessa abordagem está ilustrado na Figura 4.12 (correspondente à Figura 1 mencionada por Mariana), que apresenta a construção de um triângulo ABC, no qual o segmento AB foi utilizado como base para a definição do quadrilátero $DEGF$.

[10] Aqui, a expressão "justificativas matemáticas rigorosas" refere-se à enunciações produzidas na *direção de interlocução* da Matemática do matemático. Lins (2004) caracteriza a Matemática do matemático como internalista e simbólica. Segundo o autor, o internalismo refere-se ao fato de que, ao definir um objeto, o matemático não considera se essa definição corresponde a algo fora da própria matemática. O simbolismo, por sua vez, diz respeito à natureza dos objetos, que são conhecidos não por sua essência, mas por suas propriedades (Ferreira, 2020). Recomenda-se a leitura de Lins (2004) para uma análise mais detalhada da Matemática do matemático.

Figura 4.12 – Construção inicial de um quadrilátero inscrito em um triângulo



Fonte: Os autores.

Mariana descreve esse processo de construção em sua enunciação:

> Para resolver o problema, decidi inicialmente explorar uma solução visual no GeoGebra, ajustando os pontos manualmente para atender às condições do problema.
>
> Desenhei o triângulo no GeoGebra a partir de três pontos (A, B e C), conectados por segmentos para formar os lados. Posicionei dois pontos móveis (D e E) sobre a base AB, que seriam os vértices inferiores do quadrado. A partir desses pontos, tracei linhas perpendiculares à base para encontrar os vértices superiores (F e G) nos lados inclinados do triângulo (AC e BC) [Figura 1] (Registro de Mariana).

Essa primeira enunciação sugere que o *objeto* constituído inicialmente por Mariana é a representação visual do triângulo e do quadrilátero, construído com base em posicionar os vértices F e G por "tentativa e erro", ou seja, manipulando-os com o mouse para obter a construção de um quadrilátero $DEGF$ que se configura como um quadrado. A utilização das ferramentas do GeoGebra permite que o resultado seja manipulável, possibilitando ajustes e verificações empíricas das condições geométricas envolvidas.

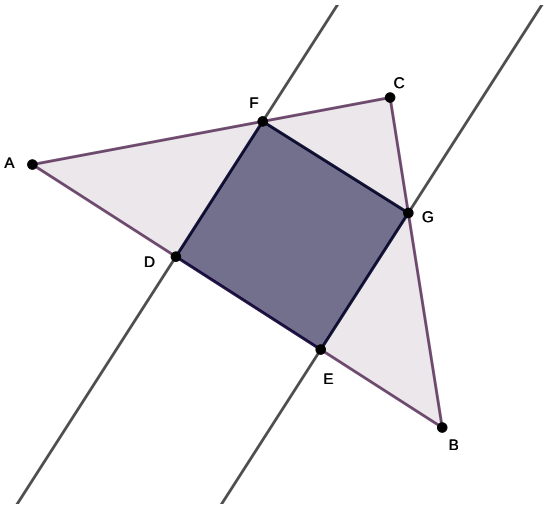
O *núcleo* que orienta essa atividade se fundamenta na ideia de "ajustar manualmente os elementos visuais para verificar condições geométricas", sustentado pela *crença* de que a manipulação gráfica pode levar a uma solução satisfatória, mesmo sem uma justificativa formal. No entanto, é relevante observar que, ao enunciar dentro dessa *direção de interlocução*, Mariana não problematiza as ferramentas do GeoGebra. Ou seja, se o software indica que uma reta é perpendicular a outra, essa informação é tomada como

"verdade absoluta", sem que haja questionamento sobre a precisão ou a fundamentação matemática dessa relação.

O *campo semântico*, nesse momento, abrange os elementos e relações geométricas representados no GeoGebra, tais como: o triângulo definido pelos vértices A, B e C; a base AB, onde estão posicionados os pontos móveis D e E; as perpendiculares traçadas a partir de D e E; e as relações geométricas implícitas, como perpendicularidade e alinhamento.

Dando continuidade à resolução, Mariana avança na direção de atender às condições estabelecidas no enunciado. Isso se reflete em sua próxima enunciação: "Ajustei manualmente a posição de D e E até que as condições do quadrado fossem satisfeitas, verificando se os lados eram iguais e os ângulos internos eram retos. Embora tenha conseguido um resultado visualmente satisfatório [Figura 2], [...]" (Registro de Caio). Nesse momento, Mariana ajusta a posição dos pontos D e E sobre a base AB, obtendo a configuração apresentada na Figura 4.13, representando a Figura 2 mencionada por ela.

Figura 4.13 – Construção inicial de um quadrado inscrito em um triângulo



Fonte: Os autores.

Observa-se que o quadrilátero $DEGF$, visualmente, pode ser considerado um quadrado, pois seus lados aparentam ter o mesmo comprimento e seus ângulos internos parecem ser retos.

Essa segunda enunciação reforça o caráter empírico da solução inicial, na qual o ajuste manual se torna a principal estratégia para alcançar um resultado que satisfaça as condições do problema. Pode-se representar essa enunciação no formato de um

conhecimento, conforme preconiza o MCS:

C_1 = ("Ajustar manualmente os pontos D e E é suficiente para satisfazer as condições geométricas de um quadrado inscrito em um triângulo", "[pois] a verificação visual no GeoGebra sugere que os lados têm o mesmo comprimento e os ângulos internos são retos").

O *objeto* constituído nesse momento é a representação visual do quadrilátero $DEGF$ no GeoGebra, que Mariana considera provisoriamente como solução do problema. O *núcleo* da atividade continua sendo a pressuposição de que ajustes manuais dos elementos gráficos permitem verificar e satisfazer as condições geométricas necessárias. O *campo semântico* é constituído pelas observações sobre a igualdade dos lados e os ângulos retos do quadrilátero.

A *direção de interlocução* permanece voltada de modo a valorizar soluções visuais e práticas, sem exigir justificativas matemáticas formais. Essa *direção* legitima uma abordagem empírica, em que o *feedback* visual do GeoGebra é considerado suficiente para validar a construção. Ademais, como consequência da *direção de interlocução* em que enuncia, Mariana não descreve o funcionamento das ferramentas do GeoGebra, apenas relata sua aplicação, sugerindo que o *interlocutor* que constitui já compreende o funcionamento interno do software.

Conforme Mariana reflete sobre sua abordagem inicial, ela explicita, em sua próxima enunciação:

> [...] Refleti que essa abordagem dependia de ajustes empíricos, o que poderia não ser aceito pelo professor por não oferecer uma solução geral ou rigorosa para o problema.
>
> Com receio de que o professor não considerasse válida minha tentativa inicial, procurei uma solução mais formal e rigorosa. Após pesquisar, descobri que o conceito de homotetia poderia ser utilizado para resolver o problema de forma geral, sem depender de ajustes manuais (Registro de Mariana).

Essa enunciação marca uma mudança na *direção de interlocução*, evidenciando a preocupação de Mariana em alinhar sua solução às expectativas do professor. Esse deslocamento indica uma reorganização no processo de elaboração da solução, modificando os critérios que orientam sua atividade.

O *conhecimento* constituído pode ser representado da seguinte forma:

C_2 = ("Uma solução geral e matematicamente rigorosa é necessária para que o problema seja aceito pelo professor", "[pois] o professor valoriza abordagens geométricas formais, que não dependam exclusivamente de ajustes manuais").

A explicitação da *justificação* sugere que Mariana produz novos *significados* para as ferramentas do GeoGebra. O foco deixa de ser a manipulação gráfica e passa a ser a validação teórica. Essa mudança marca uma reorganização do *núcleo* e do *campo semântico*, acompanhada de um deslocamento nas *justificações* que orientam a atividade, conforme será elucidado na sequência.

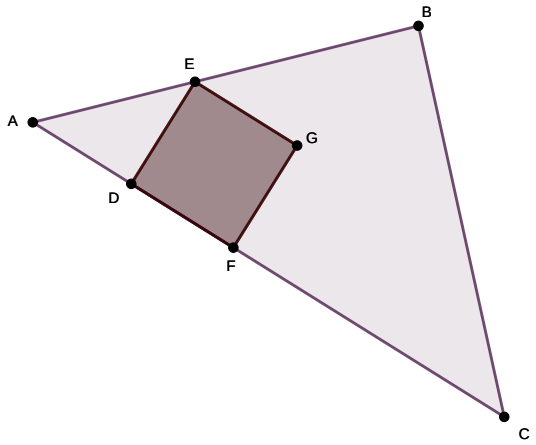
Após reconhecer as limitações[11] de sua abordagem inicial e reformular sua estratégia, Mariana explicita os passos seguidos na construção de sua nova solução:

> Segui os passos abaixo para implementar a solução rigorosa:
> **1. Construção do triângulo e do quadrado inicial:**
> – Desenhei o triângulo ABC com os lados AB, AC e BC, assumindo que AC era o maior lado;
> – Construí um quadrado com dois vértices sobre AC e outro vértice sobre AB. Para isso, posicionei um ponto móvel D sobre AC e utilizei a ferramenta *Polígono Regular* para desenhar o quadrado com base em D. O ponto G foi identificado como o quarto vértice do quadrado [Figura 3] (Registro de Mariana).

A Figura 4.14 ilustra o resultado inicial obtido por Mariana, que corresponde à Figura 3 mencionada por ela.

[11] O termo "limitações" não se refere a uma insuficiência inerente da abordagem inicial de Mariana, mas ao fato de que, diante da nova *direção de interlocução* adotada, essa estratégia deixa de ser considerada válida para a obtenção da solução.

Figura 4.14 – Construção inicial do quadrado a partir do triângulo



Fonte: Os autores.

O *objeto* constituído nesse momento é o triângulo ABC com um quadrado posicionado em relação um de seus lados. Esse *objeto* não se limita a uma representação visual, mas se constitui como uma construção matemática fundamentada na ideia de que a posição dos vértices do quadrado será determinada por relações geométricas gerais. A introdução do ponto móvel D representa uma mudança significativa em relação à tentativa anterior, pois agora Mariana concebe sua solução de modo que a construção possa ser ajustável sem comprometer sua validade matemática.

O *núcleo* dessa nova abordagem reside na ideia de "seguir um conjunto de passos matematicamente rigorosos[12] para garantir a construção correta do quadrado". Diferentemente de sua estratégia inicial, na qual a legitimidade da solução estava atrelada à verificação visual das condições do problema, a abordagem atual está fundamentada na validação geométrica e na sistematização das etapas da construção. Assim, as "verdades momentâneas" que orientam sua atividade passam a ser as propriedades geométricas dos objetos matemáticos envolvidos.

O *campo semântico* é modificado para incluir conceitos geométricos e procedimentos algorítmicos que sustentam a nova abordagem de Mariana. Além dos elementos já presentes, como o triângulo, os pontos móveis e as perpendiculares, o *campo semântico* agora

[12] Novamente, a expressão "conjunto de passos matematicamente rigorosos" refere-se à modos de produção de *significados* legítimos para a *direção de interlocução* da Matemática do matemático.

incorpora a noção de que a construção do quadrado não se baseia apenas em manipulações diretas, mas sim em argumentos matemáticos que permitem sua generalização.

Diante dessas considerações, é *plausível* inferir que a abordagem de Mariana não se baseia mais em ajustes manuais, mas em um conjunto de ações organizadas que asseguram a validade geométrica da construção. Essa mudança reflete não apenas a nova *direção de interlocução* adotada, mas também impacta diretamente como os processos de decomposição e produção de algoritmos emergem em sua atividade.

A decomposição se manifesta na maneira como Mariana organiza a construção do quadrado a partir de componentes fundamentais. Ao indicar que primeiro desenha o triângulo ABC e, em seguida, posiciona o ponto móvel D sobre AC para construir o quadrado, ela fragmenta a resolução do problema em etapas menores e manejáveis.

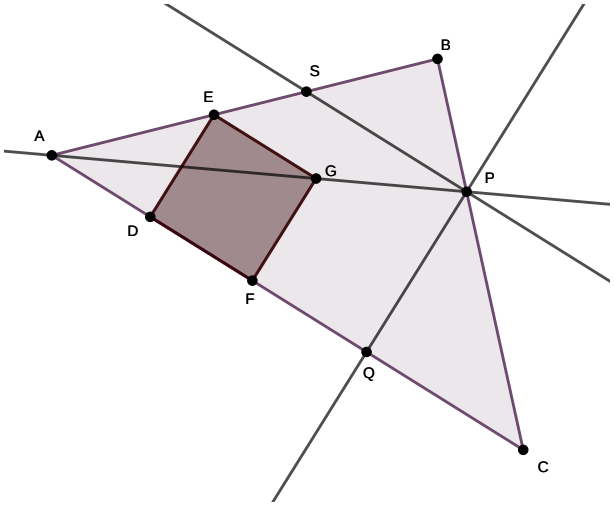
Conforme Dantas (2023), essa fragmentação permite que o sujeito concentre sua atenção na resolução de partes específicas do problema. Assim, o problema inicial -– inscrever um quadrado em um triângulo — é decomposto em subproblemas: primeiro, definir a base da construção (o triângulo) e, depois, determinar os elementos iniciais do quadrado (os vértices sobre AC e AB). Esse processo de decomposição, para além de uma divisão do problema, permite que Mariana foque em partes específicas da construção de forma independente.

A produção de algoritmos, por sua vez, se caracteriza de maneira distinta. Corroborando as ideias de Dantas (2023), um algoritmo pode ser expresso em diferentes códigos, como linguagem natural, notação matemática e programação computacional. Dessa forma, a ação de Mariana ao explicitar que "seguiu os passos abaixo para implementar a solução rigorosa" caracteriza a produção de algoritmos, pois representa a obtenção de um conjunto de passos sistematizados para a resolução de subproblemas ou problemas (Dantas, 2023).

O que diferencia essa sequência da decomposição é que cada etapa não apenas fragmenta o problema, mas estabelece relações entre as partes para garantir a construção correta do quadrado. Assim, a produção de algoritmos emerge não apenas na descrição sequencial das ações, mas no fato de que essas ações são regidas por um conjunto de regras que asseguram um resultado correto de maneira sistemática e replicável.

Após construir o triângulo e estabelecer a base para a construção do quadrado, Mariana explicita a próxima etapa para obtenção da solução: "**2. Traçado de reta auxiliar e interseções:** – Tracei a reta AG e identifiquei sua interseção P com o lado BC; – A partir de P, desenhei uma linha paralela a AC e uma perpendicular a AC, criando os pontos S e Q, respectivamente [Figura 4]" (Registro de Mariana). O resultado obtido pode ser observado na Figura 4.15, correspondente à Figura 4 mencionada por Mariana.

Figura 4.15 – Construção das retas auxiliares e interseções



Fonte: Os autores.

Essa enunciação representa um avanço significativo em relação às anteriores, pois Mariana passa a estabelecer relações geométricas explícitas entre os elementos do problema. Diferentemente das etapas anteriores, que estavam voltadas à organização dos elementos básicos da construção, aqui a introdução das retas auxiliares tem o propósito de garantir que os pontos S e Q sejam corretamente determinados de acordo com princípios geométricos formais.

No entanto, Mariana não explica como ou por que essa construção funciona, apenas descreve os passos realizados. Essa omissão indica que, apesar da transição para um pensamento sustentando por afirmações legítimas em Geometria, sua *direção de interlocução* pressupõe que não seja necessária uma justificativa detalhada do procedimento adotado.

Essa enunciação marca um ponto crucial para compreender a relação entre decomposição e produção de algoritmos. Enquanto nas etapas anteriores a decomposição esteve presente na organização dos elementos básicos do problema, aqui ela se manifesta

na identificação de novos subproblemas, como a determinação das interseções corretas para garantir a construção do quadrado.

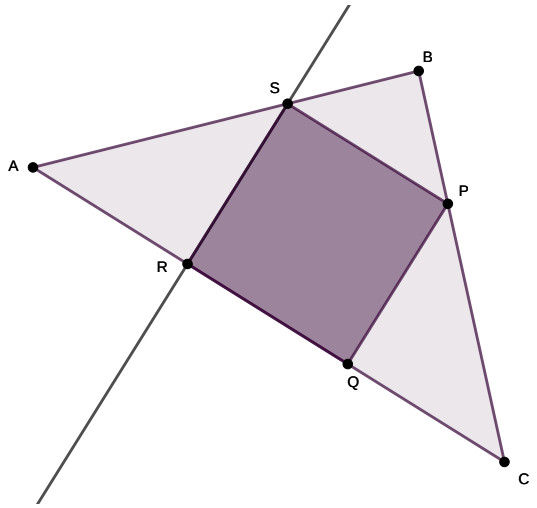
No que tange a produção de algoritmos, Mariana segue uma sequência estruturada de ações interdependentes: traça a reta AG, identifica a interseção P e, a partir desse ponto, constrói uma paralela e uma perpendicular. Cada ação não apenas complementa a anterior, mas também define os passos subsequentes, caracterizando uma estrutura algorítmica. Essa organização sequencial do pensamento garante que a solução seja replicável e generalizável, permitindo sua aplicação a qualquer triângulo, sem a necessidade de ajustes manuais.

Na sequência, Mariana enuncia:

> **3. Construção do quadrado final:**
> – Por fim, tracei uma perpendicular a AC passando por S, determinando o ponto R, a interseção dessa reta com AC;
> – Conectei os pontos P, Q, R e S com a ferramenta *Polígono Regular* [Figura 5]. Após verificações, confirmei que o quadrilátero $PQRS$ era um quadrado (Registro de Mariana).

A referida enunciação representa a finalização do processo de construção, consolidando os passos anteriores em uma estrutura geométrica rigorosa. O resultado obtido é apresentado na Figura 4.16, correspondente à Figura 5 mencionada por Mariana.

Figura 4.16 – Construção final do quadrado inscrito no triângulo



Fonte: Os autores.

A homotetia é o princípio fundamental que orienta essa construção, garantindo que a relação entre os pontos do triângulo e os vértices do quadrado seja matematicamente válida. A solução não apenas atende às condições estabelecidas pelo problema, mas também

se fundamenta em uma propriedade geométrica que assegura que qualquer transformação homotética entre os elementos manterá as proporções necessárias para a formação do quadrado. Dessa forma, a homotetia não é apenas um recurso auxiliar, mas a estrutura lógica que sustenta a solução.

Essa enunciação pode ser organizada no formato de um *conhecimento*:

C = ("O traçado de retas auxiliares e a conexão dos pontos garantem que o quadrilátero $PQRS$ seja um quadrado", "[pois] a homotetia entre os triângulos envolvidos assegura que os pontos foram corretamente posicionados, mantendo as relações de proporcionalidade e perpendicularidade").

A *crença-afirmação* expressa a segurança de Mariana em relação à validade da construção, enquanto a *justificação* se ancora na compreensão implícita de que as relações geométricas construídas garantem a correção do procedimento. Ainda que Mariana não explique explicitamente como a homotetia é empregada em sua construção, a maneira como estrutura sua construção revela que seus passos seguem um raciocínio compatível com esse princípio.

O *objeto* constituído nessa etapa é a representação visual do quadrilátero $PQRS$ no GeoGebra, que pode ser legitimamente reconhecido como um quadrado inscrito no triângulo. Diferentemente dos *objetos* constituídos ao longo do processo (como a representação visual do quadrado sem o rigor matemático[13]), esse quadrilátero representa o fechamento da solução, consolidando as condições matemáticas necessárias para garantir sua validade.

O *núcleo* reside na "certeza" de que a sequência de passos executados conduz necessariamente à construção de um quadrado. A validação de sua construção é estabelecida com base na homotetia, que garante matematicamente a correção da estrutura. Dessa forma, o princípio organizador e legitimador da solução não é mais uma verificação empírica, mas a própria homotetia, que se torna uma "verdade absoluta" dentro do processo.

O *campo semântico* dessa enunciação é expandido para incluir conceitos de validação geométrica formal, tornando-se mais abstrato e generalizável. Se antes a abordagem envolvia elementos gráficos específicos, agora a solução é sustentava por relações matemá-

[13] Isto é, sem operar com as legitimidades da Matemática do matemático.

ticas mais amplas, que garantem a aplicabilidade do método a diferentes configurações triangulares.

A decomposição se manifesta na organização das etapas finais da construção. A determinação do ponto R como interseção da perpendicular com AC é uma das últimas subetapas do processo, demonstrando como a resolução do problema ocorre de maneira estrutural e sequencial. A construção do quadrado resulta de um conjunto de decisões interligadas que viabilizam a solução.

Por sua vez, a produção de algoritmos se consolida como um procedimento lógico e reprodutível. A maneira como os pontos P, Q, R e S são determinados e conectados não se trata mais de um ajuste experimental, mas de sequência sistemática que pode ser aplicada a qualquer triângulo, garantindo a construção do quadrado de forma generalizável.

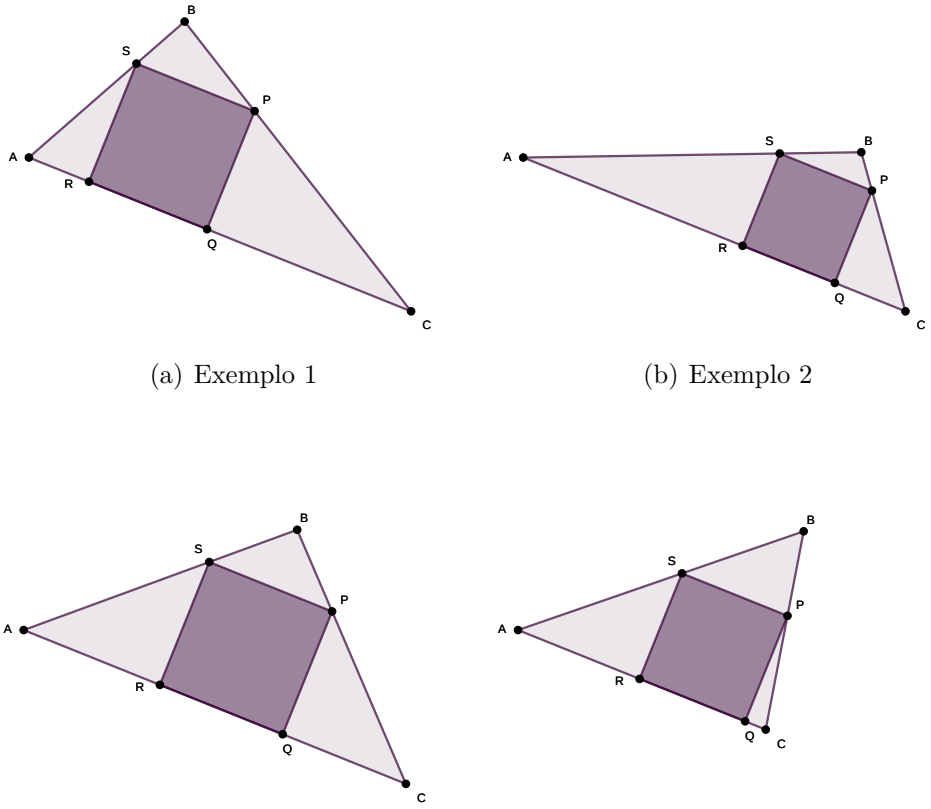
Portanto, a homotetia não apenas organiza a construção geométrica, mas também fundamenta a transição de Mariana para um pensamento abstrato e generalizável. A solução deixa de ser um processo específico para um triângulo particular e passa a ser uma construção sustentada por relações geométricas, garantindo sua aplicabilidade a qualquer caso análogo.

A homotetia, como princípio organizador, assegura que essa estrutura seja válida independentemente da configuração inicial do triângulo. Essa reflexão é evidenciada na última enunciação de Mariana:

> Comparando as duas tentativas, percebi que minha primeira abordagem, embora válida para situações específicas, não era geral e dependia de ajustes manuais. A solução rigorosa com homotetia, por outro lado, oferecia um método aplicável a qualquer triângulo, independentemente de sua forma ou proporções [Figura 6] (Registro de Mariana).

O resultado final da construção, válido para quaisquer triângulos, é apresentado na Figura 4.17, representando a Figura 5 mencionada por Mariana.

Figura 4.17 – Construção final generalizável do quadrado inscrito no triângulo



(a) Exemplo 1 (b) Exemplo 2

(c) Exemplo 3 (d) Exemplo 4

Fonte: Os autores.

A trajetória percorrida por Mariana ao longo da resolução evidencia um processo de refinamento conceitual, no qual a organização inicial do problema se transforma gradualmente em uma estrutura matematicamente fundamentada. A introdução das relações geométricas auxiliares e a adoção da homotetia não apenas tornam a solução mais precisa, mas também ampliam sua validade para diferentes configurações.

Na sequência, serão apresentadas as considerações sobre a decomposição e a produção de algoritmos na perspectiva do MCS.

4.2.2 Considerações sobre a decomposição e a produção de algoritmos desde a perspectiva do MCS

A análise do caso de Mariana evidenciou que os processos de decomposição e produção de algoritmos podem ser caracterizados de maneira precisa à luz do MCS.

Essa abordagem permite não apenas identificar enunciações que apontam para cada um desses processos, mas também compreender como eles emergem, se inter-relacionam e se transformam ao longo da atividade.

Durante a resolução do problema, foram observadas mudanças na *direção de interlocução* de Mariana, reorganizações no *núcleo* e no *campo semântico* da atividade, além da constituição de diferentes *objetos*. Essas mudanças indicam que a decomposição e a produção de algoritmos não são processos isolados, mas dinamicamente dependentes das *justificações* consideradas legítimas em cada momento da atividade.

Na perspectiva do MCS, a decomposição não é meramente um processo técnico de dividir um problema em partes menores, mas um movimento de produção de *significados* intrinsecamente ligado à atividade do sujeito. Esse processo se manifesta na estruturação do problema em subproblemas ou componentes constituintes, reorganizando o *campo semântico* da atividade e possibilitando a constituição de novos *objetos* sobre os quais se pode produzir *significado*. Tal fragmentação é orientada pelas *direções de interlocução* adotadas e pelas legitimidades que sustentam a atividade em cada momento, sendo, portanto, situada, dinâmica e dependente das condições locais da ação.

A decomposição caminha na direção de *significados* produzidos sobre *objetos* que são tomados como "dados" ou legitimados no interior da atividade, enfatizando a importância dos *resíduos de enunciação* na constituição inicial desses *objetos*. Sob essa ótica, o ponto de partida para a decomposição reside na interação do suujeito com os elementos que compõem a formulação do problema, permitindo que ele produza *significados* ao analisar as partes percebidas nesses *objetos*, bem como as relações entre elas. Assim, a decomposição é um processo que se inicia na percepção e constiuição de *objetos* no interior de uma atividade e se desenvolve à medida que esses *objetos* são fragmentos e reorganizados para tornar possível a construção de uma solução.

Além disso, a decomposição, na perspectiva do MCS, articula a análise estrutural do problema com as ações e operações concretas realizadas pelo sujeito. Não se trata apenas de uma divisão conceitual, mas também de um processo que organiza e orienta as práticas cognitivas e operatórias (ações) do sujeito, evidenciando como essas práticas

são expressões da produção de *significados*. A partir dessa articulação, a decomposição revela-se como a maneira pela qual o sujeito planeja e executa ações sobre os *objetos* constituídos, organizando o *campo semântico* e definindo os elementos que serão explorados para fins de resolução. Essas ações são moldadas pelas *direções de interlocução* vigentes, que legitimam determinadas estratégias e estipulações em detrimento de outras.

Em suma, a decomposição, nos termos do MCS, pode ser definida como um processo situado de produção de *significados* que se inicia com a análise do *objeto* constituído na referida atividade, se desdobra na estruturação do problema em partes manejáveis e se concretiza na organização das ações e operações necessárias para lidar com essas partes. Trata-se de um processo indissociável da constituição de *objetos* e da reorganização do *campo semântico*, orientado por *direções de interlocução* e legitimidades que variam ao longo da atividade, conforme evidenciado na trajetória de Mariana.

A produção de algoritmos, por sua vez, emerge quando uma sequência de passos estruturados é organizada para garantir a solução do problema. Diferente da decomposição, que fragmenta o problema, a produção de algoritmos está relacionada à estabilização de um conjunto de regras que guiam a resolução. Essa estabilização ocorre quando um *núcleo* é estabelecido e passa a ser tomado como referência para a organização das etapas da construção. No caso de Mariana, esse processo se torna evidente quando ela abandona os ajustes empíricos e passa a seguir uma sequência estruturada de construções geométricas, validando sua solução por meio da homotetia.

Entretanto, é preciso problematizar se, de fato, na primeira tentativa de resolução Mariana chegou a produzir um algoritmo, ou se sua abordagem inicial se restringiu à manipulação empírica dos elementos geométricos. A ausência de um conjunto estruturado de regras na primeira tentativa sugere que, naquele momento, sua estratégia não configurava uma produção de algoritmos no sentido estrito, mas sim um processo de exploração visual, no qual ajustes manuais eram realizados para tentar satisfazer as condições do problema. Apenas quando Mariana recorre à homotetia, sua solução passa a apresentar características mais próximas da produção de algoritmos, pois nesse momento sua abordagem se estrutura como uma sequência de passos interdependentes, cuja execução leva a um resultado

replicável e generalizável.

Essa distinção reforça a importância da *direção de interlocução* na produção de *significados* no interior da atividade. Enquanto na primeira tentativa Mariana enunciava para uma *direção* que legitimava ajustes empíricos como solução válida, na segunda tentativa, ao internalizar que uma abordagem geométrica rigorosa era necessária, reorganizou seu *campo semântico* e estabeleceu um *núcleo* que possibilitou a produção de um algoritmo. Isso evidencia que a produção de algoritmos não pode ser vista como um processo isolado, mas como algo que emerge na relação entre o sujeito e as *justificações* que ele considera legítimas em determinada atividade.

Cabe ainda destacar que, no caso analisado, a sequência de ações de Mariana se apresenta de maneira estruturada e explícita, pois esse exemplo foi construído intencionalmente para elucidar os processos de decomposição e produção de algoritmos. No entanto, em situações reais de ensino e aprendizagem, a produção de algoritmos pode se manifestar de forma fragmentada, por meio de tentativas parciais, reestruturações sucessivas e momentos de hesitação, que nem sempre evidenciam, de maneira imediata, uma sequência de passos bem definida.

Assim, ao analisar a produção de algoritmos em atividades, é fundamental considerar que sua emergência nem sempre ocorre de forma linear, e que as enunciações dos sujeitos precisam ser observadas à luz das *justificações* que orientam suas ações. No caso de Mariana, a estruturação da solução com base na homotetia permitiu que sua abordagem se consolidasse como um algoritmo, mas essa clareza pode não ser observada em outras situações, especialmente quando os sujeitos ainda estão em processo de reorganização das estratégias de resolução.

Dessa forma, embora distintos, os processos de decomposição e produção de algoritmos interagem continuamente. A decomposição permite organizar o problema em partes manejáveis, criando uma base para que a produção de algoritmos possa emergir. Por outro lado, a produção de algoritmos reorganiza o *campo semântico*, estabilizando relações entre os *objetos* constituídos e criando um procedimento replicável para a solução. No caso de Mariana, a decomposição inicial possibilitou que elementos geométricos fossem

isolados e analisados separadamente, mas foi apenas com a produção de algoritmos que sua solução se tornou sistemática e generalizável.

O MCS oferece uma contribuição essencial para essa análise, pois evidencia que a decomposição e a produção de algoritmos não são apenas habilidades técnicas ou operatórias, mas processos que envolvem a produção de *significados* dentro de uma atividade. Essa perspectiva permite compreender como diferentes *direções de interlocução*, *núcleos* e *campos semânticos* influenciam a maneira como esses processos emergem. No caso de Mariana, sua mudança de abordagem não ocorreu apenas porque ela empregou outro método, mas porque sua *direção de interlocução* se deslocou, levando-a a legitimar construções baseadas em fundamentação geométrica rigorosa, em vez de ajustes empíricos.

Ao analisar a decomposição e a produção de algoritmos nos termos do MCS, é possível identificar os *objetos* nas enunciações dos sujeitos que permitem inferir como esses processos se manifestam em diferentes atividades. Assim, o MCS não apenas descreve o que pode estar acontecendo, mas fornece lentes para analisar e compreender uma atividade em contextos educacionais diversos.

Na sequência, discorre-se sobre os processos de abstração e reconhecimento de padrões na perspectiva do MCS.

4.3 Abstração e reconhecimento de padrões desde a perspectiva do MCS

A abstração e o reconhecimento de padrões são processos centrais do Pensamento Computacional, conforme apontam diversos autores (Wing, 2008; Brennan; Resnick, 2012; Brackmann, 2017; Dantas, 2023). Esses processos possibilitam que os sujeitos organizem e simplifiquem problemas complexos, além de identificarem regularidades que otimizam suas estratégias de resolução. No entanto, assim como a decomposição e a produção de algoritmos, os processos de abstração e reconhecimento de padrões não ocorrem de maneira isolada.

Em diversas situações, conforme exposto na sequência, esses processos interagem diretamente, uma vez que a identificação de estruturas recorrentes pode orientar a construção de representações simplificadas. Dada essa relação de interdependência, esta seção

discorre sobre ambos os processos de forma integrada, analisando suas manifestações em uma atividade específica e examinando-os sob a perspectiva do MCS.

No âmbito do Pensamento Computacional, abstração refere-se à capacidade de selecionar e organizar informações relevantes para a resolução de um problema, omitindo informações, passos ou processos que não influenciam diretamente a solução (Wing, 2008; Brackmann, 2017). Wing (2008) argumenta que a abstração em computação possui particularidades que a diferenciam de outras formas de abstração, como as matemáticas ou físicas. De acordo com a autora, isso ocorre porque as abstrações computacionais nem sempre apresentam propriedades algébricas bem definidas e, ao mesmo tempo, precisam considerar as restrições do mundo físico. Assim, a escolha da abstração adequada desempenha um papel central na resolução de problemas computacionais.

A perspectiva de Brackmann (2017) sobre abstração converge com essa visão ao enfatizar que esse processo envolve a filtragem de informações, permitindo ao sujeito concentrar-se nos aspectos essenciais do problema. Como exemplo, o autor menciona a representação de mapas de metrô, nos quais detalhes geográficos irrelevantes são omitidos, destacando-se as informações necessárias para a navegação eficiente do usuário. No contexto da computação, a abstração ocorre na formulação de algoritmos, na organização de dados e na modelagem de sistemas complexos. Em consonância, Dantas (2023) destaca que a abstração consiste em um processo mental no qual o sujeito foca nos elementos essenciais e suficientes para a resolução de um problema, desconsiderando variáveis irrelevantes.

O reconhecimento de padrões, por sua vez, permite a identificação de regularidades em diferentes contextos, otimizando o processo de resolução de problemas. Segundo Brackmann (2017), esse processo frequentemente ocorre após a decomposição, pois, ao dividir um problema em partes menores, torna-se possível identificar semelhanças entre suas partes ou entre problemas previamente resolvidos. Essa habilidade possibilita a reutilização de estratégias e soluções bem-sucedidas, tornando o processo eficiente. No contexto da programação, Brennan e Resnick (2012) ressaltam que estruturas lógicas como *loops* exemplificam a aplicação do reconhecimento de padrões, permitindo que um mesmo conjunto de instruções seja repetido várias vezes sem a necessidade de reescrevê-lo.

Dantas (2023) reforça essa perspectiva ao afirmar que o reconhecimento de padrões permite ao sujeito perceber elementos recorrentes dentro de um problema, seja na identificação de estruturas familiares ou na adaptação de soluções anteriores a um novo contexto. Brennan e Resnick (2012) destacam que esse processo está diretamente ligado à prática de reutilização, favorecendo a construção de soluções mais ágeis e eficazes.

A abstração e o reconhecimento de padrões não podem ser analisados de forma dissociada do contexto em que emergem. Conforme Ferreira (2020), os objetos técnicos não são meramente ferramentas para a resolução de problemas, mas elementos que transformam o próprio problema, influenciando a forma como o sujeito formula e o resolve. Nesse sentido, a escolha de um objeto técnico, como o GeoGebra ou o Scratch, implica diferentes formas de abstração e estruturação da solução, uma vez que cada software apresenta potencialidades específicas que influenciam a produção de *significados* ao longo da atividade.

Além disso, a forma como um sujeito interage com um objeto técnico não é universal. Mesmo utilizando a mesma ferramenta e com os mesmos recursos disponíveis, sujeitos distintos podem mobilizar diferentes estratégias e conceber abstrações e padrões distintos para resolver um mesmo problema. Isso ocorre porque a produção de *significados* não ocorre exclusivamente a partir das propriedades objetivas do software, mas também das experiências, das legitimidades e das *direções de interlocução* que orientam a atividade de cada sujeito. Assim, abstração e reconhecimento de padrões não são processos fixos, mas situados, podendo variar conforme o contexto e a forma como cada sujeito interage com o objeto técnico.

Dessa forma, esta seção busca não apenas caracterizar abstração e reconhecimento de padrões de acordo com as noções do MCS, mas também discutir como um mesmo problema – neste caso, a construção de uma animação – pode exigir diferentes abstrações dependendo do objeto técnico considerado. A análise será conduzida com base no desenvolvimento de uma animação tanto no Scratch quanto no GeoGebra. A resolução do problema em cada software permitirá evidenciar que, apesar do objetivo final ser o mesmo, as estratégias de abstração e reconhecimento de padrões adotadas variam significativamente.

Inicialmente, apresenta-se o referido exemplo, seguido da análise das enunciações

produzidas na resolução do problema. Dessa maneira, busca-se evidenciar como abstração e reconhecimento de padrões emergem de forma situada, influenciados não apenas pela estrutura do problema, mas também pelas interações do sujeito com o objeto técnico e pelas *direções de interlocução* que orientam sua atividade.

4.3.1 Um exemplo para a abstração e o reconhecimento de padrões

Para elucidar os processos de abstração e reconhecimento de padrões no MCS, considere a seguinte situação: em uma determinada aula, o professor propõe que os alunos utilizem o GeoGebra e o Scratch para construir uma animação. O problema proposto é: "Utilizando as figuras em anexo, criem uma animação tanto no Scratch quanto no GeoGebra. A animação deve representar um personagem em movimento, deslocando-se da esquerda para a direita da tela enquanto as imagens alternam para simular fluidez. Após concluírem a tarefa, registrem o processo seguido em cada software e comparem as estratégias utilizadas".

A partir desse contexto, suponha que Caio tenha elaborado o seguinte registro ao longo de sua atividade, conforme apresentado nos Quadros 4.5, 4.6 e 4.7.

Quadro 4.5 – Registro de Caio

Direção de interlocução	**Enunciação**
1ª direção	Iniciei a implementação da animação pelo Scratch, pois já estava familiarizado com sua interface e os blocos de programação disponíveis. O primeiro passo foi fazer o upload das imagens fornecidas pelo professor, inserindo-as como fantasias de um mesmo ator. Em seguida, posicionei o ator na extremidade esquerda do palco, pois ele deveria deslocar-se para a direita ao longo da animação [Figura 1]. Para criar a alternância das imagens, utilizei a estrutura `[sempre]`, dentro da qual inseri o bloco `[próxima fantasia]`, seguido de um `[espere (0.1) seg]`. Esse ajuste permitiu que as imagens fossem trocadas em intervalos regulares, criando uma transição suave entre os quadros da animação [Figura 2]. No entanto, além da troca de imagens, era necessário fazer com que o personagem se movimentasse ao longo do palco. Para isso, acrescentei um bloco `[vá para x: (-200) y: (0)]`, garantindo que o ator sempre iniciasse sua trajetória na posição correta. Dentro da mesma estrutura `[sempre]`, incluí um bloco `[mova (5) passos]`, fazendo com que o personagem avançasse continuamente para direita enquanto suas fantasias eram alternadas [Figura 3]. Um problema que identifiquei durante os testes foi que, ao atingir a borda do palco, o ator desaparecia da tela. Para evitar essa interrupção, adicionei um bloco que verificava a posição do ator e, ao atingir determinado limite da coordenada x, reiniciava sua posição inicial, garantindo um movimento cíclico e contínuo. Dessa forma, a animação no Scratch foi concluída com sucesso: o personagem deslizava suavemente da esquerda para a direta enquanto suas imagens alternavam, criando a ilusão de movimento [Figura 4]. A implementação foi relativamente simples, pois o Scratch já possui ferramentas intuitivas para controle de deslocamento e troca de imagens.

Fonte: Elaborado pelo autor (2025).

Quadro 4.6 – Registro de Caio (Continuação)

Direção de interlocução	**Enunciação**
2ª direção	Ao iniciar a implementação no GeoGebra, percebi que a abordagem utilizada no Scratch não poderia ser replicada diretamente, uma vez que o software não possui recurso nativo para alternar imagens automaticamente. Isso exigiu uma reformulação da estratégia, demandando um nível maior de abstração para reinterpretar a lógica da animação dentro das possibilidades do GeoGebra. O primeiro passo foi definir a trajetória do personagem. Para isso, criei dois pontos, A e B, com coordenadas $(0, 0)$ e $(30, 0)$, respectivamente, e construí um segmento de reta entre eles. Sobre esse segmento, posicionei um ponto C, que se deslocaria ao longo da reta durante a animação. Como os eixos cartesianos e a malha quadriculada não eram necessários para a visualização da animação, optei por ocultá-los [Figura 5]. Em seguida, importei a primeira imagem e posicionei sua extremidade inferior esquerda sobre o ponto C. O GeoGebra gerou automaticamente um ponto D correspondente à extremidade inferior direita da imagem. No entanto, percebi que, para que a imagem pudesse deslocar-se corretamente pelo segmento, era necessário garantir que ela não se deformasse ao longo do movimento. Para isso, reescrevi a definição do ponto D, vinculando sua abcissa à posição do ponto C, de modo que qualquer deslocamento de C ao longo do segmento fosse acompanhado pela imagem sem alterações da escala [Figura 6]. Repeti o mesmo processo para as outras cinco imagens, inserindo-as sobre o ponto C [Figura 7]. No entanto, ao contrário do Scratch, onde a alternância das fantasias ocorre automaticamente, no GeoGebra foi necessário estruturar uma lógica para que cada imagem fosse exibida em um momento específico do movimento. Para alcançar esse efeito, escondi todas as imagens e determinei um critério matemático para definir qual delas deveria estar visível em cada instante.

Fonte: Elaborado pelo autor (2025).

Quadro 4.7 – Registro de Caio (Continuação)

Direção de interlocução	**Enunciação**
2ª direção	A estratégia adotada consistiu na criação de uma lista $l1$ contendo as seis imagens, associadas a um novo objeto que representaria dinamicamente uma delas. Para isso, utilizei o comando `Elemento(l1,Resto(⌊ x(C)⌋,6)+1` para obter um número inteiro variando entre 1 e 6, exatamente o necessário para alternar as imagens na sequência correta [Figura 8]. Dessa forma, à medida que o ponto C se movia, a imagem exibida mudava, simulando a troca de fantasias realizada no Scratch. Para completar a animação, bastou ativar o movimento automático do ponto C, fazendo com que ele deslizasse ao longo do segmento AB. Com isso, a animação foi finalizada, e o personagem passou a mover-se suavemente da esquerda para a direita enquanto suas imagens eram alternadas, recriando o efeito obtido no Scratch [Figura 9].

Fonte: Elaborado pelo autor (2025).

É *plausível* inferir que, ao longo de seu relato, Caio enuncia para *duas direções de interlocução* distintas: uma que legitima operações e construções no Scratch e outra no GeoGebra. A análise subsequente examina os processos de abstração e reconhecimento de padrões em cada uma dessas *direções*, identificando os *conhecimento*s, *objetos*, *núcleos* e *campos semânticos* constituídos ao longo da atividade. Após, discute-se como a interação do sujeito com diferentes objetos técnicos pode conduzir a distintos processos de abstrações e reconhecimento de padrões, caracterizando-os como situados e, portanto, dependentes dos objetos técnicos utilizados.

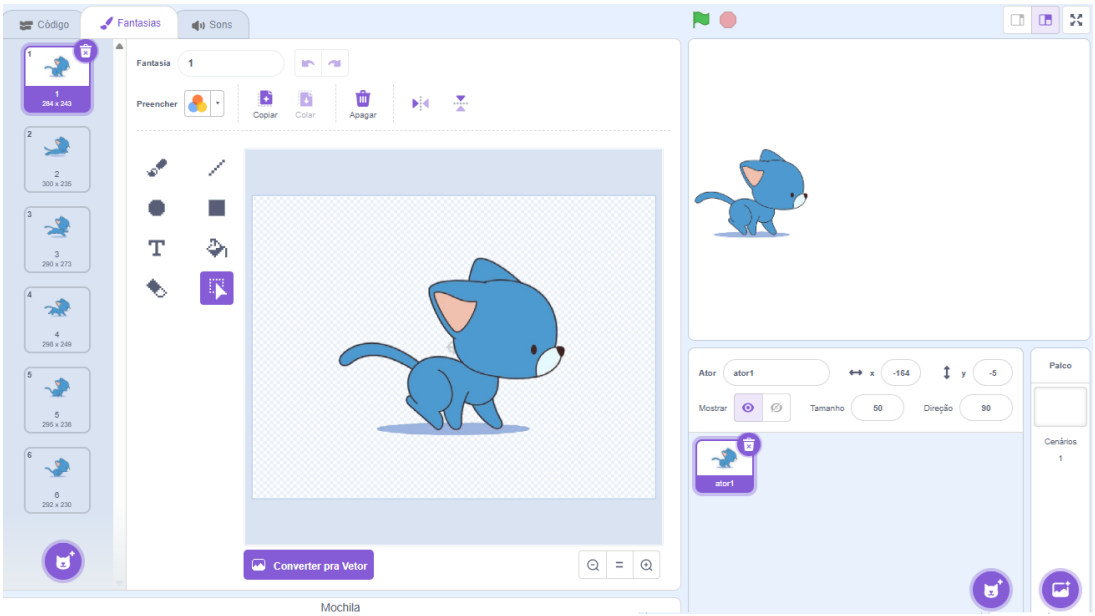
Em sua primeira enunciação, Caio afirma:

> Iniciei a implementação da animação pelo Scratch, pois já estava familiarizado com sua interface e os blocos de programação disponíveis. O primeiro passo foi fazer o upload das imagens fornecidas pelo professor, inserindo-as como fantasias de um mesmo ator. Em seguida, posicionei o ator na extremidade esquerda do palco, pois ele deveria deslocar-se para a direita ao longo da animação [Figura 1] (Registro de Caio).

O resultado dessa disposição inicial — a tela do Scratch exibindo o ator selecionado e suas fantasias — é ilustrado na Figura 4.18, correspondente à Figura 1 mencionada por

Caio.

Figura 4.18 – Tela do Scratch com o ator selecionado e suas fantasias



Fonte: Os autores.

Sob a perspectiva do MCS, essa enunciação pode ser organizada como um *conhecimento* expresso na seguinte forma:

C_1 = (“Fiz o upload das imagens e as inseri como fantasias de um mesmo ator, posicionando-o na extremidade esquerda do palco”, “pois o ator deveria deslocar-se para a direita ao longo da animação”).

Nesse contexto, a *crença-afirmação* de Caio sustenta que a implementação inicial — realizar o upload das imagens e posicionar o ator no ponto de partida — é essencial para a construção da animação. A *justificação* para essa *crença* reside na lógica operacional do software: ao posicionar o ator à esquerda, assegura-se que seu deslocamento para a direita produza o efeito visual esperado.

O *objeto* constituído nessa enunciação corresponde à configuração inicial da animação no Scratch, ou seja, ao conjunto de elementos técnicos (imagens utilizadas como fantasias e posição inicial do ator) que viabilizam a manipulação e a execução do efeito animado. Esse *objeto*, contudo, não se limita à sua representação visual, pois incorpora as especificidades do software na estruturação e organização da sequência de ações.

O *núcleo* dessa enunciação está na premissa de que a disposição inicial dos

elementos no ambiente digital é indispensável para a implementação bem-sucedida da animação. Essa "verdade local" é aceita sem necessidade de justificativa adicional no decorrer da atividade, funcionando como uma regra que orienta a resolução do problema.

O *campo semântico* mobilizado abrange noções específicas do Scratch, tais como "upload", "fantasias", "ator" e "palco". Essas noções em conjunto permitem uma produção de *significados* que orienta a construção da animação, refletindo a interação de Caio com o objeto técnico e fundamentando sua estratégia de resolução.

Além disso, é possível identificar elementos de abstração e de reconhecimento de padrões na enunciação de Caio. Ao relatar que realizou o upload das imagens e as inseriu como fantasias de um mesmo ator, ele demonstra a capacidade de selecionar apenas os elementos essenciais para representar o movimento desejado, desconsiderando detalhes irrelevantes da interface. Essa escolha caracteriza um processo de abstração, conforme discutido por Wing (2008) e Dantas (2023), pois Caio se concentra nos aspectos fundamentais da animação: a alternância das imagens e a posição inicial do ator.

Paralelamente, a forma como Caio dispõe os elementos iniciais no Scratch evidencia o reconhecimento de padrões. Ao utilizar fantasias para representar as diferentes fases da animação, ele identifica uma regularidade nas imagens fornecidas, percebendo-as como instâncias de um mesmo objeto visual. Esse reconhecimento permite estruturar um ciclo repetitivo, essencial para a construção da animação.

Na enunciação seguinte, Caio expõe:

> Para criar a alternância das imagens, utilizei a estrutura `[sempre]`, dentro da qual inseri o bloco `[próxima fantasia]`, seguido de um `[espere (0.1) seg]`. Esse ajuste permitiu que as imagens fossem trocadas em intervalos regulares, criando uma transição suave entre os quadros da animação [Figura 2] (Registro de Caio).

O resultado obtido por Caio nesse momento — a sequência de blocos que viabiliza a alternância das imagens — é apresentado na Figura 4.19, correspondente à Figura 2 mencionada por Caio.

Figura 4.19 – Sequência de blocos que controlam a alternância de fantasias na animação no Scratch

Fonte: Os autores.

Sob a perspectiva do MCS, essa enunciação pode ser organizada como um *conhecimento* expresso da seguinte forma:

C_2 = (“Utilizei a estrutura `[sempre]` com os blocos `[próxima fantasia]` e `[espere (0.1) seg]` para alternar as imagens”, “[pois] essa configuração garante uma transição suave entre os quadros da animação”).

Nesse contexto, a *crença-afirmação* de Caio é a de que a implementação de uma estrutura repetitiva, que permite a troca de fantasias em intervalos regulares, é a estratégia adequada para garantir a fluidez da animação. A *justificação* baseia-se na organização adequada dos blocos seguindo a lógica de programação do Scratch, sendo tomada uma abordagem que produz uma transição visual contínua.

O *objeto* constituído nessa enunciação é a sequência de blocos de código no Scratch que operacionaliza a alternância das imagens. Esse *objeto* não se limita à representação visual dos blocos, pois incorpora os comandos responsáveis pelo controle da temporalidade da animação, funcionando como o elemento central da dinâmica visual estabelecida.

O *núcleo* da enunciação reside na premissa de que a utilização da estrutura `[sempre]` — que possibilita a repetição contínua de ações — é uma verdade aceita no interior da atividade, dispensando justificativas adicionais no contexto do ambiente de programação. O *campo semântico*, por sua vez, se reorganiza para abranger termos e noções específicas da programação no Scratch, tais como “estrutura `[sempre]`”, “bloco `[próxima fantasia]`” e “bloco `[espere (0.1) seg]`”. Esses elementos se articulam em conjunto, permitindo uma produção de *significados* que possibilita a concepção e organização da

alternância das imagens.

Além disso, essa enunciação evidencia aspectos dos processos de abstração e reconhecimento de padrões. No que se refere à abstração, Caio demonstra a capacidade de selecionar apenas os elementos essenciais para a alternância das imagens — os blocos `[próxima fantasia]` e `[espere (0.1) seg]` —, ignorando detalhes desnecessários da interface do software.

No que tange o reconhecimento de padrões, a escolha da estrutura repetitiva e o uso dos blocos para alternar as imagens indicam que Caio identificou uma regularidade no funcionamento da animação. Ao estruturar sua solução de maneira sistemática, ele assegura a reprodução contínua desse padrão, evidenciando a relação entre esses dois processos cognitivos.

Em sua próxima enunciação, Caio relata:

> No entanto, além da troca de imagens, era necessário fazer com que o personagem se movimentasse ao longo do palco. Para isso, acrescentei um bloco `[vá para x: (-200) y: (0)]`, garantindo que o ator sempre iniciasse sua trajetória na posição correta. Dentro da mesma estrutura `[sempre]`, incluí um bloco `[mova (5) passos]`, fazendo com que o personagem avançasse continuamente para direita enquanto suas fantasias eram alternadas [Figura 3] (Registro de Caio).

O resultado obtido por Caio é apresentado na sequência, na Figura 4.20, que corresponde a Figura 3 mencionada por Caio.

Figura 4.20 – Sequência de blocos que controlam o movimento do ator na animação no Scratch

Fonte: Os autores.

Essa enunciação pode ser organizada como um *conhecimento*, conforme preconiza o MCS:

C_3 = (“Acrescentei um bloco `[vá para x: (-200) y: (0)]` para garantir que o ator iniciasse sua trajetória na posição correta e, dentro da mesma estrutura `[sempre]`, utilizei o bloco `[mova (5) passos]` para fazer com que o personagem avançasse continuamente para a direita”, “[pois] essa configuração assegura que o movimento do ator ocorra paralelamente à alternância das fantasias”).

Nesse contexto, a *crença-afirmação* de Caio é de que, para que a animação seja efetiva, é imprescindível combinar a troca de imagens com um movimento contínuo do personagem, garantindo que ele sempre inicie sua trajetória no mesmo ponto. A *justificação* baseia-se na lógica funcional do Scratch, segundo a qual o comando `[vá para x: (-200) y: (0)]` posiciona o ator de maneira que, ao executar o comando `[mova (5) passos]` repetidamente, o deslocamento ocorra de forma consistente e reproduzível.

O *objeto* constituído nessa enunciação é a sequência de blocos de código que operacionalizam o movimento do ator. Esse *objeto* integra os comandos responsáveis por definir a posição inicial e o deslocamento contínuo do personagem, conferindo dinamicidade à animação.

O *núcleo* dessa enunciação reside na premissa de que a organização e a execução de comandos repetitivos são essenciais para a produção de um movimento fluido no ambiente digital. O *campo semântico*, por sua vez, passa a abranger os termos e conceitos associados à posição e movimentação no Scratch, tais como `[vá para]` e `[mova]`. Esses elementos estruturam a estratégia de Caio ao integrar a movimentação do ator com a alternância das imagens, garantindo tanto a continuidade quanto a previsibilidade do deslocamento do personagem.

Além disso, essa enunciação evidencia aspectos relevantes dos processos de abstração e do reconhecimento de padrões. Em termos de abstração, Caio demonstra a capacidade de selecionar os comandos essenciais para implementar o movimento — especificamente, os blocos `[vá para x: (-200) y: (0)]` e `[mova (5) passos]` —, descartando outras formas de manipulação da posição do ator e concentrando-se na estrutura mínima necessária para produzir o efeito desejado. Um exemplo desse processo é a escolha de um valor numérico específico para determinar a posição inicial do ator. O número -200 não

representa uma quantidade arbitrária, mas sim um ponto de referência fixo no palco do Scratch, abstraindo o conceito de posição especial dentro do ambiente de programação e permitindo um controle mais preciso da movimentação do ator.

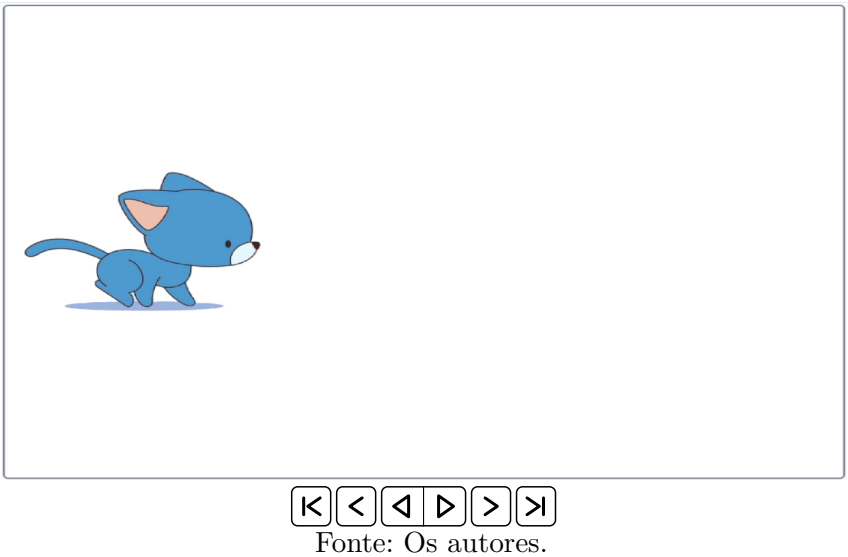
Quanto ao reconhecimento de padrões, a utilização da estrutura `[sempre]` e dos blocos responsáveis pelo deslocamento demonstra que Caio identificou a necessidade de repetir continuamente uma sequência de comandos, garantindo um movimento regular e replicável. Esse reconhecimento possibilitou uma implementação eficiente e generalizável da solução, permitindo ajustes na velocidade ou na posição inicial do ator sem comprometer a estrutura geral da animação.

Na enunciação seguinte, Caio argumenta:

> Um problema que identifiquei durante os testes foi que, ao atingir a borda do palco, o ator desaparecia da tela. Para evitar essa interrupção, adicionei um bloco que verificava a posição do ator e, ao atingir determinado limite da coordenada x, reiniciava sua posição inicial, garantindo um movimento cíclico e contínuo. Dessa forma, a animação no Scratch foi concluída com sucesso: o personagem deslizava suavemente da esquerda para a direta enquanto suas imagens alternavam, criando a ilusão de movimento [Figura 4] (Registro de Caio).

O resultado da animação obtido por Caio é apresentado a seguir, na Figura 4.21 (correspondente à Figura 4 mencionada por Caio).

Figura 4.21 – Animação no Scratch com movimento cíclico e alternância de fantasias



Fonte: Os autores.

Essa enunciação pode ser organizada como o seguinte *conhecimento*:

C_4 = ("Adicionei um bloco que verificava a posição do ator e, ao atingir determinado limite da coordenada x, reiniciava sua posição inicial", "[pois] isso garante um movimento cíclico e contínuo, impedindo que o ator desapareça da tela").

A *crença-afirmação* de Caio é que a inclusão de um mecanismo de verificação e reposicionamento do ator é necessária para assegurar a continuidade da animação, evitando que ele saia do palco e interrompa o efeito visual desejado. A *justificação* baseia-se na lógica do Scratch, que exige que a posição do ator seja controlada explicitamente para manter a fluidez da animação.

O *objeto* constituído nessa enunciação é a adaptação do código para incluir a verificação da posição do ator e seu reposicionamento automático. Desse modo, esse *objeto* representa, *plausivelmente*, a solução específica desenvolvida por Caio para garantir a continuidade ininterrupta da animação.

O *núcleo* da enunciação consiste na premissa de que, para criar um movimento cíclico e contínuo dentro do Scratch, é necessário monitorar a posição do ator e redefini-la sempre que ele atingir um limite pré-determinado. O *campo semântico* se expande para incluir conceitos associados à lógica de programação no Scratch, como "coordenada x", "verificação de posição" e "reinicialização do ator". Esse *campo semântico* se modifica à medida que Caio incorpora novas estratégias para manter a animação funcional, demonstrando um entendimento progressivo do problema e das soluções viáveis dentro do software.

Ademais, essa enunciação evidencia, mais uma vez, processos de abstração e de reconhecimento de padrões. No que tange a abstração, Caio percebe a necessidade de um controle adicional sobre a posição do ator e sintetiza essa necessidade em um critério objetivo — o limite da coordenada x -–, ignorando detalhes irrelevantes e focando apenas na regularidade essencial para o funcionamento contínuo da animação. Já no reconhecimento de padrões, a detecção do problema e a implementação de um mecanismo de reinicialização indicam que Caio identificou um comportamento recorrente (o desaparecimento do ator ao sair do palco) e formulou uma solução generalizável, aplicável a outros contextos semelhantes.

A análise das enunciações de Caio no processo de construção da animação no Scratch evidencia a relação estreita entre abstração e reconhecimento de padrões. No entanto, como apontado por Wing (2008) e Dantas (2023), a abstração não se limita à omissão de detalhes irrelevantes, mas envolve a seleção estratégia dos elementos essenciais para a solução do problema. Esse processo ocorre à medida que Caio produz *significados* para sua atividade, legitimando determinadas escolhas e constituindo *objetos* específicos ao longo da resolução.

Desde os primeiros passos, ao definir que a animação deveria ser estruturada com a alternância das imagens e o deslocamento linear do ator, Caio estabeleceu um conjunto de decisões que refletem um processo de abstração direcionado à funcionalidade da solução. Ele poderia ter considerado aspectos adicionais, como a variação da velocidade do ator, efeitos de opacidade ou interação com obstáculos. Contudo, esses elementos foram abstraídos, pois não eram essenciais para atingir o objetivo da atividade. Essa decisão revela que a abstração não é um processo arbitrário, mas uma escolha fundamentada no equilíbrio entre simplicidade e eficiência, permitindo que a atenção se concentre nos aspectos estruturais da animação.

O reconhecimento de padrões desempenhou um papel complementar nesse processo, orientando a organização da solução. Ao identificar que a alternância das imagens poderia ser sistematicamente repetida, Caio utilizou a estrutura `[sempre]` para garantir a continuidade da troca de fantasias do ator. Esse reconhecimento não apenas simplificou a implementação, mas também permitiu uma abordagem generalizável, na qual qualquer conjunto de imagens compatíveis poderia ser utilizado para criar animações semelhantes. O mesmo ocorreu com a movimentação linear do ator: ao reconhecer que seu deslocamento sempre ocorreria da esquerda para a direita, Caio estabeleceu uma regularidade que orientou a implementação do código de forma eficiente e replicável.

A análise sob a perspectiva do MCS permitiu identificar não apenas a ocorrência da abstração e do reconhecimento de padrões, mas também as legitimidades que fundamentaram as escolhas ao longo da atividade. Os *conhecimentos* produzidos por Caio, estruturados na relação entre *crença-afirmação* e *justificação*, sugerem que suas decisões

emergiram daquilo que ele considerava legítimo dentro do contexto do Scratch. Por exemplo, a decisão de usar a estrutura `[sempre]` para alternar as imagens não foi aleatória, mas justificava-se pela necessidade de garantir fluidez na animação.

A constituição dos *objetos* ao longo da atividade também foi essencial para a compreensão do processo. Desde a configuração inicial da animação até a estruturação do código, cada enunciação de Caio gerou novos *significados* e reorganizou, *plausivelmente*, o *campo semântico* da atividade. Isso evidencia que a abstração não ocorreu de forma isolada, mas como parte de um processo de interação com os elementos disponíveis no software. Abstrair, nesse contexto, significou operar sobre objetos que foram sendo estruturados ao longo da atividade, produzindo novos *significados* à medida que novas estratégias eram incorporadas.

O reconhecimento de padrões, por sua vez, pode ser analisado além da identificação de regularidades. O MCS permitiu observar que certos padrões foram incorporados ao *núcleo* da atividade, sendo aceitos como verdades locais que orientavam a resolução do problema. A ideia de que a estrutura `[sempre]` garantiria a alternância contínua de imagens, por exemplo, tornou-se um padrão estabilizado no processo de construção da animação, dispensando justificativas adicionais. Esse aspecto evidencia que o reconhecimento de padrões não foi apenas uma habilidade aplicada pontualmente, mas um processo contínuo, que emergiu da interação de Caio com o Scratch e das legitimidades que ele internalizou ao longo da atividade.

Dessa forma, a análise sob a ótica do MCS não apenas evidenciou os processos de abstração e reconhecimento de padrões na resolução de Caio, mas também permitiu compreender como esses processos emergem e se transformam ao longo da atividade. Mais do que habilidades isoladas, abstração e reconhecimento de padrões se manifestaram como construções situadas, influenciadas pelos *significados* produzidos pelo sujeito em interação com o objeto técnico e pelas legitimidades que orientaram sua atividade.

No processo de construção da animação no GeoGebra, Caio inicialmente enuncia:

> Ao iniciar a implementação no GeoGebra, percebi que a abordagem utilizada no Scratch não poderia ser replicada diretamente, uma vez que o software

> não possui recurso nativo para alternar imagens automaticamente. Isso exigiu uma reformulação da estratégia, demandando um nível maior de abstração para reinterpretar a lógica da animação dentro das possibilidades do GeoGebra (Registro de Caio).

Essa enunciação sugere um momento de reavaliação da estratégia adotada, o que pode ser organizado como um *conhecimento* na perspectiva do MCS:

C_1 = ("A abordagem utilizada no Scratch não pode ser replicada diretamente no GeoGebra", "[pois] o software não possui um recurso nativo para alternar imagens automaticamente").

Nesse contexto, a *crença-afirmação* de Caio consiste na impossibilidade de transpor diretamente a solução aplicada no Scratch para o GeoGebra. A *justificação* para essa impossibilidade reside nas diferenças entre os dois softwares: enquanto o Scratch dispõe de um mecanismo automático de alternância de fantasias, o GeoGebra não oferece esse recurso de maneira nativa, exigindo, assim, uma reformulação da estratégia adotada.

Essa enunciação marca um ponto de transição no processo de resolução e, por conseguinte, na *direção de interlocução* para qual Caio enuncia. Diante da necessidade de modificar sua abordagem, ele inicia a construção da animação no GeoGebra de maneira distinta daquela realizada no Scratch, conforme relata na sequência:

> O primeiro passo foi definir a trajetória do personagem. Para isso, criei dois pontos, A e B, com coordenadas $(0, 0)$ e $(30, 0)$, respectivamente, e construí um segmento de reta entre eles. Sobre esse segmento, posicionei um ponto C, que se deslocaria ao longo da reta durante a animação. Como os eixos cartesianos e a malha quadriculada não eram necessários para a visualização da animação, optei por ocultá-los [...] (Registro de Caio).

A disposição inicial dos elementos no GeoGebra é ilustrada a seguir, na Figura 4.22, correspondente à Figura 5 mencionada por Caio.

Figura 4.22 – Representação inicial da trajetória da animação no GeoGebra

Fonte: Os autores.

Essa enunciação pode ser organizada como um *conhecimento* na perspectiva do MCS:

C_2 = ("Criei dois pontos, A e B, e um segmento de reta entre eles, posicionando um ponto C sobre o segmento para definir a trajetória do personagem", "[pois] esse ponto se deslocaria ao longo da reta durante a animação").

A *crença-afirmação* de Caio fundamenta-se na ideia de que a criação de um segmento de reta e a definição de um ponto móvel sobre ele são essenciais para estabelecer a trajetória do personagem na animação. A *justificação* se baseia na necessidade de estruturar um movimento controlado dentro do GeoGebra, utilizando ferramentas geométricas disponíveis para simular o deslocamento desejado.

O *objeto* constituído nessa enunciação é a configuração inicial da animação no GeoGebra, composta pelos pontos A e B, pelo segmento AB e pelo ponto C posicionado sobre esse segmento. Esse *objeto*, *plausivelmente*, incorpora a propriedade de que o ponto C pertence ao segmento, permitindo sua movimentação ao longo da reta e, consequentemente, viabilizando a animação.

O *núcleo* da enunciação está na aceitação de que a movimentação do personagem pode ser representada geometricamente por um ponto móvel sobre um segmento de reta.

Essa estipulação é tomada como uma verdade local dentro da atividade e não exige justificativa adicional, pois é considerada válida para a construção da solução dentro das possibilidades do GeoGebra.

O *campo semântico* se expande para incluir conceitos relacionados à geometria dinâmica, como "pontos", "segmento de reta" e "ponto móvel". Além disso, a decisão de ocultar os eixos cartesianos e a malha quadriculada reflete um refinamento na configuração visual da animação, demonstrando uma adaptação ao ambiente do software para focar nos elementos essenciais da solução.

A enunciação de Caio evidencia um processo de abstração, pois ele simplifica a representação do movimento do personagem ao modelá-lo como um ponto móvel sobre uma reta. Em vez de tentar reproduzir diretamente a animação como no Scratch, ele abstrai a trajetória do personagem para um conceito manipulável dentro do GeoGebra. Essa abordagem demonstra a capacidade de selecionar e organizar informações relevantes para a resolução do problema, omitindo detalhes desnecessários — como os eixos e a malha quadriculada — para garantir um ambiente mais limpo e funcional.

O reconhecimento de padrões também se manifesta nesse momento, pois Caio identifica uma regularidade na necessidade de estabelecer uma trajetória linear para a animação. Ao definir um ponto móvel sobre um segmento de reta, ele reconhece um padrão de deslocamento contínuo que pode ser explorado dentro das funcionalidades do GeoGebra, estruturando sua solução com base em propriedades geométricas. Essa percepção permite que ele adapte sua estratégia à nova ferramenta, garantindo que a animação mantenha coerência com o objetivo inicial, apesar das diferenças entre os softwares.

Na enunciação seguinte, Caio afirma:

> Em seguida, importei a primeira imagem e posicionei sua extremidade inferior esquerda sobre o ponto C. O GeoGebra gerou automaticamente um ponto D correspondente à extremidade inferior direita da imagem. No entanto, percebi que, para que a imagem pudesse deslocar-se corretamente pelo segmento, era necessário garantir que ela não se deformasse ao longo do movimento. Para isso, reescrevi a definição do ponto D, vinculando sua abcissa à posição do ponto C, de modo que qualquer deslocamento de C ao longo do segmento fosse acompanhado pela imagem sem alterações da escala [Figura 6] (Registro de Caio).

A Figura 4.23, apresentada a seguir, ilustra o resultado obtido por Caio, correspondendo à Figura 6 mencionada em sua enunciação.

Figura 4.23 – Tela do GeoGebra exibindo a imagem posicionada sobre o ponto C e o ponto D ajustado para preservar a escala

Fonte: Os autores.

Sob a perspectiva do MCS, essa enunciação pode ser organizada como um *conhecimento* expresso no seguinte formato:

C_3 = (“Importei a primeira imagem e posicionei sua extremidade inferior esquerda sobre o ponto C, e o GeoGebra gerou automaticamente o ponto D”, “[pois] para que o deslocamento do ponto C acompanhasse a imagem sem que ela se deformasse, foi necessário vincular a abcissa de D à posição de C”).

Nesse contexto, a *crença-afirmação* de Caio reside na ideia de que a correta associação entre o ponto C e o ponto D é indispensável para preservar a integridade visual da imagem durante seu deslocamento. A *justificação* fundamenta-se na funcionalidade do GeoGebra, que permite redefinir a definição de um ponto para manter constantes as proporções da imagem enquanto ela se move.

O *objeto* constituído nessa enunciação corresponde a configuração inicial da imagem importada no GeoGebra, composta pela posição do ponto C — que representa o ponto de referência para o deslocamento -– e pelo ponto D, cuja definição foi ajustada para garantir que a imagem seja transportada sem alterações na escala. Esse *objeto*, portanto,

incorpora as relações matemáticas estabelecidas entre os pontos que asseguram a fidelidade da animação.

O *núcleo* da enunciação consiste na premissa de que, para evitar a deformação da imagem durante o deslocamento, é necessário ajustar a definição do ponto D de modo que sua posição permaneça vinculada à de C. O *campo semântico*, por sua vez, abrange os termos e conceitos relacionados à manipulação de imagens e pontos no GeoGebra, como "importar", "posição", "abcissa" e "escala". Esses elementos se articulam para garantir que o movimento da imagem seja acompanhado de forma precisa, sem que ocorram deformações que comprometam o efeito visual desejado.

Essa enunciação evidencia, ainda, aspectos dos processos de abstração e do reconhecimento de padrões. Em termos de abstração, Caio demonstra a capacidade de selecionar os elementos essenciais para a correta movimentação da imagem, ou seja, a necessidade de vincular a posição de D à de C, de modo a assegurar a manutenção da escala. Quanto ao reconhecimento de padrões, o ajuste realizado reflete a identificação de uma regularidade: para que a imagem se desloque sem deformação, suas proporções devem ser preservadas independentemente do movimento.

Na sequência, Caio enuncia:

> Repeti o mesmo processo para as outras cinco imagens, inserindo-as sobre o ponto C [Figura 7]. No entanto, ao contrário do Scratch, onde a alternância das fantasias ocorre automaticamente, no GeoGebra foi necessário estruturar uma lógica para que cada imagem fosse exibida em um momento específico do movimento. Para alcançar esse efeito, escondi todas as imagens e determinei um critério matemático para definir qual delas deveria estar visível em cada instante (Registro de Caio).

A Figura 4.24, apresentada a seguir, exibe o resultado obtido por Caio, correspondente à Figura 7 mencionada.

Figura 4.24 – Tela do GeoGebra exibindo a sobreposição das seis imagens

Fonte: Os autores.

Sob a perspectiva do MCS, essa enunciação pode ser organizada como um *conhecimento* expresso da seguinte forma:

C_4 = (“Repeti o mesmo processo para as outras cinco imagens, inserindo-as sobre o ponto C e defini um critério matemático para exibir cada imagem em um instante específico”, “[pois] no GeoGebra a alternância automática de imagens não está disponível, exigindo uma lógica explícita para a alternância visual”).

A *crença-afirmação* de Caio é que a replicação do procedimento de importação das imagens e a definição de um critério para alterná-las de maneira controlada são essenciais para simular o efeito de alternância da animação. A *justificação* é sustentada pelas propriedades do GeoGebra, que não dispõe de um recurso nativo para alternar imagens automaticamente, exigindo, assim, a implementação manual de uma lógica que assegure a exibição de apenas uma imagem por vez em cada instante.

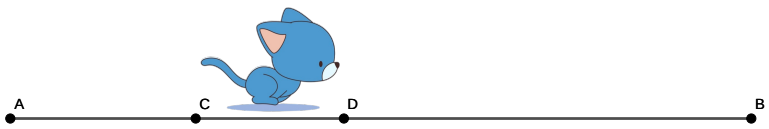
Antes de discutir o *objeto* constituído, *núcleo* e *campo semântico*, convém analisar a próxima enunciação de Caio, na qual ele detalha o critério empregado para estruturar a animação. Dessa forma, ele prossegue:

> A estratégia adotada consistiu na criação de uma lista $l1$ contendo as seis imagens, associadas a um novo objeto que representaria dinamicamente uma delas. Para isso, utilizei o comando `Elemento(l1,Resto(⌊ x(C)⌋,6)+1` para obter um número inteiro variando entre 1 e 6, exatamente o necessário para

> alternar as imagens na sequência correta [Figura 8]. Dessa forma, à medida que o ponto C se movia, a imagem exibida mudava, simulando a troca de fantasias realizada no Scratch (Registro de Caio).

A Figura 4.25, apresentada a seguir, exibe a lista $l1$ com as seis imagens e o comando utilizado para realizar a animação no GeoGebra, correspondente à Figura 8 mencionada por Caio.

Figura 4.25 – Lista de imagens e comando empregado para alternância visual no GeoGebra



Fonte: Os autores.

Essa enunciação pode ser organizada como um *conhecimento* expresso da seguinte forma:

C_5 = (“Criei uma lista $l1$ contendo as seis imagens e utilizei o comando `Elemento(l1,Resto(⌊x(C)⌋,6)+1` para alternar as imagens”, “[pois] esse comando retorna um número inteiro entre 1 e 6 que, associado ao movimento do ponto C, determina a imagem a ser exibida na sequência correta”).

Assim, a *crença-afirmação* de Caio é que a associação das imagens a um critério matemático é essencial para controlar a alternância visual de forma dinâmica, ordenada e automatizada. A *justificação* baseia-se na percepção de que o GeoGebra não dispõe de um recurso nativo para alternância automática, exigindo, portanto, a formulação de uma lógica que garanta a exibição sequencial das imagens conforme a posição de C.

O *objeto* constituído nessa enunciação corresponde ao critério definido matemati-

camente que permite a alternância das imagens no GeoGebra, ou seja, a lista $l1$ associada ao comando `Elemento(...)`, que viabiliza a seleção dinâmica de uma imagem. Esse *objeto* incorpora a lógica matemática necessária para assegurar a exibição correta dos quadros conforme o deslocamento do ponto C ao longo do segmento.

O *núcleo* da enunciação reside na premissa de que a criação de um critério matemático para alternar as imagens é indispensável para a obtenção de uma animação ordenada. O *campo semântico*, por sua vez, abrange as noções de "lista" e "resto", noções fundamentais na estruturação da animação. Dessa forma, a adaptação dos recursos matemáticos do software reflete a regularidade e continuidade necessárias para simular o efeito de troca de fantasias.

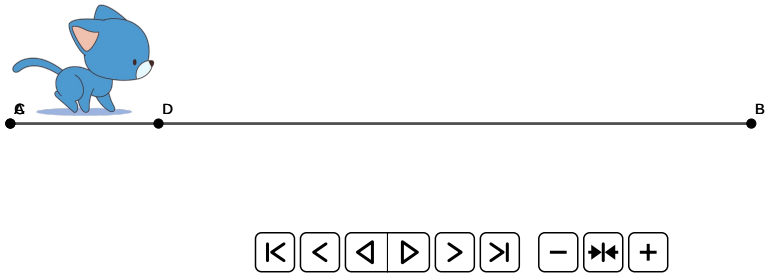
Ademais, essa enunciação evidencia outros aspectos dos processos de abstração e de reconhecimento de padrões. Em termos de abstração, Caio demonstra a capacidade de selecionar e organizar apenas os elementos essenciais – neste caso, a criação de uma lista de imagens e a definição do critério matemático – para reproduzir a alternância desejada. Esse processo abstrai a ideia de troca de imagens, traduzindo-a em um comando precisamente definido. No que se refere ao reconhecimento de padrões, a utilização do comando que retorna um número inteiro entre 1 e 6 indica que Caio identificou uma regularidade na necessidade de alternar as imagens de forma cíclica.

Em sua última enunciação na referida atividade, Caio expõe:

> Para completar a animação, bastou ativar o movimento automático do ponto C, fazendo com que ele deslizasse ao longo do segmento AB. Com isso, a animação foi finalizada, e o personagem passou a mover-se suavemente da esquerda para a direita enquanto suas imagens eram alternadas, recriando o efeito obtido no Scratch [Figura 9] (Registro de Caio).

A Figura 4.26 ilustra a animação final, apresentada como um gif, no qual o personagem se desloca da esquerda para a direita, com as imagens alternando-se continuamente. Essa figura corresponde à Figura 9 mencionada por Caio.

Figura 4.26 – Animação final no GeoGebra



Fonte: Os autores.

Essa enunciação pode ser estruturada no seguinte *conhecimento*:

C_6 = ("Ativei o movimento automático do ponto C, fazendo com que ele deslizasse ao longo do segmento AB, completando a animação", "[pois] isso permite que o personagem se mova suavemente da esquerda para a direita enquanto suas imagens são alternadas, recriando o efeito do Scratch").

A *crença-afirmação* de Caio refere-se à ativação do movimento automático do ponto C, essencial para a conclusão da animação e para garantir que o deslocamento ocorra de maneira contínua e sistemática. A *justificação* repousa na lógica funcional do GeoGebra, segundo a qual o movimento automático reproduz o efeito desejado sem a necessidade de intervenções manuais constantes.

O *objeto* constituído nessa enunciação é a configuração final da animação no GeoGebra – o mecanismo de movimento automático do ponto C ao longo do segmento AB. Esse *objeto*, *plausivelmente*, integra os comandos que asseguram a continuidade e a fluidez do deslocamento do personagem, funcionando como o elemento central que completa a solução.

O *núcleo* da enunciação reside na premissa de que a ativação do movimento automático constitui a etapa final para a resolução do problema. O *campo semântico*, por

fim, abrange os termos associados à animação no GeoGebra, como “movimento automático”, “deslizamento”, “segmento AB” e “alternância de imagens”. Esses elementos articulam-se e refletem a estratégia adotada por Caio para garantir que o personagem se desloque de forma cíclica e contínua, recriando o efeito visual desejado.

Além disso, essa enunciação evidencia aspectos dos processos de abstração e reconhecimento de padrões. No que se refere à abstração, Caio demonstra a capacidade de selecionar os elementos essenciais para a conclusão da animação – neste caso, a ativação do movimento automático do ponto C –, abstraindo a necessidade de controles manuais adicionais. Quanto ao reconhecimento de padrões, a implementação do movimento automático evidencia que Caio identificou a necessidade de que o deslocamento ocorra de forma regular e replicável, de modo a reproduzir o efeito observado no Scratch.

Na implementação da animação no GeoGebra, Caio adotou uma abordagem de abstração que difere significativamente da utilizada no Scratch. Devido à ausência de um recurso nativo para alternar imagens automaticamente, ele precisou produzir *significados* para a lógica da animação a partir de princípios matemáticos. Essa necessidade o levou a adotar outra postura diante do problema, filtrando os elementos essenciais e estruturando-os de modo a reproduzir o efeito visual desejado.

Em vez de alternar as fantasias, Caio definiu uma trajetória para o personagem por meio da criação dos pontos A e B e, sobre o segmento AB, posicionou um ponto móvel C que guiaria o deslocamento da animação. A partir dessa disposição, a abstração manifesta-se na decisão de importar individualmente cada imagem e vincular sua exibição a um critério matemático.

Ao criar uma lista $l1$ contendo as seis imagens e utilizar o comando `Elemento(l1,Resto(⌊x(C)⌋,6)+1`, Caio converte o movimento contínuo do ponto C em um ciclo que determina qual imagem será exibida a cada instante. Essa conversão exemplifica a capacidade de abstrair a noção de “alternância de imagens” — transformando-a em uma função do valor da coordenada x de C – e, assim, eliminar detalhes desnecessários, como variações não essenciais de opacidade, escala ou outras modificações visuais que poderiam complicar a solução.

Simultaneamente, o reconhecimento de padrões torna-se evidente na forma como Caio percebeu que o efeito desejado dependia de uma regularidade: o movimento do ponto C, ao longo do segmento AB, pode ser associado a um ciclo repetitivo de imagens. Ao empregar o operador Resto para limitar os valores entre 1 e 6, ele reconheceu que a alternância das imagens deve ocorrer de maneira cíclica. Essa regularidade não apenas viabilizou a implementação, mas também permitiu que a solução fosse replicada e generalizada para diferentes contextos, evidenciando a eficácia de se identificar e estruturar padrões dentro do ambiente do GeoGebra.

Na sequência, realizam-se as considerações sobre a abstração e o reconhecimento de padrões na perspectiva do MCS.

4.3.2 Considerações sobre a abstração e o reconhecimento de padrões na perspectiva do MCS

A análise do processo de construção da animação no Scratch e no GeoGebra evidencia como a abstração e o reconhecimento de padrões emergem de maneira distinta em cada objeto técnico. Apesar do objetivo final ser o mesmo, a implementação da solução exigiu diferentes estratégias em função das especificidades de cada software. Com as noções do MCS, foi possível realizar uma leitura precisa desses processos, considerando a constituição dos *objetos*, a organização dos *núcleos* e *campos semânticos* e as *direções de interlocução* que orientaram e legitimaram as decisões de Caio.

No Scratch, a abstração manifestou-se na seleção dos elementos essenciais para a animação: a alternância das fantasias do ator e seu deslocamento linear. Caio, operando com as legitimidades que ele constitui com seu uso do Scratch, utilizou recursos nativos do software para esse processo, como a estrutura `[sempre]` para repetir comandos e o bloco `[próxima fantasia]` para alternar imagens automaticamente. Assim, a abstração operada por Caio concentrou-se na definição dos passos essenciais para manter a fluidez da animação sem que fosse necessário tratar manualmente cada mudança de quadro. O reconhecimento de padrões esteve presente na percepção de que a repetição de uma sequência fixa de imagens poderia ser sistematicamente reproduzida para gerar um efeito

visual consistente. A identificação desse padrão levou Caio a estruturar sua solução de forma automatizada, garantindo a eficiência do processo.

No GeoGebra, a abstração ocorreu de forma distinta. A ausência de um recurso nativo para alternância automática de imagens exigiu uma produção de *significados* para o problema em termos matemáticos. Caio precisou abstrair a ideia de transição entre quadros e reformulá-la como uma função dependente da posição de um ponto móvel. A criação de uma lista de imagens e a utilização do operador Resto para alterná-las de acordo com a coordenada x do ponto C são exemplos dessa abstração. Nesse caso, o reconhecimento de padrões manifestou-se na percepção de que o deslocamento linear do ponto C poderia ser associado a uma sequência cíclica de imagens. Caio identificou que, ao estruturar essa relação matematicamente, poderia obter um efeito análogo àquele produzido no Scratch, mas por meio de uma abordagem completamente diferente.

Embora essas distinções práticas entre os softwares sejam relevantes, para compreender a abstração como um dos processos do Pensamento Computacional sob a ótica do MCS, é necessário ultrapassar as especificidades técnicas de cada objeto técnico e buscar uma caracterização mais profunda e epistemologicamente situada. A abstração, nesse sentido, não se limita à omissão de detalhes irrelevantes ou à criação de representações simplificadas – aspectos já tratados na literatura e ilustrados nos exemplos –, mas reside em um processo de seleção e legitimação de *objetos* e *significados*.

Na perspectiva do MCS, abstração pode ser compreendida como o processo pelo qual o sujeito, no interior de uma atividade, seleciona *objetos* e modos de produção de *significados* considerados legítimos e necessários à resolução do problema proposto. Dentre as diversas potencialidades de produção de *significados* que o sujeito dispõe em uma dada situação, a abstração é o processo que discrimina e prioriza os *significados* que, dentro do conjunto de estipulações locais (*núcleo*) e do *campo semântico* instaurado, são tidos como pertinentes e operacionais. Essa seleção não é neutra nem meramente lógica: trata-se de um ato situado de produção de *significados*, moldado pelas *direções de interlocução* que o sujeito estabelece e pelas legitimidades que internalizou ao longo de sua trajetória.

Desse modo, a diferença entre as abstrações observadas nos exemplos do Scratch e

do GeoGebra não pode ser reduzida a variações de procedimento, mas deve ser entendida como resultado de diferentes seleções de *objetos* e modos de produção de *significados*, legitimadas em cada contexto específico. O que se mantém comum entre os dois contextos é o movimento cognitivo de Caio em constituir e validar *objetos* (como "fantasia", "transição visual", "ponto móvel", "lista indexada") a partir do que considera legítimo e operacional dentro de cada objeto técnica e de seu modo de produção de *significados* para o *resíduo de enunciação* "enunciado do problema".

Portanto, a abstração, no Pensamento Computacional à luz do MCS, é um processo de validação e priorização de modos de produção de *significado*. O sujeito, em interação com o problema e os objetos técnicos, define quais *objetos* serão constituídos e quais modos de produção *significados* serão tomados como operacionais e aceitáveis dentro de seu *campo semântico*. Tal escolha é o que caracteriza, fundamentalmente, a abstração como um processo de pensamento, e não apenas como uma técnica de simplificação.

Em relação ao reconhecimento de padrões, embora tradicionalmente definido como a identificação de regularidades ou repetições em dados ou problemas, no MCS esse processo pode ganhar contornos mais específicos. Trata-se de um processo de produção de *significados* que permite ao sujeito organizar sua atividade, antecipar comportamentos e adaptar estatégias com base em regularidades percebidas como legítimas.

Sob essa perspectiva, o reconhecimento de padrões não se reduz à constatação objetiva de uma repetição, mas consiste na produção de *significado* para um *objeto* – por exemplo, "um gato animado atravessando a tela" – que possibilita ao sujeito antecipar que certas recorrências associadas a esse *objeto* tendem a se repetir. Essa antecipação está enraizada em modos de produção de *significados* prévios tomados como legítimos e em experiências anteriores que conferem estabilidade à expectativa do sujeito. Reconhecer um padrão, assim, é instituir um *objeto* com determinadas propriedades regulares e se orientar cognitivamente por essas propriedades, o que reorganiza o *campo semântico* e direciona as ações subsequentes.

Esse processo de reconhecimento e antecipação se transforma ao longo da atividade, sendo influenciado diretamente pelas interações com os objetos técnicos e pelo contexto

específico da tarefa. No caso de Caio, o reconhecimento de padrões manifestou-se de maneira diversa conforme o ambiente: no Scratch, associou a alternância de imagens à repetição de blocos `[próxima fantasia]`; no GeoGebra, vinculou a repetição visual ao deslocamento de um ponto e ao uso do operador `Resto`. Ambos os casos revelam como o reconhecimento de padrões, sob o MCS, não é uma operação fixa ou universal, mas uma construção situada, influenciada pelos objetos técnicos, pelos repertórios do sujeito e pelas *direções de interlocução* assumidas.

No cerne do reconhecimento de padrões, à luz do MCS, está a capacidade do sujeito de produzir *significados* que legitimem a antecipação de recorrências. Essa antecipação modifica os *núcleos* e *campos semânticos* da atividade e permite que o sujeito transponha soluções, reutilize estratégias ou adapte métodos a contextos semelhantes. Trata-se de um componente essencial na resolução de problemas, profundamente articulado à abstração, que permite ao sujeito trafegar pela complexidade ao focar no essencial, priorizando estruturas que podem ser reutilizadas ou replicadas com base nos modos de produção de *significado* que ele reconhece e legitima.

O MCS desempenhou um papel essencial nesse processo ao possibilitar uma análise do processo de produção de *significados* ao longo da atividade. A identificação dos *conhecimentos* permitiu compreender as *justificações* que sustentaram as decisões de Caio, evidenciando que suas escolhas não foram arbitrárias, mas legitimadas pelas *direções de interlocução* que orientavam sua atividade[14]. Os *objetos* constituídos revelaram como a interação com cada software influenciou a estruturação da solução, enquanto a análise dos *núcleos* e *campos semânticos* demonstrou que certas noções foram tomadas como verdades locais, orientando o processo de produção de *significados* no interior da atividade.

Assim, ao discutir abstração e reconhecimento de padrões na perspectiva do MCS, pode-se inferir que esses processos são inseparáveis das condições específicas da atividade e dos objetos técnicos considerados. Longe de serem habilidades fixas, eles emergem na

[14] Ao longo de seu relato, Caio não explicitou como o Scratch ou o GeoGebra funcionam nem justificou as relações matemáticas empregadas. Em vez disso, ele apresentou e aplicou diretamente suas propriedades. Isso ocorre porque, ao enunciar para uma determinada *direção de interlocução*, pressupõe que tais noções sejam compreendidas *a priori* por aqueles que compartilham dessa mesma legitimidade. Esse aspecto reforça o argumento de que as *justificações* presentes nas enunciações de um sujeito são socialmente situadas e dependem do contexto da atividade em que ocorrem.

interação entre sujeito, problema e objeto técnico, sendo moldados pelas possibilidades de cada contexto. Essa perspectiva permite evidenciar que a manifestação do Pensamento Computacional não depende apenas da capacidade individual do sujeito, mas também do ambiente em que a atividade ocorre, das legitimidades envolvidas e dos *significados* que são produzidos nesse processo.

Na sequência, discorre-se sobre o processo de depuração desde a perspectiva do MCS.

4.4 Depuração desde a perspectiva do MCS

A depuração (ou *debugging*) está presente em diversas práticas cotidianas, especialmente na interação com objetos técnicos. Por exemplo, ao tocar o ícone de um aplicativo no celular para enviar uma mensagem, o sujeito espera que o dispositivo responda conforme o esperado – ou seja, que a mensagem seja enviada ao destinatário correto. Caso o aplicativo apresente um comportamento inesperado, como não abrir ou não enviar a mensagem, o sujeito inicia uma ação de verificação e ajuste, buscando garantir que o dispositivo funcione conforme planejado. Esse movimento, característico do processo de depuração, envolve a identificação de problemas e realização de ajustes até que o resultado esperado seja alcançado.

No âmbito do Pensamento Computacional, a depuração, como defendem Espadeiro (2021), Dantas (2023) e Teixeira, Juvanelli e Dantas (2024), é um processo mental que pode se manifestar durante ou após a resolução de problemas. Esse processo envolve a busca e correção de erros por meio de testagens, verificações, refinamentos e otimizações, sendo essencial para garantir a coerência entre a solução obtida e o objetivo inicial. Ao longo da depuração, o sujeito avalia os resultados (parciais ou finais) e ajusta hipóteses e estratégias conforme necessário, mantendo a solução em alinhamento com o objetivo proposto.

No contexto da interação com os objetos técnicos[15], Teixeira, Juvanelli e Dantas

[15] Teixeira, Juvanelli e Dantas (2024) trata de Tecnologias Digitais na perspectiva de Dantas (2022), a qual brevemente apresentada anteriormente. Contudo, acredita-se que a adoção da terminologia "objetos técnicos" pode ser empregada nesta pesquisa sem alteração nas concepções teóricas envolvidas.

(2024) argumentam que a depuração adquire nuances específicas. Em programação, por exemplo, ela visa o aprimoramento e a eficiência da solução, eliminando redundâncias e ajustando comandos para que o código se torne mais otimizado e funcional, com o menor uso possível de recursos computacionais. Nesse sentido, a depuração envolve tanto o refinamento lógico quanto a adaptação do código, com o propósito de assegurar que a funcionalidade esteja em conformidade com os objetivos estabelecidos.

Ademais, Teixeira, Juvanelli e Dantas (2024) ressaltam que a depuração pode ocorrer em contextos de *interação produtiva* e *colaborativa*[16], nos quais sujeitos interagem e compartilham, ou não, *direções de interlocução* e motivos para identificar e corrigir erros em conjunto. Embora esse tipo de depuração apresente características específicas, o foco desta seção recai sobre a depuração no contexto de um sujeito interagindo individualmente com os objetos técnicos, ou seja, um processo de ajuste e refinamento realizado de forma independente, a partir das interações diretas entre o sujeito e o objeto técnico.

Para este trabalho, fundamentado nas discussões de Teixeira, Juvanelli e Dantas (2024), considera-se que os erros passíveis de depuração podem ser classificados em dois tipos principais: erros de sintaxe e erros de semântica. Cada um desses tipos será explorado sob as lentes do MCS, com o objetivo de evidenciar as diferentes dinâmicas de produção de *significados* envolvidas durante o processo de depuração.

Esta seção organiza-se da seguinte forma: inicialmente, discute-se sobre os erros de sintaxe no contexto da depuração; em seguida, examinam-se os erros de semântica e suas implicações no processo de produção de *significados*; por fim, apresentam-se as considerações gerais sobre a depuração à luz do MCS, destacando suas nuances no Pensamento Computacional.

[16] As noções de *interação produtiva* e *colaborativa* são oriundas do MCS. Suas definições detalhadas e as nuances associadas, contudo, fogem do escopo deste trabalho. Para uma compreensão aprofundada, recomenda-se a leitura de Dantas, Ferreira e Paulo (2016) e Dantas e Lins (2017).

4.4.1 Depuração de erros de sintaxe desde a perspectiva do MCS

Erros de sintaxe referem-se a desvios nas estruturas ou na ordem de elementos necessários para que um objeto técnico, como um compilador[17], editor de texto ou planilha, interprete adequadamente o que foi inserido. Esses erros incluem palavras escritas incorretamente, ausência de finalizadores de sentenças (por exemplo, ponto e vírgula, quando requeridos) ou falta de adequação a padrões específicos de um determinado objeto técnico (Teixeira; Juvanelli; Dantas, 2024). Em geral, os compiladores detectam automaticamente esses erros, avaliando a conformidade de comandos, expressões, classes, funções, parâmetros e procedimentos de acordo com a estrutura esperada.

Na perspectiva do MCS, a depuração de erros de sintaxe pode ser compreendida como um processo de produção e manutenção de *significados* que ocorre em interação com as especificidades do objeto técnico. Esse processo tem início quando o sujeito identifica uma divergência entre a estrutura que elaborou e o resultado esperado. O objeto técnico, ao sinalizar um erro, fornece *resíduos de enunciação*, que funcionam como indicadores para um ajuste em direção ao comportamento esperado, permitindo que o sujeito realize uma manutenção dos *significados* produzidos a partir do *feedback* oferecido pelo objeto técnico.

O *núcleo* é constituído pelas *crenças-afirmações* iniciais do sujeito que orientam sua interação com o objeto técnico — como a *crença* de que uma fórmula matemática pode ser transposta diretamente para uma linguagem de programação, conforme será explorado na sequência. Essas *crenças* podem ser modificadas ao longo da depuração, à medida que o sujeito ajusta sua abordagem às exigências do objeto técnico. O *campo semântico*, por sua vez, compreende o conjunto de *significados* produzidos ao longo da atividade, modificando-se conforme o sujeito integra novos elementos e adapta sua interação com o objeto técnico para alinhar o processo ao comportamento esperado.

No caso do software Word, por exemplo, a depuração de erros de sintaxe é um processo contínuo e dinâmico. O programa destaca automaticamente com sublinhados vermelhos e azuis erros de grafia e concordância, respectivamente, e, ao exibir essas

[17] Compilador é o programa que traduz um código descrito em uma linguagem de programação para seu equivalente em linguagem de máquina.

indicações visuais, pode disparar o processo de depuração. Nesse contexto, o sujeito revisa e ajusta o texto, buscando alinhá-lo às normas sintáticas. Contudo, mesmo com esses mecanismos de suporte, é importante notar que alguns erros de sintaxe não são detectados automaticamente, exigindo que o sujeito mobilize repertórios para avaliar e ajustar a estrutura textual quando necessário.

Assim, o processo de depuração de erros de sintaxe no MCS caracteriza-se por um movimento contínuo de interação entre o sujeito e o objeto técnico, com o objetivo de alinhar o código, o texto ou qualquer estrutura ao funcionamento esperado pelo sistema. Esse movimento envolve a identificação, o ajuste e a validação das estruturas utilizadas, orientando tanto o processo de produção de *significados* quanto o objetivo inicial do sujeito ao longo da atividade.

Para exemplificar o processo de depuração de erros de sintaxe, considere a situação: um professor solicita aos alunos que criem um programa em Python para calcular a área de um polígono regular. O enunciado proposto é: "Escreva um programa em Python que permita ao usuário escolher o número de lados n e o comprimento l do lado de um polígono regular e, em seguida, calcule sua área automaticamente. Registre os passos percorridos no processo de construção e descreva os ajustes realizados".

Suponha que, ao responder ao enunciado, Eduarda elabore o seguinte registro ao longo da atividade, conforme apresentado na sequência.

> Comecei escrevendo o código para calcular a área de um polígono regular usando a fórmula matemática que conheço. A ideia era que o programa pedisse ao usuário o valor de n e l, então usei `input()` para isso e escrevi a fórmula conforme imaginei que funcionaria no Python. O código ficou assim [Figura 1].
>
> Mas, ao tentar rodar o código, apareceu uma mensagem de erro logo na primeira linha da fórmula da área. Parece que o Python não entendeu o que eu escrevi. Refleti sobre o problema e percebi que talvez a entrada `input()` de n e l precisa ser convertida em número, já que é recebida como texto. Fiz o ajuste trocando `input()` por `int(input())` para garantir que o Python entenda que n e l são números inteiros. Esse ajuste parece resolver o problema [Figura 2].
>
> Dessa vez o código rodou sem erros e apresentou um valor, mas o resultado não pareceu correto para um polígono com $n = 3$ e $l = 2$. Percebi que, apesar de a fórmula agora estar correta, alguns valores fracionários não estavam sendo computados corretamente. Refleti e troquei `int(input())` por `float(input())`, pois assim o Python processará l como número decimal, o que é importante

> para cálculos de áreas precisos. Fiz esse último ajuste [Figura 3] e o programa finalmente apresentou o resultado esperado para diferentes valores de n e l (Registro de Eduarda).

A partir do exposto, é *plausível* inferir que Eduarda mobiliza o processo de depuração ao se deparar com uma série de erros sintáticos. Por meio de ajustes progressivos, ela altera o código para garantir que o resultado final corresponda ao esperado. Propõe-se, então, uma análise detalhada do processo de depuração realizado por Eduarda, evidenciando como suas enunciações refletem as mudanças no *objeto* constituído, a produção de *significados* ao longo da atividade e o aprimoramento contínuo do código.

Eduarda inicia seu relato com a seguinte enunciação:

> Comecei escrevendo o código para calcular a área de um polígono regular usando a fórmula matemática que conheço. A ideia era que o programa pedisse ao usuário o valor de n e l, então usei `input()` para isso e escrevi a fórmula conforme imaginei que funcionaria no Python. O código ficou assim [Figura 1] (Registro de Eduarda).

A primeira enunciação de Eduarda constitui o primeiro *conhecimento* produzido por ela no contexto da construção do programa. Esse *conhecimento* pode ser expresso da seguinte forma:

C_1 = (“Escrever o código da fórmula para área de um polígono regular utilizando o Python”, “[pois] essa fórmula é conhecida e deve funcionar quando programada da mesma forma que na matemática”).

É *plausível* inferir que Eduarda enuncia em uma *direção de interlocução* que legitima o uso direto de fórmulas matemáticas no contexto da programação, assumindo que o Python interpreta as operações e expressões da mesma maneira que a matemática tradicional. Eduarda enuncia em uma *direção de interlocução* que, em sua visão, compartilha a legitimidade de aplicar diretamente uma fórmula matemática no código, sem adaptações ou revisões sintáticas específicas do Python. Essa *direção de interlocução*, possivelmente um professor ou alguém familiarizado com o uso de fórmulas matemáticas em programação, entenderia sua escolha de programar a expressão tal como é na matemática, sem a necessidade de explicações adicionais.

O *objeto* constituído, então, é a fórmula matemática da área para polígonos

regulares, inserida no contexto de programação em Python. Eduarda acredita que essa fórmula será interpretada diretamente pelo código, partindo do pressuposto de que uma representação matemática conhecida pode ser aplicada sem modificações na programação.

O *núcleo* da atividade Eduarda fundamenta-se na *crença* de que "fórmulas matemáticas funcionam da mesma forma em linguagens de programação e na matemática formal". Essa *crença* serve como uma "verdade momentânea" que orienta a escrita inicial do código. A partir dessa estipulação local, Eduarda organiza suas primeiras ações, fundamentando a crença de que o código deve ser construído de maneira direta, em uma correspondência[18] entre a fórmula escrita no papel e sua representação em Python.

O *campo semântico* constituído, por sua vez, envolve conceitos de matemática e programação que, para Eduarda, funcionam de maneira uniforme. Ela pressupõe que elementos matemáticos, como a fórmula de área de polígonos regulares, são diretamente traduzíveis para o contexto de uma linguagem de programação, sem adaptações sintáticas ou estruturais. Esse *campo semântico* inclui a matemática e o uso de comandos em Python, como `input()` e `print()`, integrados ao cálculo da área.

Eduarda, então, elabora a primeira versão de seu código, apresentada na Figura 4.27 e correspondente à Figura 1 mencionada por ela.

Figura 4.27 – Primeira versão do código desenvolvido por Eduarda

main.py +

```
1  import math
2
3  n = input("Digite o número de lados: ")
4  l = input("Digite o comprimento do lado: ")
5
6  area = (n * (l**2)) / (4 * math.tan(math.pi / n))
7  print("A área do polígoo é: ", area)
```

Fonte: Os autores.

Na Figura 4.27, observa-se que Eduarda inicia o código importando a biblioteca[19]

[18] Embora o termo "correspondência" seja utilizado, ele não implica uma transcrição idêntica entre a fórmula matemática no papel e sua implementação em Python. Na prática, a implementação em Python busca preservar a estrutura conceitual da expressão matemática original, mas adapta-se às particularidades da linguagem de programação.

[19] No contexto da programação, biblioteca é um conjunto de subprogramas, funções e dados auxiliares utilizados no desenvolvimento de programas.

`math`, o que permite acesso a funções matemáticas adicionais, como `pi` e a função `tan` (tangente). Em seguida, ela utiliza o comando `input()` para solicitar ao usuário os valores de n (número de lados) e l (comprimento de cada lado), armazenando esses valores nas variáveis correspondentes. A área é então calculada utilizando a fórmula matemática da área de um polígono regular:

$$Área = \frac{n \cdot l^2}{4 \cdot \tan \frac{\pi}{n}}$$

Esse cálculo é realizado na linha 6, e o resultado é exibido no *console*[20] na linha 7, usando o comando `print()`.

Após testar o código, Eduarda encontra uma mensagem de erro e percebe que ele não funciona conforme esperado. Ela expressa essa dificuldade em sua enunciação: "Mas, ao tentar rodar o código, apareceu uma mensagem de erro logo na primeira linha da fórmula da área. Parece que o Python não entendeu o que eu escrevi". Essa enunciação sugere um *estranhamento* provocado pelo *resíduo de enunciação* — a mensagem de erro —, que contrasta com a expectativa inicial de Eduarda de que o Python interpretaria a fórmula matemática sem dificuldades.

Esse *estranhamento* leva Eduarda a questionar as *crenças-afirmações* que guiaram o processo de construção do código, especialmente a *crença* de que "fórmulas matemáticas funcionam da mesma forma em linguagens de programação e na matemática formal". Esse questionamento sugere o movimento de depuração e o redirecionamento de sua atividade para a revisão do código.

Eduarda, então, explicita uma nova reflexão sobre o problema:

> Mas, ao tentar rodar o código, apareceu uma mensagem de erro logo na primeira linha da fórmula da área. Parece que o Python não entendeu o que eu escrevi. Refleti sobre o problema e percebi que talvez a entrada `input()` de n e l precisa ser convertida em número, já que é recebida como texto. Fiz o ajuste trocando `input()` por `int(input())` para garantir que o Python entenda que n e l são números inteiros. Esse ajuste parece resolver o problema [Figura 2] (Registro de Eduarda).

[20] Na programação, *console* é uma ferramenta que registra e exibe informações sobre as operações executadas por um programa.

Essa enunciação pode ser formalizada como um *conhecimento* no MCS:

C_2 = ("Converter `input()` para `int(input())` permitirá que n e l sejam interpretados como números inteiros", "[pois] o Python reconhece `input()` como texto, e `int(input())` converte esses valores em números").

Esse *conhecimento* sugere um novo *objeto* constituído: a entrada `input()` como um dado a ser manipulado, ou seja, um elemento que precisa ser ajustado ao tipo de dado adequado para o processamento esperado pelo Python. Esse ajuste altera o foco de Eduarda, que passa da aplicação direta da fórmula matemática para o tratamento dos dados de entrada, considerando como a linguagem Python lida com diferentes tipos de dados. Nesse momento, o *objeto* deixa de ser exclusivamente a fórmula matemática e passa a incluir uma sequência de operações que devem se adequar às normas sintáticas e de tipagem da linguagem Python.

A introdução do comando `int(input())` sinaliza uma mudança no *núcleo* da atividade, que antes era orientado pela *crença* de que a fórmula poderia ser transposta diretamente para a linguagem Python. Agora, o *núcleo* passa a ser a compreensão de que o Python interpreta `input()` como uma *string* e que as entradas precisam ser convertidas explicitamente para tipos numéricos para que sejam tratadas como números pelo compilador. Esse ajuste no *núcleo* reflete uma nova "verdade momentânea" que guia as próximas ações de Eduarda, à medida que ela explora e ajusta o código.

O *campo semântico* de Eduarda também se modifica para incluir o conceito de tipos de dados em programação, abrangendo não apenas a matemática da fórmula de área, mas também os *significados* produzidos sobre os tipos e a manipulação de dados no contexto do Python. Esse *campo semântico* agora envolve a compreensão de como o Python processa entradas e a necessidade de conversão de dados, além dos conceitos matemáticos que orientam o cálculo da área.

Essa modificação indica um redirecionamento no processo de produção de *significados*: Eduarda percebe que a formulação de um programa exige que suas ações sejam ajustadas para atender às estruturas sintáticas e lógicas do Python. O processo de depuração, nesse caso, envolve a identificação e correção de problemas sintáticos, bem como

uma adaptação conceitual sobre como uma fórmula matemática é traduzida para uma linguagem de programação.

A Figura 4.28 apresenta o código revisado por Eduarda, correspondente à Figura 2 mencionada por ela.

Figura 4.28 – Segunda versão do código desenvolvido por Eduarda

main.py

```
1  import math
2
3  n = int(input("Digite o número de lados: "))
4  l = int(input("Digite o comprimento do lado: "))
5
6  area = (n * (l**2)) / (4 * math.tan(math.pi / n))
7  print("A área do polígoo é: ", area)
```

Fonte: Os autores.

Na Figura 4.28, a segunda versão do código mostra o ajuste nas linhas 3 e 4, em que Eduarda utiliza `int(input("..."))` para converter as entradas n e l em valores inteiros. Esse ajuste evidencia a compreensão de Eduarda sobre como o Python processa entradas de dados, aproximando-a da realização do cálculo da área de forma funcional e precisa.

Apesar das modificações realizadas, Eduarda relata que o código ainda não funciona conforme esperado. Embora o programa seja executado sem apresentar mensagens de erro e produza um valor, o resultado não parece correto, como ela descreve:

> Dessa vez o código rodou sem erros e apresentou um valor, mas o resultado não pareceu correto para um polígono com $n = 3$ e $l = 2$. Percebi que, apesar de a fórmula agora estar correta, alguns valores fracionários não estavam sendo computados corretamente (Registro de Eduarda).

Esse novo *estranhamento*, evidenciado pela discrepância entre o valor apresentado e o resultado idealizado, sugere uma inadequação do tipo de dado, o que desencadeia mais um movimento de depuração.

Refletindo sobre o erro, Eduarda realiza sua última enunciação no interior da atividade:

> Refleti e troquei `int(input())` por `float(input())`, pois assim o Python

> processará l como número decimal, o que é importante para cálculos de áreas precisos. Fiz esse último ajuste [Figura 3] e o programa finalmente apresentou o resultado esperado para diferentes valores de n e l (Registro de Eduarda).

Essa enunciação pode ser expressa como o seguinte *conhecimento* no MCS:

C_3 = ("Trocar `int(input())` por `float(input())` permitirá que o Python trate l como decimal, produzindo valores mais precisos", "[pois] há casos em que tanto o tamanho do lado quanto a área dos polígonos podem ser valores decimais e são importantes para a precisão e funcionamento do código").

O código final construído por Eduarda é apresentado a seguir, na Figura 4.29 (representando a Figura 3 mencionada por Eduarda).

Figura 4.29 – Versão final do código desenvolvido por Eduarda

main.py +

```
1  import math
2
3  n = int(input("Digite o número de lados: "))
4  l = float(input("Digite o comprimento do lado: "))
5
6  area = (n * (l**2)) / (4 * math.tan(math.pi / n))
7  print("A área do polígoo é: ", area)
```

Fonte: Os autores.

Observa-se, na Figura 4.29, a alteração na linha 4, em que `int(input(...))` foi substituído por `float(input(...))`. Com essa modificação, é *plausível* inferir que Eduarda passa a operar com um novo *objeto* constituído: os valores de entrada no formato decimal, garantindo que o programa considere frações na execução de cálculos matemáticos mais detalhados. Agora, o *objeto* constituído não se limita à fórmula da área; ele também inclui a entrada l como um número decimal, refletindo a compreensão de Eduarda sobre a necessidade de tratar o comprimento do lado como uma medida contínua para cálculos geométricos precisos, compatível com operações que exigem exatidão matemática.

O *núcleo* da atividade se modifica para incluir a necessidade de precisão numérica e a importância dos tipos de dados no contexto da programação. A *crença* de que valores decimais devem ser representados como *float*[21] para assegurar a precisão dos cálculos

[21] No contexto da programação, um dado *float* (ou número de ponto flutuante) representa um número

matemáticos torna-se, assim, uma estipulação local que orienta as ações de depuração e construção do código de Eduarda.

O *campo semântico* também se altera, incorporando conceitos de precisão numérica e tipos de dados em Python. Ao perceber a necessidade de precisão na representação decimal, Eduarda passa de uma produção de *significados* estritamente algébrica do cálculo de área para uma mais técnica, considerando como o Python lida com operações decimais. Esse *campo semântico* passa a envolver não apenas o cálculo geométrico, mas também os aspectos técnicos dos tipos de dados e suas implicações na precisão dos cálculos.

O caso de Eduarda ilustra como o processo de depuração é guiado por *estranhamentos* e ajustes contínuos que surgem à medida que ele confronta os resultados de sua execução com as expectativas iniciais. Cada erro de sintaxe que Eduarda encontra reflete um ponto de ruptura em suas *crenças-afirmações*. Suas tentativas de implementar a fórmula diretamente no código, produzir *significados* para `input()` como um número inteiro, e, por fim, adotar `float()` para garantir a precisão dos valores sugerem uma apropriação progressiva das características da linguagem de programação Python.

Ao longo do processo, Eduarda internaliza novas legitimidades sobre a sintaxe e os tipos de dados, demonstrando como o processo de depuração envolve tanto ajustes estruturais quanto reestruturações conceituais, adequando seu entendimento para que a solução atenda ao objetivo inicial da atividade.

Contudo, mesmo que um código esteja sintaticamente correto, isso não garante que ele realize a função idealizada por seu autor, podendo ainda apresentar um funcionamento parcial ou incorreto. Para que o resultado esperado seja efetivamente alcançado, é necessário que também não haja erros de semântica na solução (Teixeira; Juvanelli; Dantas, 2024).

Para ilustrar como a correção sintática por si só não assegura a coerência de uma solução, considere o caso de Roberto, que elabora um relatório acadêmico no Word e deseja que o texto esteja conforme as normas da língua portuguesa. Ao revisar o documento, Roberto relata:

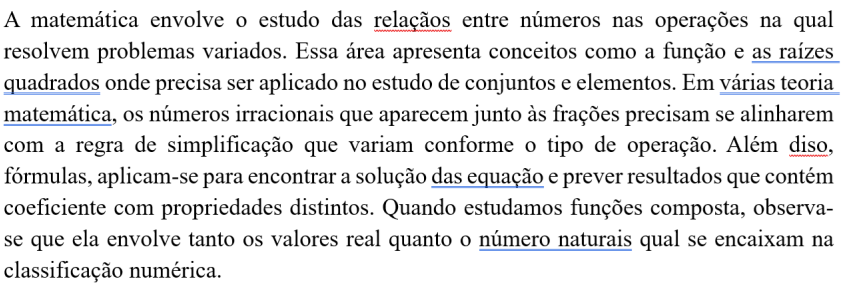
---

real com casas decimais. Ele é chamado de "ponto flutuante" porque a posição da vírgula decimal (ou ponto, dependendo da notação) pode variar, permitindo representar números muito grandes ou muito pequenos.

> Comecei escrevendo meu relatório normalmente, mas ao revisar, percebi várias palavras sublinhadas em vermelho. Decidi verificar o que o Word apontava para garantir que a ortografia estivesse correta. Ao revisar os sublinhados vermelhos, notei que várias frases estavam sublinhadas em azul, o que indica um problema de concordância ou pontuação. Ajustei a estrutura das frases de acordo com as indicações do software (Registro de Roberto).

A Figura 4.30, apresentada a seguir, ilustra o texto inicial de Roberto com as marcações realizadas pelo Word.

Figura 4.30 – Relatório escrito por Roberto



A matemática envolve o estudo das relações entre números nas operações na qual resolvem problemas variados. Essa área apresenta conceitos como a função e as raízes quadrados onde precisa ser aplicado no estudo de conjuntos e elementos. Em várias teoria matemática, os números irracionais que aparecem junto às frações precisam se alinharem com a regra de simplificação que variam conforme o tipo de operação. Além diso, fórmulas, aplicam-se para encontrar a solução das equação e prever resultados que contém coeficiente com propriedades distintos. Quando estudamos funções composta, observa-se que ela envolve tanto os valores real quanto o número naturais qual se encaixam na classificação numérica.

Fonte: Os autores.

Na Figura 4.30, observam-se palavras sublinhadas em vermelho, sinalizando erros de ortografia, além de trechos sublinhados em azul, indicando problemas de concordância e pontuação[22]. Embora Roberto corrija os erros apontados pelo Word – que são, em sua maioria, de natureza sintática —, essa correção não implica necessariamente que o texto faça sentido ou possua uma lógica interna coerente.

Esse exemplo evidencia que, tanto na depuração de textos quanto na de códigos, a ausência de erros sintáticos não garante que a solução apresentada seja semanticamente adequada ou que cumpra o propósito idealizado pelo sujeito. No caso de um texto no Word, corrigir ortografia e concordância (sintaxe) pode resultar em frases formalmente corretas; contudo, isso não assegura que o texto tenha consistência interna ou que as ideias estejam organizadas de forma coerente e inteligível. No contexto da programação, o mesmo se aplica: um código livre de erros de sintaxe ainda pode produzir resultados incorretos ou imprecisos, caso a lógica subjacente (semântica) não esteja alinhada ao objetivo da atividade.

[22] Há, inclusive, erros de concordância que não foram apontados pelo Word, como no trecho “Quando estudamos funções composta, [...]”.

Portanto, o movimento da depuração visa corrigir tanto erros de sintaxe quanto de semântica. A seguir, discute-se sobre os erros de semântica no âmbito da depuração, à luz do MCS, evidenciando como esses erros impactam a lógica e a coerência no desenvolvimento de soluções.

4.4.2 Depuração de erros de semântica desde a perspectiva do MCS

De acordo com Teixeira, Juvanelli e Dantas (2024), em contextos que envolvem objetos técnicos, a semântica refere-se ao funcionamento das estruturas sintáticas, às relações lógicas entre elas e aos resultados gerados por essas relações. Assim, a semântica vai além da estrutura formal do código ou das instruções; ela diz respeito à correspondência entre o projeto mental do sujeito e a execução realizada pelo objeto técnico. Erros de semântica, portanto, ocorrem quando o resultado gerado pelo objeto técnico — por exemplo, um software de programação ou uma ferramenta de construção geométrica — não se alinha com o objetivo inicial do sujeito. Tais erros não resultam de problemas na estrutura sintática, mas de inadequações nas relações e operações que sustentam o planejamento original.

Erros de semântica se manifestam quando o objeto técnico não produz um efeito compatível com a construção idealizada, revelando um descompasso entre o que o sujeito pretende e o que o objeto técnico executa. Ao contrário dos erros de sintaxe, que dizem respeito à conformidade estrutural e geralmente são sinalizados pelo próprio objeto técnico, os erros de semântica demandam uma revisão lógica do problema e ajustes nas relações entre as operações e os elementos utilizados na construção. Quando a solução obtida não atende ao planejamento inicial, é necessário revisar os elementos e suas relações para que o resultado final corresponda ao objetivo projetado.

Na perspectiva do MCS, a depuração de erros de semântica pode ser compreendida como um movimento de produção e manutenção dos *significados*, bem como uma reorganização das *crenças* que orientam a atividade do sujeito, em interação com os resultados apresentados pelo objeto técnico. Esse movimento não implica apenas uma adaptação do sujeito às demandas do sistema, mas exige uma análise da lógica subjacente ao problema,

promovendo ajustes nas *crenças-afirmações* que orientam a atividade. Diferentemente da depuração de sintaxe, que opera em um nível estrutural, a depuração de semântica no MCS requer uma reformulação do *núcleo* da atividade e um ajuste no *campo semântico*, buscando alinhá-los ao efeito projetado pelo sujeito.

Esse processo é guiado pela constatação de que a representação inicial do problema não gera o comportamento ou o resultado esperado, o que implica uma reavaliação das crenças que norteiam a atividade. No caso de um erro de semântica, o *núcleo* pode envolver a *crença* de que uma determinada relação ou lógica solucionaria o problema. Contudo, ao observar resultados divergentes, o sujeito ajusta suas projeções e reformula o *núcleo* da atividade para reorganizar as relações de acordo com a lógica do problema. O *campo semântico*, então, é ampliado ou redirecionado, incorporando novas relações e elementos que aproximem a construção da intencionalidade inicial, resultando em uma estrutura mais próxima do objetivo projetado.

Dessa forma, a depuração de erros de semântica no MCS caracteriza-se como um processo de produção, manutenção e refinamento de *significados*, em que o sujeito modifica o *núcleo* e o *campo semântico* da atividade para realinhar o objeto construído à sua intenção original.

Para ilustrar os erros de semântica, considere a situação: em uma aula de geometria, o professor propõe que os alunos utilizem o GeoGebra para construir um triângulo equilátero com base na medida do lado fornecida pelo usuário. O enunciado é o seguinte: “Usando o GeoGebra, construa um triângulo equilátero a partir de um segmento de comprimento l definido pelo usuário. Não utilize a ferramenta “Polígono Regular”. Registre os passos e os ajustes realizados no processo de construção”.

Suponha que Rafael desenvolva o seguinte registro ao longo de sua atividade, conforme expõe o Quadro 4.8.

Quadro 4.8 – Registro de Rafael

Direção de interlocução	**Enunciação**
1ª direção	Comecei a construção criando um controle deslizante a, configurando-o para variar entre 1 e 10, com incremento de 1, para definir o comprimento dos lados do triângulo equilátero. Em seguida, utilizando a ferramenta *Segmento com Comprimento Fixo*, desenhei um segmento de reta AB com o comprimento controlado pelo controle deslizante a, representando o primeiro lado do triângulo. Para criar o terceiro vértice, posicionei o ponto C manualmente, ajustando-o visualmente para tentar deixar o triângulo com todos os lados iguais. Depois, usei a ferramenta *Polígono* para marcar os pontos A, B e C, formando o triângulo [Figura 1].
2ª direção	Mas, ao alterar o valor de a no controle deslizante, percebi que o triângulo não mantinha a igualdade dos lados — parecia equilátero só em alguns casos [Figura 2]. Para resolver isso, decidi usar a ferramenta *Círculo: Centro & Raio*. Primeiro, criei um círculo com centro em A e raio a; depois, outro com centro em B e o mesmo raio a. Posicionei C na interseção dos dois círculos para garantir que ele ficasse a uma distância igual de A e B, assegurando que os três lados fossem realmente congruentes. Por fim, refiz o triângulo usando novamente a ferramenta *Polígono* e selecionei os três pontos A, B e C para formar o triângulo equilátero [Figura 3]. Testei para alguns valores diferentes de a no controle deslizante e confirmei que a construção agora estava funcionando conforme planejado — o triângulo permanecia equilátero para qualquer valor de a!

Fonte: Elaborado pelo autor (2025).

Ao longo de seu registro, Rafael enuncia para duas *direções de interlocução* distintas. Inicialmente, ao posicionar o ponto C visualmente, ele enuncia em uma *direção de interlocução* que considera uma construção visual como suficiente para definir um triângulo equilátero. Esta *direção de interlocução* legitima o uso de ajustes intuitivos, baseados em uma representação gráfica direta, sem exigir uma fundamentação geométrica rigorosa.

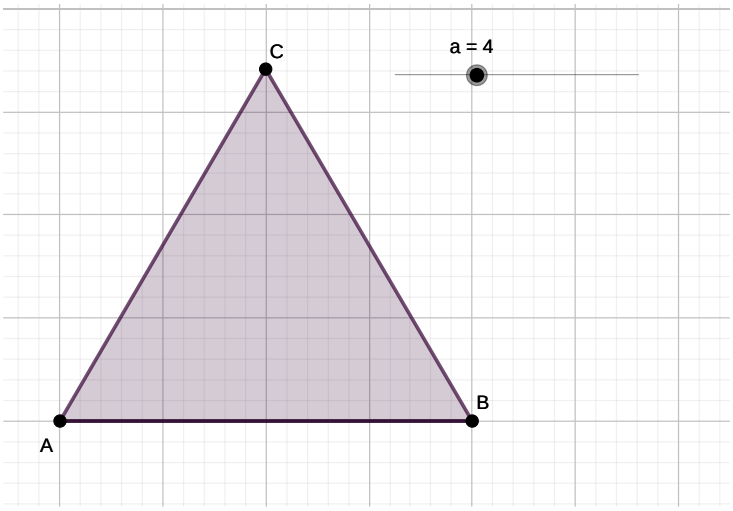
Posteriormente, ao utilizar a ferramenta *Círculo: Centro & Raio* para reposicionar o ponto C de forma mais precisa, Rafael passa a enunciar para em uma *direção de*

interlocução que ele supõe valorizar uma abordagem geométrica formal, compreendendo que o uso de círculos como ferramenta de construção assegura a congruência dos lados do triângulo.

A análise a seguir explora essas duas *direções de interlocução* e como cada uma reflete o processo de produção de *significados* no contexto da atividade de Rafael.

Rafael inicia suas enunciações explicando como a construção inicial foi realizada, utilizando uma estratégia visual para obter a representação da figura almejada. O resultado obtido por Rafael é apresentado na Figura 4.31, correspondente à Figura 1 mencionada por ele.

Figura 4.31 – Construção inicial do triângulo de Rafael



Fonte: Os autores.

Na Figura 4.31, nota-se o uso de um controle deslizante $a = 4$, o qual foi utilizado como base para a geração do segmento AB. Conforme enuncia Rafael:

> Comecei a construção criando um controle deslizante a, configurando-o para variar entre 1 e 10, com incremento de 1, para definir o comprimento dos lados do triângulo equilátero. Em seguida, utilizando a ferramenta *Segmento com Comprimento Fixo*, desenhei um segmento de reta AB com o comprimento controlado pelo controle deslizante a, representando o primeiro lado do triângulo. Para criar o terceiro vértice, posicionei o ponto C manualmente, ajustando-o visualmente para tentar deixar o triângulo com todos os lados iguais. Depois, usei a ferramenta *Polígono* para marcar os pontos A, B e C, formando o triângulo [Figura 1] (Registro de Rafael).

Assim, o ponto C foi posicionado manualmente, com Rafael buscando identificar visualmente um ponto que aparentasse ser equidistante de A e B. Em seguida, ele utilizou

a ferramenta *Polígono* para construir o triângulo.

Essa enunciação corresponde a um *conhecimento* no âmbito do MCS, o qual pode ser expresso da seguinte forma:

C_1 = ("Posicionar o ponto C manualmente permite formar um triângulo equilátero visualmente", "[pois] o alinhamento visual dos vértices A, B e C sugere que os lados do triângulo são congruentes").

A *justificação* apresentada por Rafael sugere que ele enuncia em uma *direção de interlocução* que, ele acredita, compartilha da legitimidade da representação visual como critério para a construção geométrica. Essa *direção de interlocução* legitima o posicionamento manual como uma solução inicial aceitável para a construção de um triângulo equilátero, sem questionar o rigor geométrico.

Assim, é *plausível* inferir que o *objeto* constituído inicialmente por Rafael é a representação visual do triângulo equilátero, baseada em um posicionamento intuitivo. Rafael concebe o triângulo como uma figura visual e manipula o ponto C na tentativa de criar uma forma que aparentemente atenda aos requisitos geométricos de um triângulo equilátero. Essa abordagem inicial reflete a *crença* de que o alinhamento visual seria suficiente para garantir a congruência dos lados.

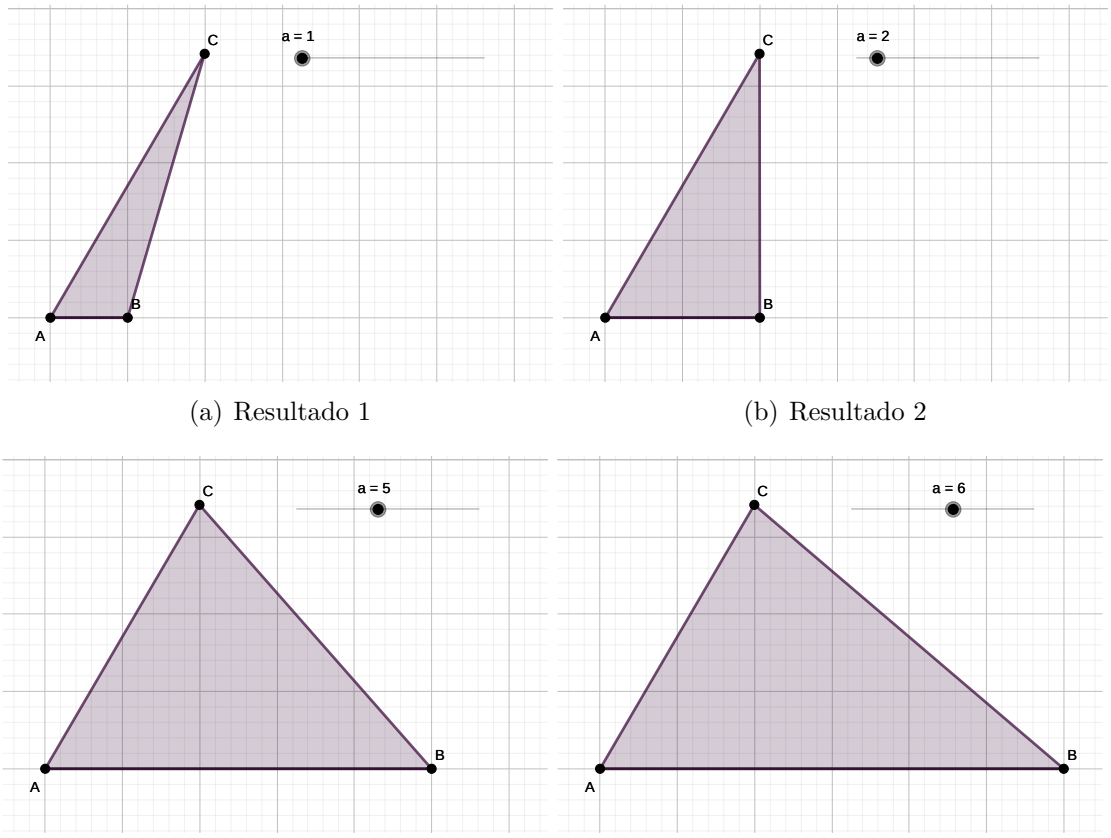
O *núcleo* da atividade de Rafael, até então, fundamenta-se na *crença* de que o triângulo equilátero pode ser construído por meio de ajustes visuais. Essa estipulação local organiza suas ações iniciais, guiando-o na escolha de ferramentas e métodos para posicionar o ponto C com base na observação. Essa *crença*, ainda não problematizada, constitui a base sobre a qual Rafael estrutura sua primeira tentativa de construção.

O *campo semântico*, nesse momento, abrange noções de geometria e manipulação gráfica no GeoGebra. A compreensão de Rafael sobre congruência de lados e construção de triângulos está inicialmente centrada na visualização da figura e ao uso das ferramentas de construção disponíveis no GeoGebra. Rafael opera em seu *campo semântico* de maneira informal – isto é, não opera com as legitimidades da Matemática de um matemático –, associando diretamente as características visuais do triângulo ao conceito de equilátero.

Ao concluir sua primeira tentativa, Rafael acredita que sua construção inicial representa um triângulo equilátero. No entanto, ao modificar o valor de a no controle deslizante, ele percebe que a figura gerada não mantém as propriedades esperadas de um triângulo equilátero, conforme explicita em sua enunciação: "Mas, ao alterar o valor de a no controle deslizante, percebi que o triângulo não mantinha a igualdade dos lados – parecia equilátero só em alguns casos [Figura 2]" (Registro de Rafael). Esse *estranhamento* sinaliza uma ruptura nas *crenças* de Rafael e sugere o início de um movimento de depuração.

A Figura 4.32, correspondente à Figura 2 mencionada por Rafael, apresenta alguns dos resultados obtidos por ele.

Figura 4.32 – Triângulo com controle deslizante sem manutenção das propriedades de equilátero



(a) Resultado 1 (b) Resultado 2

(c) Resultado 3 (d) Resultado 4

Fonte: Os autores.

Na Figura 4.32, observa-se que, embora o segmento AB obedeça ao valor estabelecido pelo controle deslizante a, os segmentos AC e BC não mantêm a mesma proporção, pois foram construídos manualmente e ajustados visualmente para representar um triângulo

equilátero, sem uma dependência direta do controle deslizante. Como consequência, ao variar o valor de a, o triângulo ABC perde a congruência de seus lados, confirmando o erro na construção inicial de Rafael.

Esse erro não se configura como um erro de sintaxe, pois as operações e comandos do GeoGebra foram realizados corretamente. Em vez disso, trata-se de um problema semântico, uma vez que o que foi projetado por Rafael não representa adequadamente o conceito geométrico de um triângulo equilátero – a relação de congruência entre os três lados. Este erro ilustra uma discrepância entre o que Rafael idealizou como triângulo equilátero e a construção efetivamente realizada no GeoGebra, levando-o a revisar sua estratégia.

Rafael prossegue com sua reflexão, enunciando:

> Para resolver isso, decidi usar a ferramenta *Círculo: Centro & Raio*. Primeiro, criei um círculo com centro em A e raio a; depois, outro com centro em B e o mesmo raio a. Posicionei C na interseção dos dois círculos para garantir que ele ficasse a uma distância igual de A e B, assegurando que os três lados fossem realmente congruentes (Registro de Rafael).

Essa enunciação sugere um novo *conhecimento*:

C_2 = ("Posicionar o ponto C na interseção dos círculos garante que os lados AC e BC sejam congruentes a AB", "[pois] ao estabelecer a distância a como raio de ambos os círculos, assegura-se a congruência dos três lados do triângulo").

Esse *conhecimento* evidencia um ajuste no modo como Rafael legitima a construção do triângulo equilátero, agora fundamentado em uma dependência geométrica rigorosa e consistente[23]. A explicação da *justificação* para essa *crença-afirmação* indica uma mudança na *direção de interlocução* de Rafael. Ele passa a enunciar para em uma *direção de interlocução* que reconhece a legitimidade de métodos geométricos precisos, como a criação de pontos na interseção de círculos para garantir a congruência dos lados de um triângulo. Essa *direção de interlocução* compartilha a expectativa de precisão e valida a nova abordagem de Rafael como um caminho mais apropriado para alcançar o triângulo equilátero.

[23] Os termos "rigorosa" e "consistente" referem-se à atuação conforme as legitimidades próprias da Matemática do matemático.

O *objeto* constituído por Rafael agora é um triângulo equilátero construído com uma dependência geométrica precisa. Ao utilizar a interseção dos círculos para posicionar o ponto C, Rafael redefine o triângulo equilátero, não mais como um ajuste visual, mas como uma construção apoiada em uma congruência geométrica. Isso sugere que Rafael não apenas considera o triângulo como uma figura geométrica composta por lados iguais, mas também como um *objeto* que depende de propriedades formais para sustentar suas características de equilátero.

O *núcleo* da atividade de Rafael se modifica com a introdução da ideia de congruência controlada por uma construção geométrica exata. A *crença* inicial de que uma representação visual seria suficiente para garantir um triângulo equilátero é substituída por uma nova estipulação local: "Para garantir a congruência dos lados de um triângulo equilátero, é necessário um processo de construção que assegure essa igualdade, como a utilização de círculos de mesmo raio". Esse *núcleo* orienta as ações subsequentes de Rafael, agora guiadas pela precisão geométrica em detrimento de ajustes visuais.

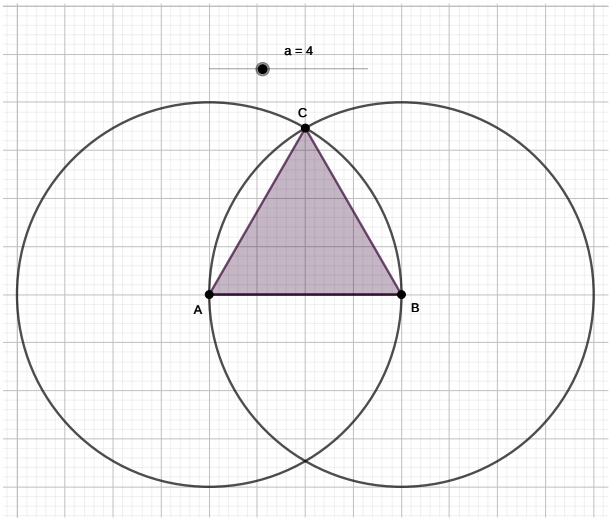
O *campo semântico* se expande, passando a incluir conceitos de geometria que garantem a congruência entre segmentos, como o uso de círculos de mesmo raio para estabelecer uma equidistância entre pontos. Esse *campo semântico* agora abrange outro modo de produção de *significados*: para que o triângulo mantenha suas propriedades de equilátero, todos os lados precisam ser controlados por uma relação de dependência geométrica.

Com essa nova abordagem, Rafael demonstra compreender as propriedades geométricas necessárias para a construção de um triângulo equilátero no GeoGebra, conforme elucida em sua enunciação:

> Por fim, refiz o triângulo usando novamente a ferramenta *Polígono* e selecionei os três pontos A, B e C para formar o triângulo equilátero [Figura 3].
>
> Testei para alguns valores diferentes de a no controle deslizante e confirmei que a construção agora estava funcionando conforme planejado — o triângulo permanecia equilátero para qualquer valor de a! (Registro de Rafael).

A Figura 4.33 apresenta o resultado obtido por Rafael (correspondente à Figura 3 mencionada por ele).

Figura 4.33 – Construção final do triângulo equilátero com dependência geométrica de congruência



Fonte: Os autores.

Conforme ilustra a Figura 4.33, a utilização dos círculos para posicionar o ponto C representa uma mudança significativa em sua atividade de construção, evidenciando um processo de depuração focado em ajustes semânticos, e não em aspectos sintáticos. Enquanto um ajuste sintático, nesse caso, poderia envolver a aplicação de comandos ou ferramentas específicas (como a seleção adequada dos pontos ou a utilização correta das ferramentas no GeoGebra), o ajuste semântico implica na produção de *significados* que relacionem os conceitos matemáticos de congruência e dependência geométrica para atingir o objetivo da referida atividade. Nesse caso, Rafael, *plausivelmente*, passa a compreender a congruência geométrica entre os lados do triângulo de maneira a garantir sua propriedade de equilátero, independentemente do valor de a no controle deslizante.

Esse triângulo, agora construído a partir de uma nova compreensão sobre congruência, gera uma figura consistente com os parâmetros estabelecidos e mantém sua propriedade de equilátero independentemente das alterações em a, como Rafael afirma em sua última enunciação: "Testei para alguns valores diferentes de a no controle deslizante e confirmei que a construção agora estava funcionando conforme planejado – o triângulo permanecia equilátero para qualquer valor de a!".

O caso de Rafael exemplifica o processo de depuração de erros de semântica em sua atividade, destacando como um *estranhamento* inicial – provocado pela incongruência

dos lados do triângulo – desencadeia uma série de revisões rumo a uma construção mais robusta e semanticamente adequada. Ao ajustar a construção para assegurar a congruência dos lados, Rafael não apenas resolve um problema técnico, mas realiza uma manutenção dos *significados* que produz para as propriedades que definem um triângulo equilátero, passando a operar a partir de um *núcleo* baseado na precisão geométrica.

A mudança no *objeto* constituído, que passa de uma representação visual inicial para um triângulo equilátero construído com dependência geométrica, reflete o processo de produção de manutenção dos *significados* produzidos por Rafael no interior da atividade. Essa transição é mediada pela *direção de interlocução* que Rafael assume ao internalizar legitimidades de rigor geométrico, permitindo-lhe reestruturar suas *justificações* e ações para alcançar uma figura que efetivamente represente o triângulo equilátero.

Assim, o processo de depuração, orientado pelos erros de semântica, manifesta-se não como uma mera correção de execução, mas como um processo de produção de *significados* que envolve a revisão e adaptação dos conceitos matemáticos para atender aos requisitos de uma construção geométrica precisa. Essa reformulação destaca o caráter dinâmico e contínuo da depuração, em que o processo de revisão é mediado pelo desenvolvimento de uma compreensão mais aprofundada dos conceitos envolvidos e de suas relações geométricas.

Na sequência, realizam-se as considerações finais sobre a depuração no âmbito do MCS.

4.4.3 Considerações sobre a depuração desde a perspectiva do MCS

A análise da depuração no contexto do MCS revela esse processo como um movimento contínuo e dinâmico de produção e manutenção de *significados*, que ultrapassa a mera correção de erros. No âmbito do Pensamento Computacional, a depuração não se restringe à eliminação de problemas técnicos ou ao ajuste imediato de sintaxe; trata-se de um processo interativo e iterativo de refinamento conceitual, no qual o sujeito reavalia suas *crenças-afirmações* ao deparar-se com resultados imprevistos ou inconsistências lógicas em relação às expectativas iniciais.

No MCS, a depuração pode ser compreendida como um processo de *estranhamento* que ocorre quando o *objeto* realizado em uma atividade não corresponde ao *objeto* idealizado no momento do design. Esse *estranhamento* emerge quando há uma ruptura entre o que se esperava produzir e o que, de fato, foi produzido. Nesse momento, torna-se evidente que as estipulações locais que sustentavam o *núcleo* da atividade deixam de se sustentar. A depuração, nesse sentido, marca o início de um movimento de instabilidade e reorganização do *núcleo* e, consequentemente, de mudança no *campo semântico* da atividade.

Esse processo de estranhamento pode se manifestar de diferentes formas. A distinção entre erros de sintaxe e de semântica evidencia que esses tipos de erros desencadeiam movimentos distintos de revisão e reorganização. Em casos de depuração de sintaxe, o *estranhamento* incide sobre os comandos, blocos ou estruturas formais que não operam como esperado – seja por erro de escrita, ordem ou uso inadequado da linguagem. Como demonstrado na análise do caso de Eduarda, esses erros geralmente envolvem questões formais e estruturais, refletindo a produção de *significados* vinculada à conformidade com normas específicas da linguagem ou do objeto técnico. A depuração de erros de sintaxe pode ser compreendida, então, como um processo de alinhamento entre a intenção do sujeito e as exigências formais do objeto técnico.

Por outro lado, na depuração de semântica, o *estranhamento* é mais profundo: refere-se à percepção de que o *objeto* realizado – mesmo funcional do ponto de vista técnico – não expressa aquilo que se pretendia comunicar ou alcançar com a atividade. No exemplo de Rafael, a criação do triângulo equilátero passa a depender de uma compreensão geométrica mais rigorosa, levando-o a substituir uma construção baseada em ajustes visuais por uma abordagem fundamentada em relações geométricas precisas. Nesse caso, a depuração exige não apenas adequação técnica, mas também a modificação do *núcleo* e a modificação do *campo semântico* da atividade, de forma a garantir a coerência entre a solução obtida e o objetivo projetado.

O *estranhamento*, ao produzir essa tensão entre o esperado e o realizado, instaura um movimento de *descentramento*. Conforme preconiza o MCS, esse *descentramento* consiste em suspender provisoriamente as certezas que sustentavam o *núcleo* anterior

e permitr que novas estipulações possam ser constituídas como legítimas. Dessa forma, a depuração não se limita à correção de erros técnicos ou formais, mas implica um reposicionamento do sujeito na atividade, abrindo novas possibilidades de produção de *significados*. Em alguns casos, isso pode levar à reconstrução integral da atividade – um novo design –; em outros, o sujeito pode mandar partes da construção, ajustando apenas algumas estipulações ou redefinindo parcialmente os *objetos* envolvidos.

Portanto, a depuração pode ser definida, nos termos do MCS, como um processo de produção de *significados* desencadeado por um *estranhamento*, no qual o sujeito reconhece que o *objeto* que sustenta a atividade não mais se sustenta como tal. Esse reconhecimento leva à necessidade de produzir novos *significados*, reorganizar a atividade com base em novas *direções de interlocução* e, em última instância, restaurar a coerência entre o que se deseja produzir e o que efetivamente se realiza. A depuração é, portanto, inseparável do movimento cognitivo de produção de manutenção de *significados*, e ocupa um lugar central na articulação do Pensamento Computacional na perspectiva do MCS.

O MCS desempenha um papel central na análise desses processos, permitindo captar as nuances da produção de *significados* e as reestruturações que emergem à medida que o sujeito recebe os *feedbacks* dos objetos técnicos. Por meio das noções de *núcleo*, *campo semântico*, *direção de interlocução* e *objeto*, a análise da depuração revela como as ações dos sujeitos se integram em um contexto mais amplo de práticas e modos de produção de *significados*. Inserir as discussões sobre a depuração no âmbito do MCS permite, assim, compreendê-la não como um simples ajuste técnico, mas como uma prática de redefinição de *crenças-afirmações* e reorganização da atividade em direção a um objetivo final.

Essa análise evidencia que a depuração, no contexto do Pensamento Computacional, é uma atividade fundamentalmente integrada à produção e manutenção de *significados*, na qual a busca por solução envolve tanto a precisão técnica quanto a adequação semântica e a reorganização das estruturas cognitivas que sustentam a atividade.

Na sequência, realizam-se a considerações finais desta pesquisa.

5 CONSIDERAÇÕES FINAIS

O objetivo deste trabalho foi apresentar *uma* perspectiva de Pensamento Computacional fundamentada no Modelo dos Campos Semânticos (MCS) no âmbito da Educação Matemática, investigando como esse modo de pensar pode emergir nas enunciações em atividades diversas, tanto computacionais quanto não computacionais. Para tanto, a pesquisa desenvolveu-se em duas etapas principais.

A primeira etapa consistiu em uma revisão teórica sobre o Pensamento Computacional e o MCS. Esse processo envolveu a análise das definições, abordagens e potencialidades de ambas as noções, possibilitando uma compreensão mais ampla de suas bases teóricas e a identificação de conexões entre elas. Conforme discutido no capítulo *O Pensamento Computacional desde a perspectiva do Modelo dos Campos Semânticos*, essa revisão evidenciou a relevância dessas noções para a análise da produção de *significados* em contextos educacionais, fornecendo uma fundamentação teórica consistente para compreender como os *significados* são produzidos durante a resolução de problemas.

Na segunda etapa, foram elaborados ensaios teóricos que exemplificam processos de produção de *significados* para conceitos computacionais em diferentes objetos técnicos, a partir da perspectiva do MCS. Esses ensaios se concentraram na *leitura plausível* de situações-problema e suas soluções, explorando os processos do Pensamento Computacional e analisando a constituição dos *objetos*, *núcleos* e *campos semânticos*. Os ensaios sugerem, *plausivelmente*, processos de design, decomposição, produção de algoritmos, abstração, reconhecimento de padrões, abstração e depuração, que se manifestam de acordo com o contexto da atividade e o objeto técnico em questão.

Desse modo, a partir do exposto ao longo dos capítulos, pode-se inferir que a expressão “Pensamento Computacional”, conforme frequentemente aponta a literatura (Wing, 2006, 2008; Brackmann, 2017; Boucinha et al., 2019), poderia ser mais bem descrita como “Procedimento Computacional”, uma vez que a ênfase recai mais sobre a atividade do sujeito do que sobre uma discussão aprofundada do pensamento em si. Inserida nesse cenário, a perspectiva teórico-epistemológica do MCS se mostra como uma abordagem

capaz de oferecer uma leitura significativamente precisa dos processos cognitivos subjacentes ao Pensamento Computacional, deslocando o foco do fazer computacional para as formas como os sujeitos produzem *significados* nesse contexto.

Essa perspectiva reconhece que o Pensamento Computacional não se restringe a atividades mediadas por computadores, mas emerge nas enunciações dos sujeitos em diferentes contextos, computacionais e não computacionais. No âmbito do MCS, isso implica que os processos do Pensamento Computacional podem se manifestar quando os sujeitos produzem *significados* para variados *resíduos de enunciação*, constituindo *objetos* e *núcleos* em variados *campos semânticos*.

Os *objetos* constituídos pelos sujeitos durante a resolução de problemas – nos processos de design, decomposição, produção de algoritmos, abstração, reconhecimento de padrões e depuração – não são entidades absolutas, mas construções que emergem das atividades e são legitimadas dentro de determinados *campos semânticos*. Essa visão se alinha à concepção do MCS, segundo a qual o *conhecimento* é sempre local, contingente e situado, e não uma entidade universal e absoluta.

A seleção dos seis processos fundamentais analisados neste trabalho não foi arbitrária, mas resultado de uma análise crítica da literatura sobre Pensamento Computacional, fruto de discussões sobre o tema no Grupo de Pesquisa Autômato e da produção de *significados dos autores* desta pesquisa. A partir dessa base buscou-se caracterizar cada um dos processos não como habilidades isoladas, mas como movimentos situados de produção de *significados*, definidos com base nas noções do MCS.

O design, à luz do MCS, pode ser compreendido como um metaprocesso de produção de *significados* que orienta e estrutura os demais processos do Pensamento Computacional. Ele consiste na atividade de concepção de *objetos*, legitimação de *direções de interlocução* e constituição de *núcleos*, a partir dos quais os *campos semânticos* são organizados. Não se trata apenas da formulação inicial do problema, mas de um movimento contínuo e adaptativo, no qual o sujeito articula e rearticula, produzindo *significados* para os *resíduos de enunciação* provenientes da atividade em questão. Nos exemplos analisados, o design revelou-se como o espaço em que se decide o que é relevante, que *objetos* devem

ser constituídos e quais estratégias são legítimas para a busca da solução. Ao ser tomado como metaprocesso, o design torna-se o eixo estruturante da atividade, dentro do qual os demais processos se situam, se reorganizam e ganham coerência.

A decomposição, por sua vez, foi compreendida não como uma simples fragmentação técnica do problema, mas como um processo situado de produção de *significados*. A partir da interação com os *resíduos de enunciação* do problema, o sujeito identifica e constitui *objetos*, que passam então a ser analisados separadamente. Essa análise se concretiza em ações e operações cognitivas organizadas em função de um *campo semântico* em mudança. Ao fragmentar o problema em partes manejáveis, o sujeito reorganiza seus *núcleos* e legitima outras *direções de interlocução*, permitindo o avanço da atividade. Trata-se, portanto, de um processo de constituição e reorganização de *objetos*, orientado por *estipulações locais* e articulado ao planejamento e à execução da resolução.

A produção de algoritmos foi definida como o processo em que o sujeito estabiliza uma sequência estruturada de passos, legitimada em uma *direção de interlocução* específica, para garantir a solução de um problema. Quando o sujeito consolida um *núcleo* a partir do qual essa sequência se torna replicável e generalizável, ocorre a produção de um algoritmo. Essa estabilização não decorre apenas da criação de regras formais, mas da legitimação dessas regras como válidas no contexto da atividade. No exemplo de Mariana, o uso da homotetia marcou essa transição de uma abordagem empírica para uma formalmente estruturada, revelando como a produção de algoritmos depende da constituição de um *núcleo* que organiza o *campo semântico* da resolução.

A abstração, na perspectiva do MCS, foi caracterizada como o processo de seleção e legitimação de *objetos* e modos de produção de *significados* considerados essenciais à construção da solução. Entre todas as possibilidades de produção de *significados* em uma dada atividade, o sujeito discrimina e prioriza aqueles que são tomados como operacionais e válidos dentro do *núcleo* e *campo semântico* estabelecidos. Essa seleção não é neutra nem exclusivamente lógica, mas situada, e depende da *direção de interlocução* adotada e das legitimidades internalizadas. A diferença nas soluções de Caio no Scratch e no GeoGebra ilustra como diferentes *objetos* (como “fantasia” ou “função”) foram constituídos e validados

conforme os recursos e exigências de cada ambiente. Assim, a abstração revela-se como uma prática cognitiva de organização e foco, que permite ao sujeito operar com o essencial dentro de uma atividade situada.

O reconhecimento de padrões, igualmente, foi definido no âmbito do MCS como um processo de produção de *significados* que permite a antecipação de recorrências. Essa antecipação, fundamentada em experiências anteriores e modos de produção de *significados* previamente legitimados, possibilita que o sujeito organize sua atividade de forma mais eficiente, reutilize estratégias ou transfira estruturas entre contextos. O reconhecimento de padrões não se reduz à observação de repetições, mas envolve a constituição de *objetos* com propriedades regulares que são, em determinado contexto, legítimas para o sujeito. A forma como Caio identificou e operacionalizou os padrões de alternância visual no Scratch e no GeoGebra mostra que esse processo está profundamente vinculado ao objeto técnico, ao contexto da atividade e à *direção de interlocução* assumida.

Por fim, a depuração foi definida como um processo de produção de *significados* desencadeado por um *estranhamento*, que emerge quando o *objeto* realizado não corresponde ao *objeto* idealizado no momento do design. Esse *estranhamento* gera uma instabilidade no *núcleo* da atividade, exigindo a reorganização do *campo semântico* e a produção de novos *significados* para restaurar a coerência entre intenção e resultado. O processo de depuração manifesta-se tanto na correção de erros de sintaxe (alinhamento com exigências formais do objeto técnico) quanto em erros de semântica (redefinição de conceitos e estratégias). Esse movimento de *descentramento* é essencial na articulação do Pensamento Computacional, pois permite ao sujeito reorientar sua atividade a partir de novas legitimidades. A depuração, portanto, não é apenas correção técnica, mas um exercício contínuo de revisão, reposicionamento e produção e manutenção de *significados*.

Essas definições, construídas ao longo do trabalho com base nos ensaios teóricos e nas noções do MCS, evidenciam que os processos do Pensamento Computacional não são habilidades genéricas ou universais, mas formas situadas de produção de *significados*, profundamente ancoradas na atividade, nas experiências dos sujeitos e nas relações que eles estabelecem com os objetos técnicos e os contextos em que atuam.

A perspectiva adotada neste trabalho reconhece que os seis processos do Pensamento Computacional não operam isoladamente, mas se entrelaçam e se complementam na produção de *significados*. As análises dos ensaios teóricos demonstraram essa interdependência nas enunciações dos sujeitos. Essa interdependência reflete a natureza holística do Pensamento Computacional como um modo específico de produção de *significados* que transcende categorizações rígidas.

O design, por exemplo, frequentemente incorpora elementos de decomposição, ao fracionar o problema, e de abstração, ao identificar os aspectos essenciais. A relação entre decomposição e produção de algoritmos evidencia que a primeira cria as condições para a emergência da segunda. A abstração e o reconhecimento de padrões também se articulam dinamicamente, enquanto a depuração permeia todos os processos, viabilizando a reflexão e a manutenção dos *significados* produzidos ao longo da atividade.

Os ensaios teóricos elaborados neste trabalho foram fundamentais para elucidar os processos do Pensamento Computacional sob a perspectiva do MCS. A análise das enunciações plausíveis de sujeitos em diferentes contextos e com distintos objetos técnicos permitiu observar como esses processos emergem e se manifestam nos modos de produção de *significados*.

Os exemplos com o Scratch ilustraram como o design e a produção de algoritmos contribuem para a constituição de *objetos* relacionados à programação visual. Nos casos de Ana e Pedro, ao desenvolverem algoritmos para desenhar polígonos regulares, a decomposição e a abstração emergiram nas enunciações, refletindo os *núcleos* constituídos a partir de seus modos de produção de *significados* para o *resíduo de enunciação* “enunciado da situação-problema”.

Já os ensaios com o GeoGebra, como o caso de Mariana e sua construção de um quadrado inscrito em um triângulo, evidenciaram o reconhecimento de padrões e a abstração em um contexto de geometria dinâmica. A análise dos processos de depuração, exemplificada na transição de versões iniciais para soluções refinadas, revelou como os sujeitos produzem novos *significados* a partir da identificação de inconsistências em suas abordagens.

A diversidade de objetos técnicos e contextos explorados – de ambientes de programação visual, como o Scratch, a sistemas de geometria dinâmica, como o GeoGebra – possibilitou observar as distintas, porém complementares, manifestações dos processos do Pensamento Computacional. Essa diversidade reflete a versatilidade do Pensamento Computacional como um modo de produção de *significados*.

Espera-se com este trabalho disparar discussões e trazer contribuições para a Educação Matemática, especialmente ao compreender o Pensamento Computacional como um modo específico de produção de *significados*. Uma de suas principais contribuições é a proposição de uma perspectiva epistemológica que não reduz o Pensamento Computacional a um conjunto de habilidades técnicas isoladas, mas o reconhece como processo emergente em diferentes contextos e atividades. Essa abordagem amplia seu escopo para além das fronterias da Ciência da Computação, evidenciando sua relevância para a Educação Matemática e outras áreas do conhecimento.

Além disso, a utilização do MCS como referencial teórico-epistemológico permitiu uma nova forma de analisar os processos de produção de *significados* relacionados ao Pensamento Computacional. Essa perspectiva oferece aos educadores matemáticos uma lente para compreender como os alunos produzem *significados* ao resolverem problemas, sejam eles computacionais ou não.

Por fim, os ensaios teóricos deste trabalho trazem *insights* sobre como fomentar o Pensamento Computacional em contextos educacionais, por meio de atividades que incentivem a produção de *significados* em diferentes contextos. A análise das enunciações dos sujeitos evidencia como esses processos emergem naturalmente na resolução de problemas, sugerindo abordagens pedagógicas que valorizem e promovam essa dinâmica.

Apesar das contribuições apresentadas, este estudo enfrentou desafios e limitações que merecem ser reconhecidos. Um dos principais desafios foi a articulação teórica entre o Pensamento Computacional e o MCS, duas abordagens com tradições epistemológicas distintas. A produção de *significados* para os processos do Pensamento Computacional à luz do MCS exigiu um esforço de tradução conceitual que, embora produtivo, pode ter deixado lacunas teóricas a serem exploradas em estudos futuros.

A seleção dos seis processos do Pensamento Computacional analisados, embora justificada pela literatura e pelos objetivos do estudo, pode não ter contemplado outros aspectos relevantes que emergiram paralelamente ou após o desenvolvimento desta pesquisa. A constante evolução dos campos da Ciência da Computação e da Educação Matemática implica revisões teóricas contínuas que podem complementar ou reformular a perspectiva aqui proposta.

Essas limitações apontam para direções promissoras para pesquisas futuras. Um desdobramento relevante seria a realização de estudos empíricos para refinar os *insights* teóricos apresentados, investigando como os processos do Pensamento Computacional emergem nas enunciações de estudantes em situações reais de aprendizagem.

Outra possibilidade seria a elaboração de propostas didáticas fundamentadas na perspectiva teórica desenvolvida neste trabalho, explorando como os processos do Pensamento Computacional podem ser fomentados em sala de aula por meio de atividades que promovam a produção de *significados* em diferentes contextos e com variados objetos técnicos. A implementação e avaliação dessas propostas em contextos educacionais reais gerariam dados empíricos que complementariam as análises teóricas aqui desenvolvidas.

Em síntese, este trabalho representa um passo inicial na direção de uma compreensão mais aprofundada do Pensamento Computacional como um processo de produção de *significados* fundamentado na perspectiva teórico-epistemológica do MCS. As análises e reflexões apresentadas não esgotam o tema, mas podem oferecer contribuições para a Educação Matemática e apontar caminhos promissores para investigações futuras.

REFERÊNCIAS

ANGELO, C. L. et al. **Modelo dos Campos Semânticos e Educação Matemática: 20 anos de história**. São Paulo: Midiograf, 2012.

ASBAHR, F. da S. F. A pesquisa sobre a atividade pedagógica: contribuições da teoria da atividade. **Revista Brasileira de Educação**, Rio de Janeiro, n. 29, p. 108–118, 2005. Disponível em: https://www.scielo.br/j/rbedu/a/nS8cDBnyryfhQzBLFCqrRVc/abstract/?lang=pt. Acesso em: 9 dez. 2023.

BARON, M. E. **Curso de história da matemática: origens e desenvolvimento do cálculo**. Brasília: Editora Universidade de Brasília, 1985.

BARROS, T. T. T. **Formação em Pensamento Computacional utilizando Scratch para Professores de Matemática e Informática da Educação Fundamental**. 2020. Tese (Doutorado em Informática na Educação) – Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: https://lume.ufrgs.br/handle/10183/219412. Acesso em: 19 out. 2023.

BERTAZINI, E. **Pensamento Computacional em Livros Didáticos do Ensino Médio: sobre Atividades e Possibilidades**. 2022. Tese (Doutorado em Educação Matemática) – Universidade Anhanguera de São Paulo, São Paulo. Disponível em: https://repositorio.pgsskroton.com/handle/123456789/52076. Acesso em: 19 out. 2023.

BOUCINHA, R. M. et al. Relationship between the Learning of Computational thinking and the Development of Reasoning. **International Journal of Advanced Engineering Research and Science**, [s. l.], v. 6, n. 6, p. 623–631, 2019. Disponível em: https://ijaers.com/detail/relationship-between-the-learning-of-computational-thinking-and-the-development-of-reasoning/. Acesso em: 26 out. 2023.

BRACKMANN, C. P. **Desenvolvimento do Pensamento Computacional através de atividades desplugadas na educação básica**. 2017. Tese (Doutorado em Informática na Educação) – Universidade Federal do Rio Grande do Sul, Porto Alegre. Disponível em: https://lume.ufrgs.br/handle/10183/172208. Acesso em: 19 out. 2023.

BRASIL, Ministério da Educação. **Base Nacional Comum Curricular**. Brasília: [s.n.], 2018. Disponível em: http://basenacionalcomum.mec.gov.br/abase/. Acesso em: 15 set. 2023.

BRENNAN, K.; RESNICK, M. New frameworks for studying and assessing the development of computational thinking. In: PROCEEDINGS of the 6th Annual Meeting Program. Washington, DC, USA: [s.n.], 2012. P. 1–25.

COSTA, E. B. L. da. **Charles Babbage (1791-1871) e a mecanização do cálculo: das engrenagens à máquina de pensar**. 2012. Tese (Doutorado em História da Ciência) – Pontifícia Universidade Católica de São Paulo, São Paulo. Disponível em: https://repositorio.pucsp.br/jspui/handle/handle/13273. Acesso em: 10 nov. 2024.

DANTAS, S. C. A Geometria Escolar e os pensamentos matemático e computacional. In: BALDINI, L. A. F.; MORAN, M. (Ed.). **Geometria: práticas e aprendizagens**. São Paulo: Livraria da Física, 2022. P. 19–51.

_____. **Design, implementação e estudo de uma rede sócio profissional online de professores de Matemática**. 2016. Tese (Doutorado em Educação Matemática) – Universidade Estadual Paulista, Rio Claro. Disponível em: https://repositorio.unesp.br/items/b4e343eb-33ed-4a14-badf-c3b6935ce6ef. Acesso em: 17 mar. 2024.

_____. Pensando e resolvendo problemas com o GeoGebra. **Revista do Instituto GeoGebra Internacional de São Paulo**, [s. l.], v. 12, n. 2, p. 133–164, 2023. Disponível em: https://revistas.pucsp.br/index.php/IGISP/article/view/63719. Acesso em: 29 out. 2023.

DANTAS, S. C.; FERREIRA, G. F.; PAULO, J. P. A. de. Uma noção de interação colaborativa elaborada à luz do modelo dos campos semânticos e da teoria da atividade. **Revista Paranaense de Educação Matemática**, [s. l.], v. 5, n. 8, p. 213–236, 2016. Disponível em: https://periodicos.unespar.edu.br/index.php/rpem/article/view/6018. Acesso em: 6 dez. 2023.

DANTAS, S. C.; LINS, R. C. Reflexões sobre Interação e Colaboração a partir de um Curso Online. **Bolema: Boletim de Educação Matemática**, v. 31, n. 57, p. 1–34, 2017. Disponível em: https://www.scielo.br/j/bolema/a/R7nLBYQh3YkmHjgLZwNw93j/?lang=pt. Acesso em: 4 dez. 2023.

DENNING, P. J.; TEDRE, M. **Computational Thinking**. Cambridge, MA: The MIT Press, 2019.

DISESSA, A. **Changing Minds: Computers, Learning, and Literacy**. [s. l.]: The MIT Press, 2000.

EIGENMANN, R.; LILJA, D. J. Von Neumann Computers. **Wiley Encyclopedia of Electrical and Electronics Engineering**, Wiley, [s. l.], v. 23, p. 387–400, 1998.

ESPADEIRO, R. G. O Pensamento Computacional no currículo de Matemática. **Educação e Matemática: Revista da Associação de Professores de Matemática**, [s. l.], n. 162, p. 5–10, 2021. Disponível em: https://em.apm.pt/index.php/em/article/view/2737. Acesso em: 30 out. 2023.

FERREIRA, G. F. **Por uma epistemologia da tecnologia na Educação Matemática**. 2020. Tese (Doutorado em Educação Matemática) – Universidade Estadual Paulista, Rio Claro. Disponível em: https://repositorio.unesp.br/items/fe8420ce-8266-48d1-9989-6c19fbcf235d. Acesso em: 17 mar. 2024.

KNUTH, D. E. Computer Science and Its Relation to Mathematics. **The American Mathematical Monthly**, v. 81, n. 4, p. 323–343, 1974.

LEITHOLD, L. **O Cálculo com Geometria Analítica. Vol. 1**. 3. ed. São Paulo: Harbra, 1994.

LEONTIEV, A. **O Desenvolvimento do Psiquismo**. Tradução: Rubens Eduardo Frias. 2. ed. São Paulo: Centauro, 2004.

LINS, R. C. Epistemologia, História e Educação Matemática: tornando mais sólidas as bases da pesquisa. **Revista de Educação Matemática**, São Paulo, n. 1, p. 75–91, 1993.

_____. Matemática, Monstros, Significados e Educação Matemática. In: BICUDO, M. A. V.; BORBA, M. de C. (Ed.). **Educação Matemática: Pesquisa em Movimento**. São Paulo: Cortez, 2004. P. 92–120.

_____. Modelo Teórico dos Campos Semânticos: uma análise epistemológica da álgebra e do pensamento algébrico. **Revista Dynamis**, Blumenau, v. 1, p. 29–39, 1994.

LINS, R. C. O Modelo dos Campos Semânticos: estabelecimentos e notas de teorizações. In: ANGELO, C. L. et al. (Ed.). **Modelo dos Campos Semânticos e Educação Matemática: 20 anos de história**. São Paulo: Midiograf, 2012. P. 11–30.

_____. Por que discutir teoria do conhecimento é relevante para a Educação Matemática. In: BICUDO, M. A. V. (Ed.). **Pesquisa em Educação Matemática: Concepções e perspectivas**. São Paulo: Editora da UNESP, 1999. P. 75–94.

LINS, R. C.; GIMENEZ, J. **Perspectivas em aritmética e álgebra para o século XXI**. Campinas: Papirus, 1997.

LIRIO, J. R.; PRADO, S. P. do. Contradições da BNCC acerca do desenvolvimento e uso das TDICs e do Pensamento Computacional. **Educação Matemática Sem Fronteiras: Pesquisas em Educação Matemática**, Brasil, v. 5, n. 1, p. 59–75, 2023. Disponível em: https://periodicos.uffs.edu.br/index.php/EMSF/article/view/13404. Acesso em: 21 mar. 2025.

LUCHETTA, V. J. O. **Uma Possível Produção de Significados para as Séries no Livro Elementos de Álgebra de Leonhard Euler**. 2017. Tese (Doutorado em Educação Matemática) – Universidade Estadual Paulista, Rio Claro. Disponível em: https://repositorio.unesp.br/handle/11449/152354. Acesso em: 24 set. 2024.

MORAES, M. G. **Pesquisas sobre Educação e Tecnologias: Questões emergentes e configuração de uma temática**. 2016. Tese (Doutorado em Educação) – Pontifícia Universidade Católica de Goiás, Goiás. Disponível em: http://tede2.pucgoias.edu.br:8080/handle/tede/3436. Acesso em: 10 nov. 2024.

PAPERT, S. **Mindstorms: Children, Computers, and Powerful Ideas**. New York: Basic Books, 1980.

PAULO, J. P. A. de. **Compreendendo Formação de Professores no Âmbito do Modelo dos Campos Semânticos**. 2020. Tese (Doutorado em Educação Matemática) – Universidade Estadual Paulista, Rio Claro. Disponível em: https://repositorio.unesp.br/items/3b90899d-a612-43c3-a5d6-711826f3852c. Acesso em: 29 out. 2023.

_____. Por um Método de Pesquisa em Educação Matemática Fundamentado no Modelo dos Campos Semânticos. **Revista de Educação, Ciências e Matemática**, [s. l.], v. 12, n. 3, p. 1–19, 2022. Disponível em:

https://publicacoes.unigranrio.edu.br/index.php/recm/article/view/6248. Acesso em: 4 nov. 2023.

PEIXOTO, J. Relações entre sujeitos sociais e objetos técnicos: uma reflexão necessária para investigar os processos educativos mediados por tecnologias. **Revista Brasileira de Educação**, Rio de Janeiro, v. 20, n. 61, p. 317–332, 2015. Disponível em: https://www.scielo.br/j/rbedu/a/hnpBTsy6vMXzmNjZzDtXCsq/abstract/?lang=pt. Acesso em: 30 out. 2024.

PEIXOTO, J.; MORAES, M. G. Education and Technologies: some Tendencies of this Tehmatic in Educational Research. **Revista Educativa - Revista de Educação**, Goiânia, Brasil, v. 20, n. 1, p. 233–252, 2017. Disponível em: https://seer.pucgoias.edu.br/index.php/educativa/article/view/5875. Acesso em: 22 mar. 2025.

PLEWKA, V. G.; DANTAS, S. C. Concepções acerca do Pensamento Computacional presentes na BNCC e do referencial curricular do Paraná no Ensino Médio. In: ANAIS do III Encontro Paranaense de tecnologia na Educação Matemática. Apucarana – PR: [s.n.], 2023.

PRADO, S. P. do. **Perspectivas sobre Pensamento Computacional construídas a partir do diálogo com educadores matemáticos**. 2024. Dissertação (Mestrado em Educação Matemática) – Universidade Estadual do Paraná, União da Vitória. Disponível em: https://prpgem.unespar.edu.br/dissertacoes/defesas-2024. Acesso em: 10 nov. 2024.

RAABE, A.; COUTO, N. E. R.; BLIKSTEIN, P. Diferentes abordagens para a computação na educação básica. In: RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. (Ed.). **Computação na Educação Básica: fundamentos e experiências**. Porto Alegre: Penso, 2020. P. 3–15.

REIS, S. R. dos; BARICHELLO, L.; MATHIAS, C. V. Novos conteúdos e novas habilidades para a área de Matemática e suas Tecnologias. **Revista Internacional de Pesquisa em Educação Matemática**, Brasília, v. 11, n. 1, p. 37–58, 2021. Disponível em: https://www.sbembrasil.org.br/periodicos/index.php/ripem/article/view/2539. Acesso em: 22 mar. 2025.

REZENDE, W. M. **O Ensino de Cálculo: Dificuldades de Natureza Epistemológica**. 2003. Tese (Doutorado em Educação) – Universidade de São Paulo, São Paulo.

RIBEIRO, L.; FOSS, L.; CAVALHEIRO, S. A. da. C. Entendendo o pensamento computacional. In: RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. (Ed.). **Computação na Educação Básica: fundamentos e experiências**. Porto Alegre: Penso, 2020. P. 16–30.

SBC, Sociedade Brasileira de Computação. **Nota Técnica da Sociedade Brasileira de Computação sobre a BNCC-EF e a BNCC-EM**. [S.l.], 2018. Disponível em: https://bit.ly/3uJmMAA. Acesso em: 20 set. 2023.

_____. **Referenciais de Formação em Computação: Educação Básica**. [S.l.], 2017. Disponível em: https://bit.ly/3qSqzur. Acesso em: 20 set. 2023.

TEDRE, M.; DENNING, P. J. The Long Quest for Computational Thinking. In: PROCEEDINGS of the 16th Koli Calling Conference on Computing Education Research. Koli, Finland: [s.n.], 2016. P. 120–129.

TEIXEIRA, F. O.; JUVANELLI, C.; DANTAS, S. C. Movimentos manifestados em processos de depuração por estudantes de um curso de pensamento computacional. **Revista de Produtos Educacionais e Pesquisas em Ensino**, Cornélio Procópio - PR, v. 8, n. 2, p. 1820–1847, 2024. Disponível em: https://seer.uenp.edu.br/index.php/reppe/article/view/1666. Acesso em: 27 out. 2024.

VALENTE, J. A. **A Espiral da Espiral de Aprendizagem: o processo de compreensão do papel das tecnologias de informação e comunicação na educação**. 2005. Tese (Livre docência) – Universidade Estadual de Campinas, São Paulo.

_____. Pensamento computacional, letramento computacional ou competência digital? Novos desafios da educação. **Revista educação e cultura contemporânea**, [s. l.], v. 16, n. 43, p. 147–168, 2019. Disponível em: http://periodicos.estacio.br/index.php/reeduc/article/viewArticle/5852. Acesso em: 30 out. 2023.

VALENTE, J. A. et al. Alan Turing tinha Pensamento Computacional? Reflexões sobre um campo em construção. **Tecnologias, Sociedade e Conhecimento**, Campinas, v. 4, n. 1, p. 7–22, 2017. Disponível em: https://econtents.bc.unicamp.br/inpec/index.php/tsc/article/view/14482. Acesso em: 30 out. 2023.

VARGAS, F. L. da S.; VICENTE, V. R. R. de; DANTAS, S. C. Matemática e suas tecnologias no Novo Ensino Médio: a dualidade presente nas habilidades específicas da Base Nacional Comum Curricular. **Em Teia | Revista de Educação Matemática e**

Tecnológica Iberoamericana, [s. l.], v. 13, n. 3, p. 359–384, 2022. Disponível em: https://periodicos.ufpe.br/revistas/index.php/emteia/article/view/254699. Acesso em: 21 mar. 2025.

VIEIRA, M. F. V.; CAMPOS, F. R.; RAABE, A. O legado de Papert e da linguagem Logo no Brasil. In: RAABE, A.; ZORZO, A. F.; BLIKSTEIN, P. (Ed.). **Computação na Educação Básica: fundamentos e experiências**. Porto Alegre: Penso, 2020. P. 49–63.

VIOLA DOS SANTOS, J. R.; PAULO, J. P. A. de; JULIO, R. Ensaio filosófico com um Modelo dos Campos Semânticos. **Revista de Educação Matemática**, [s. l.], v. 20, p. 1–28, 2023. Disponível em: https://www.revistasbemsp.com.br/index.php/REMat-SP/article/view/30. Acesso em: 3 nov. 2023.

WING, J. M. Computational thinking. **Communications of the ACM**, [s. l.], v. 49, n. 3, p. 33–35, 2006.

_____. Computational thinking and thinking about computing. **Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences**, [s. l.], v. 366, n. 1881, p. 3717–3725, 2008.

_____. **Computational thinking: What and Why?** Disponível em: https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why. Acesso em: 25 mar. 2024. 2011.

_____. Computational thinking's influence on research and education for all. **Italian Journal of Educational Technology**, [s. l.], v. 25, n. 2, p. 7–14, 2017. Disponível em: https://ijet.itd.cnr.it/index.php/td/article/view/922. Acesso em: 25 mar. 2024.