VI Workshop-Escola de Informática Teórica

17 a 19 de novembro de 2021

ANAIS

VI Workshop-Escola de Informática Teórica

Anais

Hélida Salles Santos
Ana Paula Lüdtke Ferreira
Renata Hax Sander Reiser
(Orgs.)

Organização

Apoio

Workshop-Escola de Informática Teórica (6. : 2021: Bagé, RS)
WEIT 2021 [recurso eletrônico] / Hélida Salles Santos [et. al.] (Orgs.). – Bagé: Universidade Federal do Pampa (UNIPAMPA), 2021.

200 p.:il.

Modo de acesso: <https://eventos.unipampa.edu.br/WEIT2021/anais/>
ISSN: 2236-9163
ISBN: 978-65-00-34950-4

Instituições promotoras: Universidade Federal do Pampa, Universidade Federal do Rio Grande, Universidade Federal de Pelotas, Universidade Federal do Rio Grande do Sul, Universidade de Passo Fundo, Universidade Federal de Santa Maria.

1. Informática teórica. 2. Lógica. 3. Métodos formais. 4. Matemática intervalar. 5. Modelos de computação. 6. Linguagens. 7. Aplicações. I. Ferreira, Ana Paula (coord.). II. Universidade Federal do Pampa. III. Universidade Federal do Rio Grande. IV. Universidade Federal de Pelotas. V. Universidade Federal do Rio Grande do Sul. VI. Universidade de Passo Fundo. VII. Universidade Federal de Santa Maria. VIII. Título. IX. Anais do WEIT 2021. X. Anais do Workshop-Escola de Informática Teórica.

CDU: 004

APRESENTAÇÃO

É com imenso prazer que apresentamos os Anais do WEIT 2021 (VI Workshop-Escola de Informática Teórica), promoção de um grupo de universidades gaúchas – Universidade Federal do Pampa (UNIPAMPA), Universidade Federal do Rio Grande (FURG), Universidade Federal de Pelotas (UFPEL), Universidade Federal do Rio Grande do Sul (UFRGS), Universidade de Passo Fundo (UPF) e Universidade Federal de Santa Maria (UFSM) – realizado em formato totalmente online entre 17 e 19 de novembro de 2021. Nesta edição o evento contou com o apoio do Instituto de Estudos Avançados da Universidade de São Paulo (IEA-USP) e da Sociedade Brasileira de Computação (SBC).

O evento tem como objetivo aumentar a participação de discentes, professores e pesquisadores nas áreas relacionadas à Informática Teórica, proporcionando um espaço de discussão qualificada com apresentação de trabalhos, abrindo fronteiras em oportunidades de pesquisa e desenvolvimento científico e tecnológico. O WEIT 2021, em sua sexta edição e primeira ocorrendo de forma totalmente online, selecionou 22 trabalhos técnicos, oriundos das mais diversas instituições do Rio Grande do Sul, do Brasil e do exterior. O WEIT 2021 contou, em sua programação, com duas palestras, um painel, cinco sessões técnicas de apresentação de trabalhos e o I Concurso de Teses e Dissertações do WEIT, dando visibilidade aos excelentes trabalhos de pós-graduação desenvolvidos nas áreas teóricas da Computação.

O Comitê de Programa foi formado por pesquisadores de todas as regiões do país. As revisões de artigos foram cuidadosas e formativas, no sentido de auxiliar o autor a identificar oportunidades de melhoria em sua pesquisa ou escrita. Os coordenadores do evento agradecem a todos que participaram do processo de avaliação, visto que sem essa valiosa contribuição o evento não teria sido possível.

Gostaríamos de agradecer aos nossos apoiadores: ao Instituto de Estudos Avançados da Universidade de São Paulo (IEA-USP) que nos cedeu a sala Zoom na qual o evento teve lugar e à Sociedade Brasileira de Computação (SBC), que passará a publicar os trabalhos do WEIT a partir desta edição. Agradecemos aos palestrantes e painelistas, que produziram material interessante de alta qualidade e que se disponibilizaram a compartilhá-lo conosco. Agradecemos também aos 112 participantes que se inscreveram no evento. Esperamos que tenham gostado e se motivado para ingressar em uma área fascinante da Computação.

Bagé, 19 de novembro de 2021.

Hélida Salles Santos
Coordenação do Comitê de Programa

Renata Hax Sander Reiser
Coordenação do CTD/WEIT

Ana Paula Lüdtke Ferreira
Organização Geral

COMITÊS

Comitê de Organização
Ana Paula Lüdtke Ferreira (UNIPAMPA, Coordenação Geral)
Carlos Amaral Hölbig (UPF)
Hélida Salles Santos (FURG)
Juliana Kaizer Vizzotto (UFSM)
Leila Ribeiro (UFRGS)
Luciana Foss (UFPEL)
Lúcio Mauro Duarte (UFRGS)
Renata Max Sander Reiser (UFPEL)
Rodrigo Machado (UFRGS)
Simone André da Costa Cavalheiro (UFPEL)

Coordenação do Comitê de Programa
Hélida Salles Santos (FURG)

Coordenação do Concurso de Teses e Dissertações
Renata Max Sander Reiser (UFPEL)

Comitê de Programa
Adriano Velasque Werhli (FURG)
Alice Fonseca Finger (UNIPAMPA)
Aline Brum Loreto (UFSM)
Ana Paula Lüdtke Ferreira (UNIPAMPA)
Annaxsuel Araújo (IFRN)
Antônia Jocivania Pinheiro (UFERSA)
Antônio Diego Silva Farias (UFERSA)
Benjamin Bedregal (UFRN)
Carlos Amaral Hölbig (UPF)
Diana Francisca Adamatti (FURG)
Éder Mateus Nunes Gonçalves (FURG)
Eduardo Nunes Borges (FURG)
Eduardo Piveta (UFSM)
Edward Hermann Haeusler (PUC-Rio)
Giancarlo Lucca (FURG)
Gleifer Alves Vaz (UTFPR)
Graçaliz Pereira Dimuro (FURG)
Ivan Mezzomo (UFERSA)
Leila Ribeiro (UFRGS)
Leonardo Ramos Emmendorfer (FURG)
Luciana Foss (UFPEL)

PROGRAMAÇÃO

Painel

INFORMÁTICA TEÓRICA COMO SUPORTE AO CONCEITO DE EXPLICABILIDADE EM INTELIGÊNCIA ARTIFICIAL

***Painelistas**:*

Graçaliz Pereira Dimuro
Universidade Federal do Rio Grande (FURG)
Professora Associada do Centro de Ciências Computacionais da Universidade Federal do Rio Grande (FURG), tem graduação em Engenharia Civil pela Universidade Católica de Pelotas (1980), mestrado em Ciências da Computação pela Universidade Federal do Rio Grande do Sul (1991) e doutorado em Ciências da Computação pela Universidade Federal do Rio Grande do Sul (1998). Atualmente é – FURG, docente dos Programas de Pós-Graduação em Computação (PPGCOMP) e de Modelagem Computacional (PPGMC) da FURG, avaliador de cursos de graduação do BASIS do Ministério da Educação e Pesquisador 1D do Conselho Nacional de Desenvolvimento Científico e Tecnológico. Atua nos seguintes temas: conjuntos e lógica fuzzy, sistemas fuzzy, sistemas de classificação baseados em regras fuzzy, classificação de grande volume de dados, regulação de trocas sociais em sistemas multiagentes, processos de tomada de decisão, teoria do jogos e do drama, simulação social e matemática intervalar.

Ricardo Matsumura de Araújo
Universidade Federal de Pelotas (UFPel)
Professor Associado no Centro de Desenvolvimento Tecnológico da Universidade Federal de Pelotas desde 2008, é doutor e mestre em Ciência da Computação pela UFRGS (2009), tendo tese defendida com louvor, e Engenheiro de Computação pela FURG. Bolsista de Produtividade Desen. Tec. e Extensão Inovadora do CNPq - Nível 2. Atuou como consultor pesquisador para empresas como Google e Myspace, realizando análise e mineração de dados sociais (social analytics). Teve projetos financiados, como pesquisador principal, em editais ARD FAPERGS (2010), Bolsa UOL Pesquisa (2013), Editais Universal CNPq (2013 e 2016) e Google Latin American Research Awards (2016), além de diversos outros como colaborador. Foi coordenador do Programa de Pós-Graduação em Computação (PPGC) da UFPel de 2015 a 2017. Atua nas áreas de Aprendizado de Máquina, Inteligência Artificial, Mineração de Dados e Ciência de Dados.

Cláudia Nalon
Universidade de Brasília (UnB)
Professora Associada na Universidade de Brasília, tem graduação em Ciência da Computação pela Universidade de Brasília (1993), mestrado em Ciência da Computação pela Universidade Estadual de Campinas (1997) e doutorado em Ciência da Computação pela University of Liverpool (2004). Já atuou como Coordenadora de Graduação do Bacharelado em Ciência da Computação e como Coordenadora de Extensão do Instituto de Ciências Exatas. Atua na área de Teoria da Computação, com interesse, principalmente, em métodos de prova baseados em resolução para combinações de lógicas não-clássicas.

Mediadora:

Leila Ribeiro
Universidade Federal do Rio Grande do Sul (UFRGS)
Professora Titular do Instituto de Informática da Universidade Federal do Rio Grande do Sul, tem graduação em Ciências da Computação pela Universidade Federal do Rio Grande do Sul (1988), mestrado em Ciências da Computação pela Universidade Federal do Rio Grande do Sul (1991) e doutorado em Informática pela Universidade Técnica de Berlim/Alemanha (1996) com tese na área de modelos semânticos para Computação. Fez estágio de pós-doutorado na Universidade de York, Inglaterra (2008/2009). Atua nas áreas dos Fundamentos da Computação, principalmente nas em Engenharia de Software (especificação e verificação formal), Bioinformática, Modelos de Computação e ensino de Computação (com ênfase em pensamento computacional). É a representante brasileira no IFIP Technical Committee 1 (Foundations of Computer Science) e membro efetivo da Sociedade Brasileira de Computação (SBC), onde atua como Diretora de Ensino de Computação na Educação Básica.

Palestra

DILEMAS ÉTICOS PARA A PESQUISA EM INTELIGÊNCIA ARTIFICIAL

Ana Cristina Bicharra Garcia
Universidade Federal do Estado do Rio de Janeiro (UNIRIO)

Professora Titular do Departamento de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO), pesquisadora CNPq e Ph.D. em Engenharia pela Stanford University. Ocupou os cargos de professora no departamento de Informática da Universidade Federal Fluminense (UFF) (1996-2017), fundadora e coordenadora do ADDLabs (1996-2017), pesquisadora visitante no MIT Sloan School of Management (2013-2014), no Stanford Center for Integrated Facility

Engineering (2002-2003) e na XEROX PARC (1991-1992) em Palo Alto, CA, EUA. É membro do conselho editorial do periódico *Digital Government: Research and Practice* e do *Frontiers in Neuroinformatics*. Desenvolveu cerca de 30 projetos de P&D em Inteligência Artificial, com desenvolvimento de cerca de 2 anos cada, em parceria com empresas, em especial da indústria de petróleo. Orientou 36 mestres e 11 doutores. Publicou cerca de 180 artigos em conferências e periódicos indexados. Possui sólida formação em inteligência artificial, inteligência coletiva e interação humano-computador, desenvolvendo pesquisas multidisciplinares aplicadas, estando profundamente empenhada em desenvolver pesquisas de IA que melhorem o bem-estar da sociedade. Seu interesse de pesquisa atual envolve os desafios éticos para o desenvolvimento Inteligência Artificial.

Palestra

O PODEROSO ENCONTRO DE TEORIA E PRÁTICA NA INDÚSTRIA

Marcel Vinícius Medeiros Oliveira
Universidade Federal do Rio Grande do Norte (UFRN)

Professor Associado do Departamento de Informática e Matemática Aplicada (DIMAp) da Universidade Federal do Rio Grande do Norte (UFRN), é Bacharel em Ciência da Computação pela Universidade Federal de Pernambuco (2000), Mestre em Ciência da Computação pela Universidade Federal de Pernambuco (2002) e Ph.D. em Ciência da Computação pela Universidade de York, Inglaterra (2006). Atualmente, é membro do Instituto Nacional de Engenharia de Software, membro do Comitê Especial de Métodos Formais da Sociedade Brasileira de Computação, membro do Colegiado da Pós-Graduação em Sistemas e Computação (PPgSC) da UFRN, e Coordenador de Cursos Técnicos do Instituto Metrópole Digital da UFRN com cerca de 2000 alunos. Marcel Oliveira tem experiência na área de Ciência da Computação, com ênfase em Métodos Formais. Mais especificamente, sua pesquisa tem focado em cálculo e táticas de refinamentos, concorrência, semântica de linguagens formais, integração de métodos formais e síntese de código a partir de especificações formais. Ele tem ensinado as disciplinas de Banco de Dados, Lógica Aplicada a Engenharia de Software e Métodos Formais.

PREMIAÇÕES

Melhor artigo com autoria de discente de graduação

A New Total Order for Triangular Fuzzy Numbers with an Application
Valentina Rosas, Benjamín Bedregal, José Canumán, Roberto Díaz, Edmundo Mansilla, Nicolás Zumelzu

Melhor artigo com autoria de discente de pós-graduação

An Application for Medical Diagnosis Using Correlation Coefficient With Modal Operators and Operator Identifying and Unary
Alex Bertei, Renata Reiser, Luciana Foss

Concurso de Teses e Dissertações

Tese – 1º lugar
An approach for consensual analysis on Typical Hesitant Fuzzy Sets via extended aggregations and fuzzy implications based on admissible orders
Mônica Lorea Matzenauer
Orientadora: Renata Hax Sander Reiser, Coorientadora: Hélida Salles Santos
Programa de Pós-Graduação em Computação, Universidade Federal de Pelotas

Tese – 2º lugar
n-Dimensional Fuzzy Implications: Analytical, Algebraic and Applicational Approaches
Rosana Medina Zanotelli
Orientadora: Renata Reiser, Coorientador: Benjamin Bedregal
Programa de Pós-Graduação em Computação, Universidade Federal de Pelotas

Dissertação – 1º lugar
Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos
Júlia Krüger Vieira
Orientadora: Luciana Foss, Coorientadora: Simone André da Costa Cavalheiro
Programa de Pós-Graduação em Computação, Universidade Federal de Pelotas

ARTIGOS

1 **A criação de jogos para o ensino de computação: uma análise comparativa**
Júlia Veiga da Silva, Braz Araujo Da Silva Junior, Luciana Foss, Simone André Da Costa Cavalheiro 1
2 **A Methodology for Opacity verification for Transactional Memory algorithms using Graph Transformation System**
Diogo J. Cardoso, Luciana Foss, André R. Du Bois 9
3 **A New Total Order for Triangular Fuzzy Numbers with an Application**
Valentina Rosas, Benjamín Bedregal, José Canumán, Roberto Díaz, Edmundo Mansilla, Nicolás Zumelzu 17
4 **A-Games: using game-like representation for representing finite automata**
Cleyton Slaviero, Edward Hermann Haeusler 25
5 **An Application for Medical Diagnosis Using Correlation Coefficient With Modal Operators and Operator Identifying and Unary**
Alex Bertei, Renata Reiser, Luciana Foss 33
6 **Aplicando Análise Consensual Fuzzy para Tomada de Decisão na Alocação de Recursos em Nuvem Computacionais**
Guilherme Bayer Schneider, Bruno Moura, Eduardo Monks, Adenauer Yamin, Helida Santos, Renata Reiser 41
7 **Compilação e Execução de código da linguagem QML no Computador Quântico da IBM**
João Gabriel da Cunha Schittler, Juliana Kaiser Vizzotto 49
8 **Consolidando a Modelagem do Jogo Aventura Espacial**
Yuri da Silva Rosa, Renata Reiser, Simone André Da Costa Cavalheiro, Luciana Foss, André Rauber Du Bois 54
9 **Description of Command and Control Networks in Coq**
Guilherme Gomes Felix da Silva, Edward Hermann Haeusler, Cláudia Nalon 60
10 **Desenvolvimento de uma Ferramenta para Teste Diferencial de Compiladores Usando Códigos Gerados Aleatoriamente**
Bruno S Coelho, Luiz Felipe Kraus, Samuel da Silva Feitosa 68
11 **Estudos sobre o algoritmo de Grover e sua Implementação**
Liliana S. Carmo, Gisele B. Freitas 73
12 ***Gramers*: Agentes Pedagógicos para uma plataforma de jogos baseada em Gramática de Grafos**
Júlia Veiga da Silva, Braz Araujo da Silva Junior, Luciana Foss, Simone André da Costa Cavalheiro 80
13 **Implicações Representáveis por Funções *Overlap* e *Grouping***
Cecília Botelho, Alessandra Galvão, Adenauer Yamin, Helida Santos, Renata Reiser 88
14 **On the problem of compensatory mating in animal breeding**
Ana Paula Lüdtke Ferreira 96
15 **Proposta de Atividades Lúdicas no Minecraft Educacional para a Promoção do Pensamento Computacional**
Jonnhy Moraes Marques, Luciana Foss, Simone André da Costa Cavalheiro 104

16 **Representabilidade de Operadores via Ordens Admissíveis**
Lidiane da Silva, Guilherme Schneider, Adenauer Yamin, Helida Santos, Benjamín Bedregal, Renata Reiser 112
17 **Sintaxe baseada em gramáticas de grafos para uma linguagem funcional visual voltada ao aprendizado de programação**
Marina Silva, Ana Paula Ludtke Ferreira 120
18 **Testando Mecanismos de Refatoração Utilizando Programas Gerados Aleatoriamente**
Luiz Felipe Kraus, Bruno Coelho, Samuel da Silva Feitosa 128
19 **Theoretical Computer Science in Basic Education: a Systematic Review**
Braz Araujo Da Silva Junior, Simone André da Costa Cavalheiro, Luciana Foss . . . 133
20 **Uma análise de técnicas de classificação para predição de valores atribuídos a jogadores no jogo FIFA**
Maurício Dorneles Caldeira Balboni, Helida Santos, Giancarlo Lucca 141
21 **Uma classificação de vinhos baseada em regras fuzzy utilizando o algoritmo FARC-HD**
Vitor M. Magalhães, Jair O. G. Carmona, Giancarlo Lucca, Helida Santos, Eduardo N. Borges 148
22 **Visualização e Extensão de um Verificador de Modelos para a ferramenta Verigraph-GUI**
Arthur L. Fuchs, Rodrigo Machado, Leila Ribeiro 156

CONCURSO DE TESES E DISSERTAÇÕES

1 **An approach for consensual analysis on Typical Hesitant Fuzzy Sets via extended aggregations and fuzzy implications based on admissible orders**
Mônica Lorea Matzenauer, Renata Reiser, Helida Santos 163
2 **n-Dimensional Fuzzy Implications: Analytical, Algebraic and Applicational Approaches**
Rosana Medina Zanotelli, Renata Reiser, Benjamin Bedregal 171
3 **Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos**
Júlia Krüger Vieira, Luciana Foss, Simone André da Costa Cavalheiro 179

A criação de jogos para o ensino de computação: uma análise comparativa*†

Júlia Veiga da Silva[1], Braz Araujo da Silva Junior[1], Luciana Foss[1], Simone André da Costa Cavalheiro[1]

[1]Laboratório de Fundamentos da Computação – Universidade Federal de Pelotas (UFPel)
CEP 96.010-610 – Pelotas – RS – Brasil

{jvsilva, badsjunior, lfoss, simone.costa}@inf.ufpel.edu.br

Abstract. *Considering the challenges of teaching computing and the use of digital games as one of the ways to help this theme, this paper presents a comparative analysis of three game creation platforms used in computer education: Greenfoot, Scratch and GrameStation. For the analysis, we define topics in common and discuss specific features of each tool. We conclude that the GrameStation platform, although based on a mathematical formalism, it has characteristics equivalent to the others and can be also used to introduce formal specification in computer education.*

Resumo. *Considerando os desafios do ensino de computação e a utilização de jogos digitais como maneira de auxiliar esse cenário, este artigo apresenta uma análise comparativa de três plataformas destinadas à criação de jogos, utilizadas no ensino de computação: Greenfoot, Scratch e GrameStation. Foram definidos, para a análise, tópicos comuns às ferramentas, bem como suas características individuais. A plataforma GrameStation, embora baseada em um formalismo matemático, possui características equivalentes às demais, sendo possível a discussão quanto a sua utilidade para a introdução da especificação formal no ensino de computação.*

1. Introdução

Sendo introduzida ao currículo escolar em diversos países, a computação provoca desafios aos educadores. Assim, o aprendizado de pedagogias apropriadas para o ensino de conteúdos como algoritmos, programação, Pensamento Computacional (PC), entre outros, torna-se fundamental [Sentance e Csizmadia 2017]. Neste cenário, os jogos digitais são comumente utilizados como recurso educacional, pois, interativos e visualmente atraentes, proporcionam diversão e estimulam a criatividade dos estudantes. Além disso, métodos de ensino centrados na criação de jogos digitais possuem um elevado potencial educacional, proporcionando, inclusive, o desenvolvimento de habilidades em STEM[1] [Wernbacher et al. 2020]. Egenfeldt-Nielsen (2010) evidencia três maneiras diferentes de aplicação de jogos na educação formal, sendo elas:

*Trabalho concluído.

†O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do MCTIC/CNPq (Rede Sacci), da FAPERGS, do CNPq, da SMED/Pelotas e da PREC e PRPPG/UFPel.

[1]Ciência, Tecnologia, Engenharia e Matemática (do inglês *Science, Technology, Engineering and Mathematics*).

1. Aprendizagem com o auxílio de jogos: refere-se à utilização de jogos com o objetivo de ensinar um determinado conteúdo. São jogos desenvolvidos, especificamente, para fins educacionais (jogos sérios).
2. Aprendizagem com jogos: refere-se à adaptação de jogos que não possuem fins didáticos, para serem utilizados durante o estudo de temas, conceitos, métodos etc.
3. Aprendizagem por meio da criação de jogos: refere-se ao desenvolvimento de um projeto, permitindo a sistematização do conhecimento quanto a um determinado tópico. Não é necessário que o jogo possua uma finalidade educativa. As competências, habilidades e conhecimento adquiridos durante seu desenvolvimento são mais importantes do que a própria finalidade.

Trabalhos relacionados apresentam levantamentos em relação aos recursos utilizados para o ensino de computação na educação. Doss et al. (2011) exibem uma pesquisa referente às principais plataformas de desenvolvimento de jogos utilizadas como ferramentas de ensino. Os aspectos técnicos – como tipo de interface, mecanismo de depuração, compartilhamento de projetos etc. – também são analisados e comparados. Já Brezolin e Silveira (2021), expõem um mapeamento sistemático – realizado a partir dos principais fóruns brasileiros promovidos pela Sociedade Brasileira de Computação – relativo às principais ferramentas tecnológicas utilizadas para o ensino de programação e desenvolvimento do PC. Destacando o terceiro tópico apontado por Egenfeldt-Nielsen (2010), este artigo apresenta uma análise comparativa de três plataformas utilizadas para o ensino de computação por meio da criação de jogos. Além disso, evidencia a importância da plataforma GrameStation – baseada em um formalismo matemático – neste contexto.

O artigo está organizado como segue. A Seção 2 conta com a metodologia abordada para a seleção das plataformas. A Seção 3 apresenta e descreve as plataformas selecionadas. A Seção 4 é destinada à análise dessas ferramentas. A Seção 5 discorre acerca das considerações finais e trabalhos futuros desta proposta.

2. Metodologia

A pesquisa foi realizada por meio do mecanismo Google Acadêmico, com foco em plataformas destinadas à criação de jogos para o ensino de computação. No processo de seleção, foram excluídos trabalhos que propunham a utilização de jogos para o ensino de computação, trabalhos cujas plataformas propostas não eram destinadas à criação de jogos e, por fim, trabalhos em que as plataformas propostas não permitiam a criação de projetos próprios. Por fim, a busca resultou em três plataformas: Greenfoot, Scratch e GrameStation.

Para a fase da análise, os critérios definidos foram divididos em duas categorias: características que normalmente são observadas em plataformas de desenvolvimento utilizadas na educação e características específicas, definidas a partir das plataformas selecionadas. Da primeira categoria, foram analisados quatro critérios: **(1)** Disponibilidade: as plataformas estão facilmente disponíveis (incluindo custo)? **(2)** Personalização: as plataformas permitem a importação de recursos externos (como imagens e sons), possibilitando a criação de jogos personalizados? **(3)** Compartilhamento: há a possibilidade de comunicação e compartilhamento de atividades entre os usuários? **(4)** Feedback imediato: as plataformas oferecem feedback visual imediato sobre as ações dos usuários?

Neste caso, feedback imediato refere-se à perspectiva WYSIWYG – acrônimo em inglês para "*What You See Is What You Get*" – ou seja, verifica se as ferramentas permitem a visualização em tempo real dos elementos a serem publicados ou impressos na tela, ainda durante a edição do documento.

Da segunda categoria, referente às características específicas, foram analisados seis critérios: **(1)** Conceitos introduzidos: quais conceitos da computação são introduzidos em cada uma dessas ferramentas? **(2)** Público-alvo: qual a idade indicada para a utilização das ferramentas? **(3)** Linguagem: que tipo de linguagem é utilizada para a criação dos jogos? **(4)** Amigabilidade: os usuários são capazes de descobrir as funcionalidades necessárias para atingir um determinado objetivo facilmente? **(5)** Conhecimento prévio: é necessário algum nível de conhecimento prévio a respeito do tema trabalhado pela plataforma antes de sua utilização? **(6)** Paradigma: qual o paradigma de programação utilizado pela plataforma?

3. Plataformas

Nesta seção são apresentadas as ferramentas selecionadas para a análise.

3.1. Greenfoot

Proposto por Henriksen e Kölling (2004), o Greenfoot[2] é um ambiente de desenvolvimento que visa apoiar o ensino e a aprendizagem do paradigma de Programação Orientada a Objetos (POO) a partir da criação de jogos. O ambiente possui um sistema integrado – ou seja, além da interface principal, dispõe de um editor, de um compilador e de um depurador – e utiliza a linguagem de programação Java para o desenvolvimento dos jogos. Além disso, conta com 9 classes nativas que facilitam o processo de criação: 1) Actor, classe que contém os métodos disponíveis para os atores do jogo; 2) Color, classe utilizada para o preenchimento de cores na tela; 3) Font, classe correspondente à fonte utilizada para os textos apresentados na tela; 4) Greenfoot, classe utilizada para a comunicação com o ambiente; 5) GreenfootImage, classe que contém os métodos de apresentação e manipulação de imagens; 6) GreenfootSound, classe que contém os métodos de apresentação e manipulação de sons; 7) MouseInfo, classe que contém os métodos de captura de eventos do mouse; 8) UserInfo, classe utilizada para o armazenamento permanente de dados em um servidor e para o compartilhamento desses dados entre diferentes usuários quando o cenário é executado no site da plataforma; e 9) World, classe responsável pela implementação dos métodos relacionados ao cenário do jogo.

O design do ambiente possibilita a introdução de conceitos fundamentais da POO, tradicionalmente difíceis de ensinar devido ao alto grau de abstração, de uma maneira facilmente compreensível, por meio de interações [Kölling 2010]. Entre os principais conceitos estão: 1) Programa, o qual consiste em um conjunto de classes; 2) Classes, das quais se podem criar objetos; 3) Criação de vários objetos a partir de uma classe; 4) Identificação de métodos e campos para objetos de uma mesma classe; 5) Estado individual de objetos dado pelos valores de seus campos; 6) Comunicação entre objetos por meio de seus métodos; 7) Métodos, que podem possuir parâmetros e retornar valores; e 8) Tipos, de parâmetros e de valores de retorno. Assim, as atividades dos alunos incluem: instanciar objetos, executar uma simulação, chamar interativamente métodos únicos, ler e

[2]Disponível em: https://www.greenfoot.org/.

modificar o código, compilar e criar subclasses de objetos de simulação etc. O Greenfoot tem o objetivo de apresentar conceitos antes da sintaxe. Para Kölling (2010), a sintaxe não deve ser o primeiro obstáculo a ser enfrentado pelos alunos antes de atingir o sucesso em seus projetos. No entanto, pela experiência de manipulação do código-fonte – ainda que posterior –, o Greenfoot é recomendado para um público-alvo a partir de 14 anos, também podendo ser utilizado no Ensino Superior.

De acordo com Kölling (2010), uma série de objetivos sustentam o design do ambiente. Tais objetivos podem ser resumidos em dois pontos, a partir de duas perspectivas diferentes: da perspectiva do aluno, o objetivo é tornar a programação envolvente, criativa e satisfatória; do ponto de vista do professor, o objetivo é que o ambiente auxilie ativamente no ensino de conceitos de programação importantes e universais.

3.2. Scratch

O Scratch[3] [Resnick et al. 2009] é uma linguagem de programação em blocos, disponível on-line, que possibilita a criação não só de jogos, mas também de animações e histórias por iniciantes, sem a necessidade de uma sintaxe tradicional. Um dos principais objetivos do Scratch é apresentar a programação àqueles sem experiências prévias, motivando o aprendizado de conceitos de uma maneira lúdica. Esse objetivo impulsionou diversos aspectos do design do Scratch, como a escolha de uma linguagem de blocos, do layout de interface e de um conjunto mínimo de comandos [Maloney et al. 2010]. A interface, por exemplo, é dividida em quatro painéis, de modo a garantir que os componentes principais estejam sempre visíveis: o primeiro painel refere-se aos comandos disponíveis, divididos em 8 categorias – entre elas, “Movimento”, “Controle”, “Som” e “Operadores”; o segundo painel apresenta os *scripts* relacionados ao *sprite*[4] selecionado no momento, bem como sons e imagens referentes a esse *sprite*; o terceiro painel é onde a ação acontece; e o quarto painel exibe as miniaturas de todos os *sprites* no projeto.

Os *scripts* são construídos a partir da conexão de blocos que representam instruções, expressões e estruturas de controle. As formas dos blocos sugerem seu encaixe e o sistema de arrastar e soltar se recusa a conectar blocos de formas incoerentes. Com isso, a gramática visual das formas dos blocos e suas regras de combinação desempenham o papel de sintaxe em uma linguagem baseada em texto. Conforme Maloney (2010), a linguagem de blocos elimina a possibilidade de erros de sintaxe, permitindo que os usuários se concentrem em problemas de interesse imediato ao invés de se empenhar em apenas fazer seu programa compilar. Além disso, o sistema está sempre ativo: não há etapa de compilação ou alternância entre a execução e a edição do programa. Desse modo, para o autor, o feedback visual enquanto os blocos são arrastados auxilia na compreensão da organização de programas e utilização de diferentes tipos de dados.

3.3. GrameStation

O GrameStation[5] é um motor de jogos baseado em Gramática de Grafos (GG). A GG é um formalismo matemático para especificar e verificar sistemas. Formalmente é definida por um grafo tipo, que diferencia e restringe os elementos (vértices e arestas) permitidos

[3]Disponível em: https://scratch.mit.edu/.

[4]No Scratch, *sprites* são objetos que executam funções controladas por *scripts* (são os personagens e objetos dos projetos).

[5]Disponível em: https://wp.ufpel.edu.br/pensamentocomputacional/gramestation-pt/.

em um sistema; um grafo inicial (composto por elementos de tipos definidos pelo grafo tipo), que especifica o estado inicial do sistema; e um conjunto de regras de transformação de grafos, que representam os possíveis comportamentos do sistema. Dessa forma, é possível realizar a simulação de um sistema a partir da especificação de uma GG, verificando estados alcançáveis e detectando e evitando estados indesejados. Silva Junior (2020) relaciona características das GGs a diversas habilidades do PC, como análise e representação de dados, decomposição de problemas, abstração, algoritmos e processos, simulação e paralelismo.

No GrameStation, a especificação de um jogo corresponde à especificação de uma GG, com a definição de grafos e regras. Sendo assim, o motor é estruturado em módulos: Game Tutorial, onde o usuário pode aprender a utilizar a plataforma; Game Builder, onde o usuário pode construir seus jogos especificando as GGs; e Game Player, onde o usuário pode executar seus jogos simulando as GGs, isto é, selecionando regras e aplicando-as a partir do grafo inicial.

4. Análise das Plataformas

Ainda que possuam focos distintos, as três plataformas utilizam a criação de jogos como maneira de estimular o desenvolvimento de habilidades relacionadas à computação. Greenfoot e Scratch, por exemplo, são indicados para atividades relacionadas ao ensino de programação. Scratch, no entanto, conta com uma abordagem de introdução a lógica, enquanto Greenfoot faz uso da POO, ou seja, é indicado a quem já possui conhecimentos acerca do desenvolvimento de programas. Dessa forma, por visar usuários mais jovens e focar na aprendizagem autodirigida, enfatizando a facilidade da manipulação, o Scratch pode ser utilizado como uma introdução ao Greenfoot [Maloney et al. 2010].

Com relação a características comumente observadas em tecnologias voltadas à educação, Greenfoot, Scratch e GrameStation são consideravelmente equivalentes. A Tabela 1 ilustra os quatro aspectos considerados. Relativo ao primeiro item, as três plataformas encontram-se disponíveis on-line e de forma gratuita. A importação de recursos externos também é permitida, proporcionando a criação de jogos personalizados. Referente ao compartilhamento de projetos, apenas Greenfoot e Scratch contam com esse recurso, permitindo o compartilhamento em seus sites. Por último, o feedback imediato está presente em todas as plataformas.

Tabela 1. Características comuns das plataformas

	Greenfoot	**Scratch**	**GrameStation**
Disponibilidade	✓	✓	✓
Personalização	✓	✓	✓
Compartilhamento	✓	✓	×
Feedback imediato	✓	✓	✓

Conforme ilustrado na Tabela 2, os conceitos introduzidos pelas plataformas são: Programação Orientada a Objetos (Greenfoot), Lógica de Programação (Scratch) e Gramática de Grafos (GrameStation). Informações relativas aos conceitos trabalhados em Greenfoot e Scratch foram identificadas em [Henriksen e Kölling 2004] e [Resnick et al. 2009], respectivamente, também podendo ser encontradas em seus sites

oficiais. Sobre o público-alvo, não foram encontradas informações referentes ao GrameStation. Greenfoot e Scratch, de acordo com informações encontradas em seus sites, são recomendados para usuários a partir de 14 e 8 anos, respectivamente. Scratch ainda ressalta, em seu site, a linguagem ScratchJr[6], projetada para crianças entre 5 e 7 anos. No que se refere às linguagens, Greenfoot utiliza Java, o que, para os autores, produz efeitos em diversos aspectos da plataforma – enquanto alguns são benéficos, outros impõem limitações e complexidades. O domínio sobre uma linguagem textual é, em geral, consideravelmente mais complexo para usuários jovens, o que estabelece um nível mínimo de maturidade para sua manipulação. Scratch faz uso de blocos que podem ser arrastados pelo usuário. Esses blocos possuem conectores que sugerem sua organização. De acordo com os autores, tal mecanismo facilita o processo de criação de projetos por crianças. O GrameStation, por sua vez, utiliza grafos. Apesar de visual, a linguagem pode oferecer certo grau de dificuldade na especificação de jogos àqueles que não estão habituados a esse formalismo.

Tabela 2. Características específicas das plataformas

	Greenfoot	**Scratch**	**GrameStation**
Conceitos introduzidos	POO	Lógica de Programação	GG
Público-alvo	14 anos+	8 anos+	Não determinado
Linguagem	Java	Blocos	Grafos
Amigabilidade	Sim	Sim	Parcial
Conhecimento prévio	Orientação a Objetos	Programação Procedural	GG
Paradigma	Imperativo (Orientado a Objetos)	Imperativo (Orientado a Eventos)	Declarativo

Com relação ao quarto tópico, Greenfoot é considerado amigável pois, para os autores, mesmo fazendo uso de uma linguagem de programação padrão, a plataforma exclui diversos mecanismos encontrados em IDEs genéricas, como controle de versão, teste de unidade e refatoração [Henriksen e Kölling 2004]. Henriksen e Kölling (2004) ressaltam que o objetivo é promover aos alunos a adaptação completa com a interface em poucos dias. A fim de facilitar a experiência dos usuários com o Scratch, três princípios básicos de design foram estabelecidos: torná-lo mais manipulável, mais significativo e mais social em relação a outros ambientes de programação [Resnick et al. 2009]. A manipulação é obtida por meio da programação em blocos: mesmo que os alunos iniciem, simplesmente, conectando as estruturas em diferentes sequências e combinações e verificando o resultado, os blocos são moldados apenas para encaixarem-se de formas que façam sentido sintático. Para torná-lo mais significativo, dois critérios de design foram evidenciados: a diversidade, com a possibilidade de elaborar diferentes tipos de projetos (histórias, jogos e animações); e a personalização, visto que os usuários podem customizar seus projetos por meio da importação de recursos. Por último, a opção de compartilhamento das atividades foi estabelecida a fim de tornar a ferramenta mais social. Segundo os autores, para muitos usuários, a oportunidade de apresentar seus projetos a um grande público, bem como receber feedback e conselhos de outros usuários é um forte fator de motivação. Por fim, o GrameStation possui amigabilidade parcial pois, ainda que intuitivo, a especificação de uma GG pode não ser trivial a usuários que não possuem conhecimentos prévios.

O quinto tópico refere-se aos conhecimentos prévios necessários para a criação e

[6]Disponível em: https://www.scratchjr.org/.

o desenvolvimento de jogos nas plataformas. Greenfoot trabalha o paradigma de POO e, sendo assim, um conhecimento prévio a respeito desse assunto torna-se necessário. Para o Scratch, são necessários conhecimentos prévios sobre programação procedural, devido a manipulação de estruturas de sequência, decisão e iteração durante o desenvolvimento dos projetos. Para o GrameStation, é necessário conhecimento prévio em GG, uma vez que a especificação dos jogos na plataforma é baseada nesse formalismo.

Quanto ao último tópico, Greenfoot possui paradigma imperativo (orientado a objetos), trabalhando conceitos como objeto, herança, polimorfismo, entre outros; Scratch possui paradigma imperativo (orientado a eventos), trabalhando conceitos como ciclos, loops, entre outros; e GrameStation possui paradigma declarativo, trabalhando conceitos como grafos, regras de transformação de grafos, *match*, entre outros.

5. Considerações Finais

Este artigo apresenta a análise comparativa de três plataformas destinadas à criação de jogos utilizadas no ensino de computação: Greenfoot, Scratch e GrameStation. Embora possuam objetivos de aprendizagem distintos, as plataformas contam com diversas características semelhantes. Com base na análise, foi observada uma tendência em lidar/facilitar a questão da sintaxe em ambientes de desenvolvimento – reconhecida como um empecilho no aprendizado. A plataforma Greenfoot, por exemplo, é indicada para usuários acima de 14 anos pois, em determinada etapa da criação, é preciso lidar com a sintaxe da linguagem.

A plataforma GrameStation, embora baseada em um formalismo matemático – uma linguagem que, apesar de intuitiva, pode não ser trivial àqueles que não estão habituados –, possui diversas características comuns às demais plataformas. Tal fato pode ser interpretado como um esforço para a introdução da especificação formal na educação em computação. Segundo Silva Junior (2019), o processo para o desenvolvimento de conhecimentos associados à GG pode ser facilitado por meio da interatividade e de recursos visuais. Neste sentido, a plataforma apoia-se em recursos frequentemente defendidos na literatura, como o uso de linguagens visuais para facilitar a sintaxe e ambientes relacionados a jogos para manter a motivação. Como trabalho futuro, pretende-se expandir a presente análise, considerando plataformas que objetivam o aprendizado de demais conceitos da computação – seja por meio do desenvolvimento de jogos ou não – e, também, plataformas especificamente voltadas para o ensino/introdução de especificações formais.

Referências

Brezolin, C. V. S. e Silveira, M. S. (2021). Panorama Brasileiro de Uso de Ferramentas para Desenvolvimento do Pensamento Computacional e Ensino de Programação. In *Workshop sobre Educação em Computação (WEI)*, pages 398–407.

Doss, K., Juarez, V., Vincent, D., Doerschuk, P., e Liu, J. (2011). Work in Progress — A Survey of Popular Game Creation Platforms Used for Computing Education. In *Frontiers in Education Conference (FIE)*, pages F1H–1–F1H–2.

Egenfeldt-Nielsen, S. (2010). The Challenges to Diffusion of Educational Computer Games. *Leading Issues in Games Based Learning*.

Henriksen, P. e Kölling, M. (2004). Greenfoot: Combining Object Visualisation with Interaction. page 73–82. Association for Computing Machinery (ACM).

Kölling, M. (2010). The Greenfoot Programming Environment. Association for Computing Machinery (ACM).

Maloney, J., Resnick, M., Rusk, N., Silverman, B., e Eastmond, E. (2010). The Scratch Programming Language and Environment. *Transactions on Computing Education*, 10(4).

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., e Kafai, Y. (2009). Scratch: Programming for All. *Communications of the ACM*, 52(11):60–67.

Sentance, S. e Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 22(2):469–495.

Silva Junior, B. A. (2020). GGasCT: Bringing Formal Methods to the Computational Thinking. Master's thesis, Universidade Federal de Pelotas, Pelotas, RS, Brasil.

Silva Junior, B. A., Cavalheiro, S. A. C., e Foss, L. (2019). Métodos Formais na Educação Básica: Operando Gramáticas de Grafos em um Jogo Educacional. *Workshop-Escola de Informática Teórica*, pages 178–186.

Wernbacher, T., Reuter, R., Denk, N., Pfeiffer, A., König, N., Fellnhofer, K., Grixti, A., Bezzina, S., e Jannot, E. (2020). Create Digital Games for Education: Game Design as a Teaching Methodology. In *Proceedings of ICERI2020 Conference*, volume 9.

A Methodology for Opacity verification for Transactional Memory algorithms using Graph Transformation System*†

Diogo J. Cardoso[1], **Luciana Foss**[1], **Andre R. Du Bois**[1]

[1]Programa de Pós-Graduação em Computação - CDTec
Universidade Federal de Pelotas (UFPel) – Pelotas, RS – Brasil

{diogo.jcardoso,lfoss,dubois}@inf.ufpel.edu.br

Abstract. *With the constant research and development of Transactional Memory (TM) systems, various algorithms have been proposed, and their correctness is always an important aspect to take into account. When analyzing TM algorithms, one of the most commonly used correctness criterion is opacity, which infers that executions only observe consistent states of the shared memory. This paper proposes a formal definition to demonstrate that a given TM algorithm only generates opaque histories using a Graph Transformation System. The methodology introduced consists of translating an algorithm into production rules that manipulate the state of a graph. The proposed approach has demonstrated capability to deal with some of the complexity of TM algorithms and a case study has shown the working proof of opacity of the algorithm in question.*

Resumo. *Com a constante pesquisa e desenvolvimento de sistemas de Memória Transacional (TM), vários algoritmos têm sido propostos, e sua corretude é sempre um aspecto importante a se levar em consideração. Ao analisar algoritmos de TM, um dos critérios de corretude mais comumente usados é opacidade, que infere que execuções só observam estados consistentes da memória compartilhada. Este artigo propõe uma definição formal para demonstrar que um determinado algoritmo TM só gera histórias opacas usando um Sistema de Transformação de Grafos. Uma metodologia é introduzida para traduzir um algoritmo para regras de produção que manipulam o estado de um grafo. A abordagem proposta demonstrou a capacidade de lidar com algumas das complexidades de algoritmos TM e um caso de estudo mostra o funcionamento da prova de opacidade do algoritmo em questão.*

1. Introduction

To this day, the research and development of advances in multiprocessor programming still try to leverage processing power of multi-core systems. The synchronization of shared memory accesses such that safety and liveness properties are preserved is an inherent challenge associated with multiprocessor algorithms. Safety ensures that an algorithm is correct with respect to a defined correctness condition while liveness ensures that the program threads terminate according to a defined progress guarantee [Peterson and Dechev 2017].

*Work in progress.

†This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

Transactional Memory (TM) provides a high level concurrency control abstraction. At language level, TM allows programmers to define certain blocks of code to run atomically [Herlihy and Moss 1993], without having to define *how* to make them atomic. Also, at implementation level, TM assumes that all transactions are mutually independent, therefore it only retries an execution in the case of conflicts. There are benefits of using TM over lock based systems, such as, composability [Harris et al. 2005], scalability, robustness [Wamhoff et al. 2010] and increase in productivity [Pankratius and Adl-Tabatabai 2011]. There are several proposals of implementations of TM: exclusively Software [Shavit and Touitou 1997], supported by Hardware [Herlihy and Moss 1993], or even hybrid approaches [Matveev and Shavit 2015].

TM allows developing programs and reasoning about their correctness as if each atomic block executes a *transaction*, atomically and without interleaving with other blocks, even though in reality the blocks can be executed concurrently. The TM runtime is responsible to ensure correct management of shared state, therefore, correctness of TM clients depends on a correct implementation of TM algorithms [Khyzha et al. 2018]. A definition of what correctness is for TM becomes necessary when defining a correct implementation of TM algorithms. Intuitively, a correct TM algorithm should guarantee that every execution of an arbitrary set of transactions is indistinguishable from a sequential run of the same set. Several correctness criteria were proposed in the literature [Guerraoui and Kapalka 2008, Lesani and Palsberg 2014] and they rely on the concept of transactional histories.

Recent works on formal definitions for TM focuses on consistency conditions [Khyzha et al. 2018, Bushkov et al. 2018], fault-tolerance [Marić 2017], and scalability [Peluso et al. 2015]. Of the several correctness criteria proposed for TM, opacity is very well defined and known. Opacity and its sub-classes use a graph characterization composed of a conflict graph that represents how the transactions relate to each other by some defined notion of conflict. A history, sequence of transactional events that have access to a shared memory, is considered correct if the conflict graph of its transactions presents no cycles.

This work aims to propose a methodology to define a Graph Transformation System (GTS) that represents a TM algorithm and demonstrate that, considering the notion of conflict introduced by [Guerraoui and Kapalka 2008], the algorithm only generates "correct" histories. As an initial result and proof of concept, a GTS was constructed and demonstrated that from a single history execution it is possible to generate its conflict graph and evaluate the correctness of the history [Cardoso et al. 2019]. The main goal of this work is to expand that idea for an entire TM algorithm that generated such history. Meaning that for a set of transactions, the aim of the methodology is to show that every execution observes a correct state of the shared memory. The main contribution of this work is a methodology to formalize complex TM algorithms and its safety property check. This approach is capable of dealing with different characteristics of TM algorithms, in terms of versioning and conflict detection, which can make it a powerful tool for their correctness verification and also a formalization that can possibly be extended to new or existing graph characterization of other safety properties. A case study of a complex algorithm that deals with partial rollbacks was explored in [Cardoso et al. 2021].

2. Background

This chapter briefly describes some of the background knowledge necessary for the rest of this work. For space reasons not all background explanations along with the case study are present in this text, however, they can be found in a public repository[1].

2.1. Transactional Memory

Transactional Memory (TM) borrows the abstraction of atomic transaction from the data base literature and uses it as a first-class abstraction in the context of generic parallel programs. TM only requires of the programmer the identification of which blocks of code must be executed in a atomic way, but not how the atomicity must be achieved. TM has been shown as an efficient way of simplifying concurrent application development. Besides being a simple abstraction, TM also demonstrates an equal performance (or even better) than refined and complex lock mechanisms.

Transactional memory enables processes to communicate and synchronize by executing *transactions*. A transaction is a sequence of actions that appears indivisible and instantaneous to an outside observer. Any number of operations on *transactional objects* (*t-objects*) can be issued, and the transaction can either *commit* or *abort*. When a transaction T commits, all its operations appear as if they were executed instantaneously (atomically). However, when T aborts, all its operations are rolled back, and their effects are not visible to any other transactions [Guerraoui and Kapałka 2010].

A TM can be implemented as a shared object with operations that allow processes to control transactions. The transactions, as well as t-objects, are then "hidden" inside the TM. Conflict detection between concurrent transactions may be eager, if a conflict is detected the first time a transaction accesses a t-object, or lazy when the detection only occurs at commit time. When using eager conflict detection, a transaction must acquire ownership of the value to use it, hence preventing other transactions to access it, which is also called pessimistic concurrency control. With optimistic concurrency control, ownership acquisition and validation only occurs when committing.

To realize operations in shared objects the TM algorithm provides implementations of read, write, commit and abort procedures. These procedures are called transactional operations. A history H of a transactional memory contains a sequence of transactional operations calls.

2.2. Opacity

There are two important characteristics of the safety property for TM implementations: (1) transactions that commit must result in a total order consistent with a sequential execution; (2) it is desired that even transactions that abort have access to a consistent state of the system (resulted from a sequential execution).

The opacity correctness criterion was firstly introduced by [Guerraoui and Kapalka 2008] with the purpose of dealing with these two characteristics. In an informal way, opacity requires the existence of a total order for all transactions (that committed or aborted). This total order is equivalent to a sequential execution where only committed transactions make updates.

[1]Full text and case study explanations can be found in `https://github.com/diogocrds/tmgts`.

Definition 1 (Graph characterization of Opacity)**.** *Let H be any TM history with unique writes and $V_{\ll}$ any version order function in H. Denote $V_{\ll}(x)$ by $\ll_x$. The directed, labelled graph OPG(H, $V_{\ll}$) is constructed in the following way:*

1. *For every transaction T_i in H (including T_0) there is a vertex T_i in graph OPG(H, $V_{\ll}$). Vertex T_i is labelled as follows: vis if T_i is committed in H or if some transaction performs a read operation on a variable written by T_i in H, and loc, otherwise.*
2. *For all vertices T_i and T_k in graph OPG(H, $V_{\ll}$), $i \neq k$, there is an edge from T_i to T_k (denoted $T_i \rightarrow T_k$) in any of the following cases:*
   (a) *If $T_i \prec_H T_k$ (i.e., T_i precedes T_k in H); then the edge is labelled rt (from "real-time") and denoted $T_i \xrightarrow{rt} T_k$;*
   (b) *If T_k reads from T_i, meaning that T_i writes to the variable before T_k reads it; then the edge is labelled rf and denoted $T_i \xrightarrow{rf} T_k$;*
   (c) *If, for some variable x, $T_i \ll_x T_k$; then the edge is labelled ww (from "write before write") and denoted $T_i \xrightarrow{ww} T_k$;*
   (d) *If vertex T_k is labelled vis, and there is a transaction T_m in H and a variable x, such that: (i) $T_m \ll_x T_k$, and (ii) T_i reads x from T_m; then the edge is labelled rw (from "read before write") and denoted $T_i \xrightarrow{rw} T_k$;*

Theorem 1 (Graph characterization of Opacity [Guerraoui and Kapałka 2010])**.** *Any history H with unique writes is final-state opaque if, and only if, exists a version order function $V_{\ll}$ in H such that the graph OPG(H,$V_{\ll}$) is acyclic.*

Proof. Proof can be found in [Guerraoui and Kapałka 2010]. □

3. Translation to GTS

The first step of the proposed methodology is the translation of the logic of the algorithm to production rules, this step is made manually by analysing the procedures defined in the algorithm to create the state graphs desired. First, a representation of sequential operations is needed that will compose transactions and histories used in the entire approach. If `x` is any variable, # is any number/value, transactions can have the following operations: `begin`, `write(x,#)`, `read(x)` and `tryCommit`. Conflict comes form two or more transactions executing operations on the same variable.

Figure 1 demonstrates how a transaction is represented in a graph manner. This is the **initial state** as an input to the GTS. Each operation (`begin`, `read`, `write` and `tryCommit`) is represented by a node with relevant information to the operation itself, the main node `T` represents the identifier for the transaction with an unique id. Sequential operations are connected by an directed edge `next` that represents the order in which these operations must execute. The edge `op` represents the current operation to be executed, in a approach like this every transactional operation is considered to be atomic: the response of the operation happens immediately after the invocation. In a previous work, a GTS formalism for histories where the invocations and responses are processed separately [Cardoso et al. 2019] was explored. However, because some of the correctness criteria for the TM algorithms use atomic operations, it was also chosen to be used in this approach, which in turns decreases the number of nodes in a transaction or history, making it more readable.

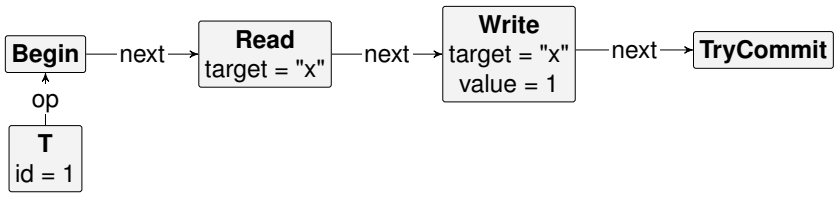


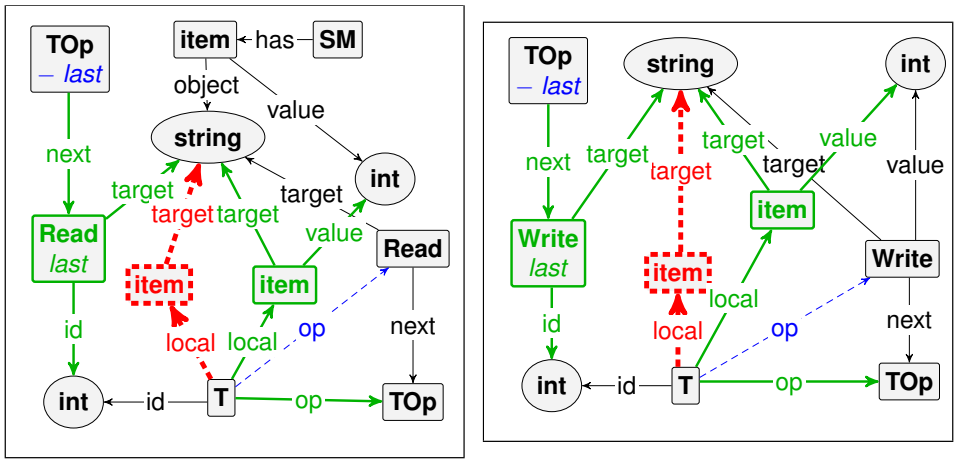
Figure 1. Graph representation of a transaction as an input for the GTS.

The initial state of the GTS includes the transactions (as seen in Figure 1) and some global objects like the shared memory, global clock, list of active transactions and so on. Which objects are treated globally or locally will depend on the algorithm itself. Having these global objects from the start allows the algorithm's logic to access them at any moment. Restriction on this access is enforced by the transformation rules. Another important aspect of the GTS formalism is the type graph. This special graph will determine what nodes and edges can exist in the system, this results in a controlled behavior by the production rules.

3.1. TM Procedures

The next step in formalizing a TM algorithm with a GTS is dealing with the procedures of the algorithm. The main procedures used are: begin, read, write, commit and abort. Some algorithm might have extra procedures such as a rollback or verify operation.

The first operations we describe are **read operation** and **write operation**. TM procedures can demonstrate different approaches in terms of versioning (eager or lazy) and conflict detection (also eager or lazy).



(a) Lazy versioning Read

(b) Lazy versioning Write

Figure 2. Example of production rules for lazy versioning Read and Write Operations.

In Figure 2 it is showcased a lazy versioning read and write operation. These two examples manipulate the values of the shared memory (reading or writing a specific variable) and create a new object in the last position of the history.

The **begin operation**, that starts a transaction has two situations where it occurs: either it is the first operation of the entire system, therefore the history is empty; or the transaction is starting in the middle of the execution where the history is no longer empty. To accommodate for both situations two separate production rules are needed. The last two TM operations of the GTS are **commit** and **abort**. These rules are executed only when the transaction can in fact commit, otherwise an abort production rule would execute and deal with rollback, which can be specially difficult in an eager versioning algorithm.

Note that so far only versioning was covered in the production rules, but conflict detection is also an important characteristic to take into consideration when designing a TM algorithm so it will be reflected in the GTS. In an algorithm with eager conflict detection, at any point if a conflict happens some transaction is likely to be aborted. This can approached by always checking the version of a variable read by the transaction. A *local* node `conflict` stores the value read of each variable the transaction performed a read operation on, this can be used as validation that the transaction has observed a stable state of the system. This verification can be read as: for all local nodes `conflict` that store a value read (valRead-edge), their respective objects in the shared memory (`SM` node) must not have a different value. While the verification of conflict in a commit, read or write operation ensures that all values read are stable, the abort operation that uses the quantifier *exists* ($\exists$), can be triggered with at least one conflict. Both instances of verification are mutually exclusive, a transaction cannot commit (or read or write for that matter) and abort at the same time.

4. Generating Histories

After translating the algorithm to production rules that correctly modify the state of the system and makes the decision of committing or aborting a transaction, the next step is deal with all possible sequences of operations that generate different histories. Because production rules are being used as a one-step operation that state of the graph, it is possible to use the Labelled Transition System (LTS) Simulation tool that GROOVE offers. Given the initial state of three transactions similar as seen in Figure 1 (in addition of the global nodes `History` and `SM`) the simulation of a *lazy-versioning* and *eager-conflict* algorithm will generate a LTS with 231 states where 70 of those are called "final". In GROOVE the LTS is visualized with a tree-like graph.

Each node in the LTS can be expanded (by clicking on it) to visualize the current state of the system resulted from the sequence of production rules applied to that particular state up to that point. At the top of the LTS the initial state is labelled *start* and at the bottom the final states as green nodes labelled *result*. A final state just means that no more production rules can be applied to that state, this means that there are no more transactional operations left to be executed and the history generated by that sequence of operations is complete. Another feature of GROOVE that can be applied to the generated LTS is the use of Computation Tree Logic (CTL). CTL allows for the verification of a properties in the graphs states in the LTS by using a special production rule called *graph condition* (a production rule that does not create or delete elements).

5. Correctness Criterion

The third, and final, step of the proposed methodology is to observe the correctness of the algorithm being evaluated. The correctness criteria used is mainly based on the graph

representation of Opacity introduced by [Guerraoui and Kapałka 2010]. This graph characterization is dependant on a predetermined set of conflicts, and the conflict graph itself is created via the verification of which of these conflicts can be observed between transactions in a history. The LTS is used to explore every combination of transactional operations therefore exploring all possible histories, now the next step is to analyse each of these histories and create their respective conflict graph.

Using the same principles applied in a previous paper [Cardoso et al. 2019], where the opacity of a single history was observed using a GTS, now the proposed approach was able to analyse an entire set of histories using the LTS constructed above. The process of creating a conflict graph can be separated from the creation of the history itself. Moreover, because it deals with already existing data it only needs to observe the set of conflicts and modify edges between `T`-nodes, that represent each transaction, which results in very simple production rules.

Lastly, now that the histories were generated, and from each one a conflict graph was extracted, the LTS is complete allowing the use of the Computation Tree Logic (CTL) tool in GROOVE to check for acyclicity of these conflict graphs. With some auxiliary production rules to path through the conflict graphs a condition is looked for where: following the direction of the edges results in a path that goes back to a node that was already visited. The CTL check consists of a pattern match test of the special production rules named *graph condition*. The verification is made by the formula: `AG !cyclic`, which means that, for every path following the current state, the entire path holds the property of not pattern matching the graph condition `cyclic`. If the formula holds for all states in the LTS, the condition of acyclicity is true and the algorithm only generated opaque histories.

6. Conclusions

In this work we presented an approach to verify opacity as a correctness criterion for transactional memory algorithms via a translation to a Graph Transformation System (GTS). We used a graph characterization of opacity [Guerraoui and Kapałka 2010] well known in the literature to demonstrate opacity to every history the algorithm generates for a given entry program. For this approach it was necessary to represent sequential operations which compose the initial state (set of transactions) and the histories. We used production rules that "execute" a full transactional operation in a single step. The Labelled Transition System (LTS) generated by the GROOVE tool represents all possible sequences of rule applications. The leafs of this tree contain a full history, thus the set of all possible rule applications represents the set of all possible histories the current algorithm generates from a set of transactions given in the initial state.

We extended a previous work, that evaluated a single history to observe its opacity using a GTS, to now prove the opacity to all the histories generated by the translated algorithm. For future work, we want to explore a more concrete initial state as input for the correctness test against an algorithm. This introduces the complexity of dealing with loops and more complex data structures other than single-value t-objects. We also have in mind the use of other TM algorithms with different approaches to detecting and dealing with conflict/versioning. Another important point that we want to address is the choice of correctness criteria, opacity was an easy answer because it already has a graph

characterization, however we want to explore other safety properties and possibly liveness properties too.

References

Bushkov, V., Dziuma, D., Fatourou, P., and Guerraoui, R. (2018). The pcl theorem: Transactions cannot be parallel, consistent, and live. *Journal of the ACM (JACM)*, 66.

Cardoso, D., Foss, L., and Du Bois, A. (2021). A graph transformation system formalism for correctness of transactional memory algorithms. In *25th Brazilian Symposium on Programming Languages*, pages 49–57.

Cardoso, D. J., Foss, L., and Bois, A. R. D. (2019). A graph transformation system formalism for software transactional memory opacity. In *Proceedings of the XXIII Brazilian Symposium on Programming Languages*, pages 3–10.

Guerraoui, R. and Kapalka, M. (2008). On the correctness of transactional memory. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 175–184.

Guerraoui, R. and Kapałka, M. (2010). Principles of transactional memory. *Synthesis Lectures on Distributed Computing*, 1(1):1–193.

Harris, T., Marlow, S., Peyton-Jones, S., and Herlihy, M. (2005). Composable memory transactions. In *Proceedings of the Tenth ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, PPoPP '05, pages 48–60, New York, NY, USA.

Herlihy, M. and Moss, J. E. B. (1993). *Transactional memory: Architectural support for lock-free data structures*, volume 21. ACM.

Khyzha, A., Attiya, H., Gotsman, A., and Rinetzky, N. (2018). Safe privatization in transactional memory. *ACM SIGPLAN*, 53:233–245.

Lesani, M. and Palsberg, J. (2014). Decomposing opacity. In *International Symposium on Distributed Computing*, pages 391–405. Springer.

Marić, O. (2017). *Formal Verification of Fault-Tolerant Systems*. PhD thesis, ETH Zurich.

Matveev, A. and Shavit, N. (2015). Reduced hardware norec: A safe and scalable hybrid tm. In *ACM SIGARCH Computer Architecture News*, volume 43, pages 59–71.

Pankratius, V. and Adl-Tabatabai, A.-R. (2011). A study of transactional memory vs. locks in practice. In *Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures*, pages 43–52. ACM.

Peluso, S., Palmieri, R., Romano, P., Ravindran, B., and Quaglia, F. (2015). Disjoint-access parallelism: Impossibility, possibility, and cost of TM implementations. In *Proce. of the 2015 ACM Symp. on Principles of Distri. Computing*, pages 217–226.

Peterson, C. and Dechev, D. (2017). A transactional correctness tool for abstract data types. *ACM Transactions on Architecture and Code Optimization*, 14(4):1–24.

Shavit, N. and Touitou, D. (1997). Software transactional memory. *Distributed Computing*, 10:99–116.

Wamhoff, J.-T., Riegel, T., Fetzer, C., and Felber, P. (2010). Robustm: A robust software tm. In *Symp. on Self-Stabilizing Systems*, pages 388–404.

A New Total Order for Triangular Fuzzy Numbers with an Application

Valentina Rosas[1], Benjamín Bedregal[2], José Canumán[3], Roberto Díaz[4], Edmundo Mansilla[5] and Nicolás Zumelzu[5]

[1]Carrera de Pedagogía en Matemática para Educación Media
Departamento de Educación y Humanidades – Universidad de Magallanes
Punta Arenas – Chile

[2]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte – Natal, RN – Brazil

[3]Departamento de Ingeniería en Computación e Informática
Universidad de Magallanes – Punta Arenas – Chile

[4]Departamento de Ciencias Exactas
Universidad de Los Lagos – Osorno – Chile

[5]Departamento de Matemática y Física – Universidad de Magallanes
Punta Arenas – Chile

{valrosas,jose.canuman,edmundo.mansilla,nicolas.zumelzu,}@umag.cl

bedregal@dimap.ufrn.br,roberto.diaz@ulagos.cl

Abstract. *This work deals with the study of a new total order Triangular Fuzzy Numbers and arithmetic properties that are maintained in relation to the operations of addition and subtraction. Additionally, we present as example of application the shortest path solution for the Travelling Salesman Problem with fuzzy distances.*

Resumo. *Este trabalho trata do estudo de uma nova ordem total para números fuzzy triangulares e propriedades aritméticas que são mantidas em relação às operações de adição e subtração. Além disso, apresentamos como exemplo de aplicação a solução do caminho mais curto para o Problema do Caixeiro Viajante com distâncias fuzzy.*

1. Introduction

In [Zadeh 1965] was proposed the concept of fuzzy sets in order to formalize and model the ambiguities and imprecision inherent to some linguistic terms, like to downs, young and quick, associated to linguistic variables like temperature, age and speed. Fuzzy numbers, a special class of fuzzy sets, were proposed in [Zadeh 1965] to model a quantity which is imprecise, rather than exact as is the case of a real number. Besides, arithmetic operations for fuzzy numbers and their properties had been widely study (see [Dubois and Prade 1978, Wang and Wang 2014]). In particular, we only consider fuzzy numbers with a triangular shape, which are called of Triangular Fuzzy Numbers, TFN in short.

In this sense, besides to serious algebraic consequences, the impossibility of obtaining both multiplicative and additive inverses is an inconvenient when trying to solve equations, even if it is a simple one of the first degree, which makes it necessary to find methods to solve this problem.

There are several proposal for orders on fuzzy numbers in general [Buckley 2005, Wang and Wang 2014, Zumelzu et al. 2020] and for TFN in particular (see for example [Akyar et al. 2012, Asmus.T.C. et al. 2017, Nasseri and Behmanesh 2013]) and here we propose a new total order for TFNs which be compatible, in some sense, with the addition and subtraction. Thereby, we propose to retrieve some characteristics of fields using order properties. We also apply both, the addition and the total order on TFN, to solve the minimum path problem in fuzzy weighted graphs.

This paper is organized as follows: In Section 2, in addition to establishing the notation used, we recall some essential notions for the remaining sections. In Section 3 we see the more basic order on fuzzy numbers and properties of order are studied with arithmetic operations. The notion of orders for fuzzy numbers is applied in Section 4 including an algorithm used to find these shortest routes. Finally, Section 5 provides some final remarks.

2. Preliminaries

In this section, we provide the concepts of fuzzy number and the main tools that will be used to obtain the main results. Let $\mathbb{R}^3 = \mathbb{R}\times\mathbb{R}\times\mathbb{R}$, with $\mathbb{R}$ being the set of real numbers. The notation $<$, $>$, $\leq$ or $\geq$ stands for usual order of $\mathbb{R}$, and $[a,b]$, $]a,b]$, $[a,b[$ and $]a,b[$ the intervals of $\mathbb{R}$ closed, right-closed, right-open, and open respectively (see [Rudin 1964]).

Definition 2.1. *[Klir and Yuan 1995] A fuzzy set A on $\mathbb{R}$ is called a fuzzy number if it satisfies the following conditions*

(i) A is normal, i.e., $\sup A(x) = 1$;
(ii) ${}^{A}\!/_{\alpha}$ is a closed interval for every $\alpha \in]0,1]$;
(iii) the support of A is bounded,

where ${}^{A}\!/_{\alpha}$ is the α-level (or α-cut) of A.

A fuzzy number A is crisp if there exists $r \in \mathbb{R}$ such that $A(r) = 1$ and $A(x) = 0$ for each $x \neq r$. In this case we will denote A by $\widetilde{r}$. Finally, $\mathcal{F}(\mathbb{R})$ will denote the set of all fuzzy numbers.

Proposition 2.1. *[Dubois and Prade 1978] Let $A, B \in \mathcal{F}(\mathbb{R})$ then the fuzzy sets $A + B$ and $A - B$ defined by*

$$(A+B)(x) = \sup_{x=y+z} \min\{A(y), B(z)\} \text{ and } (A-B)(x) = \sup_{x=y-z} \min\{A(y), B(z)\},$$

for each $x \in \mathbb{R}$ is a fuzzy number.

Definition 2.2. *[Klir and Yuan 1995] A fuzzy number A, is called a triangular fuzzy number,* TFN *in short, if there is $(a,b,c) \in \mathbb{R}^3$ such that $a \leq b \leq c$ and*

$$A(x) = \begin{cases} 1, & \text{if } x = b, \\ \frac{x-a}{b-a}, & \text{if } x \in]a,b[, \\ \frac{c-x}{c-b}, & \text{if } x \in]b,c[, \\ 0, & \text{other cases.} \end{cases}$$

Proposition 2.2. *[Dubois and Prade 1978] Let* $A = (a_1, b_1, c_1)$ *and* $B = (a_2, b_2, c_2)$ *be TFNs. Then*

$$A + B = (a_1 + a_2, b_1 + b_2, c_1 + c_2) \text{ and } A - B = (a_1 - c_2, b_1 - b_2, c_1 - a_2).$$

In the following, we consider the notion of fuzzy weighted graphs, which are weighted graphs (see [Bondy and Murty 1976]) where the weights are fuzzy numbers.

Definition 2.3. *[Cornelis et al. 2004] A fuzzy weighted graph is a triple* $G = \langle V, E, c\rangle$ *where* V *is a set whose elements are called vertices,* $E \subseteq V \times V$ *is a set of edges and* $c : E \to \mathcal{F}(\mathbb{R})$ *is the cost (or weight) function. Given* $v, u \in V$*, a* (v, u)*-path in* G *is a finite and non empty sequence of edges* $p = (e_1, \ldots, e_n) = ((v_1, u_1), \ldots, (v_n, u_n))$ *such that* $v_i = u_{i-1}$ *for each* $i = 2, \ldots, n$*,* $v_1 = v$ *and* $u_n = u$*. A* (v, u)*-path is a cycle if* $v = u$*.*

The cost of a (v, u)-path $p = (e_1, \ldots, e_n)$, denoted by $c(p)$ is given by the addition of the cost of each edge in the path, i.e. $c(p) = \sum_{i=1}^{n} c(e_i)$. Given a pair of vertices (v, u) in a fuzzy weighted graph G there may exist several or no (v, u)-path.

3. Total order on triangular fuzzy numbers

In this section, we prove that the order to propose is a total order, we also study properties that it verifies with addition and subtraction operations.

Definition 3.1. *Let* (a_1, b_1, c_1) *and* (a_2, b_2, c_2) *two triangular fuzzy numbers.*

$$(a_1, b_1, c_1) \leq_{OT} (a_2, b_2, c_2) \text{ if and only if } \begin{cases} b_1 < b_2, \text{ or} \\ b_1 = b_2 \text{ and } a_1 < a_2, \text{ or} \\ b_1 = b_2 \text{ and } a_1 = a_2 \text{ and } c_1 \leq c_2. \end{cases}$$

Proposition 3.1. *The relation* $\leq_{OT}$ *is a total order.*

Proof. Let $(a_1, b_1, c_1), (a_2, b_2, c_2), (a_3, b_3, c_3) \in \mathcal{F}(\mathbb{R})$.

1. Reflexivity: Straightforward from Definition 3.1.

2. Antisymmetry: Let $(a_1, b_1, c_1) \leq_{OT} (a_2, b_2, c_2)$ and $(a_2, b_2, c_2) \leq_{OT} (a_1, b_1, c_1)$. Suppose that $A \neq B$. Then $a_1 \neq a_2$, $b_1 \neq b_2$ or $c_1 \neq c_2$. If $b_1 \neq b_2$ then either $A <_{OT}$ B or $B <_{OT}$ A but not both. So, $b_1 = b_2$. If $b_1 = b_2$ and $a_1 \neq a_2$ then, $A <_{OT} B$ or $B <_{OT} A$ but not both. Hence $b_1 = b_2$ and $a_1 = a_2$. If $b_1 = b_2$ and $a_1 = a_2$ and $c_1 \neq c_2$ then either $A <_{OT} B$ or $B <_{OT} A$ but not both. Thereby, $a_1 = a_2$, $b_1 = b_2$ and $c_1 = c_2$.

3. Transitivity: Let $(a_1, b_1, c_1) \leq_{OT} (a_2, b_2, c_2)$ and $(a_2, b_2, c_2) \leq_{OT} (a_3, b_3, c_3)$ is equivalent to

i) $b_1 < b_2 \leq b_3$ or $b_1 = b_2 < b_3$, or
ii) $b_1 = b_2 = b_3$ and $(a_1 < a_2 \leq a_3$ or $a_1 = a_2 < a_3)$, or
iii) $b_1 = b_2 = b_3$ and $a_1 = a_2 = a_3$ and $(c_1 \leq c_2 \leq c_3)$.

For the case i) we have to $b_1 < b_3$. For the other cases ii) and iii) it is analogous. Therefore, $(a_1, b_1, c_1) \leq_{OT} (a_3, b_3, c_3)$.

4. Totality: Let $A \neq B$, where $A = (a_1, b_1, c_1)$ and $B = (a_2, b_2, c_2)$. Then by Definition 3.1 and trichotomy property of real numbers we have $b_1 < b_2$ or $b_1 > b_2$ or $b_1 = b_2$. For the first two cases $(a_1, b_1, c_1) <_{OT} (a_2, b_2, c_2)$ or $(a_2, b_2, c_2) <_{OT} (a_1, b_1, c_1)$. Furthermore, if $b_1 = b_2$ then by Definition 3.1 and trichotomy property of real numbers $a_1 < a_2$ or $a_1 > a_2$ or $a_1 = a_2$. For the first two cases $(a_1, b_1, c_1) <_{OT} (a_2, b_2, c_2)$ or $(a_2, b_2, c_2) <_{OT} (a_1, b_1, c_1)$. But, if $a_1 = a_2$ then by Trichotomy on real numbers $c_1 < c_2$ or $c_1 > c_2$ or $c_1 = c_2$. Therefore, $A \leq_{OT} B$ or $B \leq_{OT} A$. In this way the Proposition is proven. □

Definition 3.2. *Let A be a TFN. Then,*

1. *A is OT-positive if $A >_{OT} \widetilde{0}$.*
2. *A is OT-negative if $A <_{OT} \widetilde{0}$.*

Proposition 3.2. *Let (a, b, c) be a TFN. Then, (a, b, c) is OT-positive if and only if $b > 0$ or ($a = b = 0$ and $c > 0$). Dually, (a, b, c) is OT-negative if and only if $b \leq 0$ and $a < 0$.*

Proof. Let (a, b, c) be a OT-positive TFN. Then, $0 < b$, or, $0 = b$ and $0 < a$ (contradiction), or, $0 = a$ and $0 = b$ and $0 < c$. The, converse is trivial. Moreover, let (a, b, c) be a OT-negative TFN. Then $b < 0$ (and therefore $a < 0$), or $b = 0$ and $a < 0$, or $a = b = 0$ and $c < 0$ (contradiction). So, $b \leq 0$ and $a < 0$. □

Corollary 3.1. *Let A be a TFN. Then, A is OT-positive if and only if $-A$ is OT-negative.*

Proof. Straightforward from Proposition 3.2. □

The following theorem proposes a property of sub-additive inverse.

Theorem 3.1. *Let A be a TFN. Then $A - A \leq_{OT} \widetilde{0}$. In addition, $A - A = \widetilde{0}$ if and only if A is crisp.*

Proof. Let $A = (a, b, c)$ be a TFN. By Proposition 2.2, $A - A = (a - c, b - b, c - a)$. Then $b - b = 0$ and $a - c \leq 0$. If $a - c = 0$ then $A - A = \tilde{0}$ and otherwise $A - A <_{OT} \widetilde{0}$. This completes the proof. In addition, $A - A = \widetilde{0}$ iff $(a - c, b - b, c - a) = (0, 0, 0)$ iff $a = c$ iff $a = b = c$ iff A is crisp. □

Theorem 3.2. *Let A and B two TFNs. If A and B are OT-positive then $A + B$ is also a OT-positive TFN.*

Proof. Let $A = (a_1, b_1, c_1)$ and $B = (a_2, b_2, c_2)$ two OT-positive TFNs. Then from Proposition 3.2 we have two cases. (1) $b_1 > 0$ or $b_2 > 0$ then $b_1 + b_2 > 0$. (2). $a_1 = b_1 = b_2 = a_2 = 0$ and $c_1 > 0$ and $c_2 > 0$ and therefore $a_1 + a_2 = b_1 + b_2 = 0$ and $c_1 + c_2 > 0$. Therefore, in both cases, $A + B >_{OT} \widetilde{0}$. □

Proposition 3.3. *Let A, B, C and D be TFNs. The following properties are satisfied*

1. *If $A \leq_{OT} B$ then $A + C \leq_{OT} B + C$.*
2. *If $A \leq_{OT} B$ and $C \leq_{OT} D$ then $A + C \leq_{OT} B + D$.*

Proof. Let $A = (a_1, b_1, c_1)$, $B = (a_2, b_2, c_2)$ and $C = (a_3, b_3, c_3)$ be TFNs. So, $A \leq_{OT} B$ if and only if the following assertions are verified:

i) $b_1 < b_2$, or
ii) $b_1 = b_2$ and $a_1 < a_2$, or
iii) $b_1 = b_2$ and $a_1 = a_2$ and $c_1 \leq c_2$.

Firstly we note that i) $b_1 + b_3 < b_2 + b_3$ or ii) $b_1 + b_3 = b_2 + b_3$ and $a_1 + a_3 < a_2 + a_3$, or $b_1 + b_3 = b_2 + b_3$, $a_1 + a_3 = a_1 + a_3$ and $c_1 + c_3 < c_2 + c_3$ then $A + C \leq_{OT} B + C$.

For 2. Straightforwardly from Definition 3.1 and assertion 1. This completes the proof. □

Proposition 3.4. *For each TFN A and B, there is a TFN X which is a solution of the inequality $A + X \geq_{OT} B$.*

Proof. Let A and B be TFNs. According with the Theorem 3.1 and Proposition 3.3, we have:

$$X \geq_{OT} (A - A) + X \geq_{OT} B - A.$$

Therefore $X \geq_{OT} B - A$. In this way the Proposition is proven. □

4. Selecting route to visit South American

In this section we collect distances between the capitals of South American countries on the following websites:

1. `https://www.distanciasentreciudades.com/`
2. `https://www.distancia.co/`
3. `https://www.geodatos.net/`
4. `https://es.distance.to/`

For example, between Caracas Venezuela to La Paz Bolivia where the distances are 3006.02, 2988, 3009, 3004.04 the number is $(2988, 3005.03, 3009)$. In this problem of the postman with 13 capital cities the possibilities of routes is very large number (6,227,020,800 routes) and it is possible to calculate it by the permutations considering 13 elements and only by adding the initial element at the end we would obtain the desired paths. Table 1 indicates the distances between the cities and the Figure 1 represents the locations on the continent. We have to consider that the names of the capitals are associated its acronym under the Code IATA[1].

Algorithm 1 For graph $G(V, E)$ find all paths and calculate total sum of paths. Finally select the shortest distance like Travelling Salesman Problem Algorithm.

```
1: for all w ∈ V do
2:     allpaths=findAllPaths(w,P)        ▷ P is the weight fuzzy of the path
3:     select-min(sort(allpaths))
```

[1]See `https://www.iata.org/en/publications/directories/code-search/`.

Algorithm 2 Find all paths from a given source to all nodes in a fuzzy weighted graph.

function FINDALLPATHS($startnode$, $FuzzP$)
  $init(addFuzz)$
  v = $startnode$;
  e = $\{v, w\}$ ▷ w some vertex different from the previous
  Add $FuzzP[e]$ to $addFuzz$
  A = $\{v, w\}$ ▷ A set of vertices visited
  if w is not visited **then**
    Add e to A;
  while (A! = V) **do**
    Find e = $\{v, w\}$ | $v \in A$ and $w \in V - A$
    Add $FuzzP[e]$ to $addFuzz$
    Add w to A
  return A

Algorithm 2 determines all the routes that go through all the capitals of South America, starting from any one of them, to order them according to their cost. Algorithm 1 determines all routes starting from a specific capital. In addition, Table 2 shows the 10 shortest roads starting in Santiago de Chile. In our case, a Chilean who wants to go out traveling for the capitals of South America starting from Santiago, the algorithm, points out the shortest route.

Table 1. Triangular fuzzy numbers for distances between the capitals of the countries of South America

KM	ASU	BOG	BSB	BUE	CCS	GEO	LPB	LIM	MVD	PBM	POS	UIO	SCL
ASU		(3772, 3774.10, 3775.38)	(1460, 1463.84, 1712.77)	(1036, 1037.80, 1040)	(4082, 4105.06, 4108)	(3536.41, 3560.69, 3573)	(1462, 1465.06, 1466)	(2510, 2512.58, 2515)	(1069.33, 1075.89, 1078)	(3457, 3474.94, 3475.89)	(4000, 4020.63, 4025)	(3568, 3578.45, 3583)	(1547.80, 1550.48, 1555)
BOG	(3772, 3774.10, 3775.38)		(3280, 3663.50, 3671.16)	(4644, 4662.59, 4665)	(1019.86, 1026.89, 1028)	(1779.03, 1779.70, 1781)	(2424, 2438.58, 2442.27)	(1870, 1882.69, 1889.74)	(4759, 4776.36, 4781.46)	(2095.84, 2100.21, 2101)	(1533.53, 1538.40, 1541)	(728, 730.43, 737.12)	(4229, 4250.49, 4254)
BSB	(1460, 1463.84, 1712.77)	(3280, 3660, 3671.16)		(2333, 2346.20, 2677.81)	(3143.46, 3583, 3597.45)	(2290.46, 2730.77, 2748)	(2067.57, 2163, 2165.29)	(2987.61, 3168, 3171.08)	(2273, 2284.77, 2633.27)	(2074.20, 2524.50, 2538.49)	(2837.99, 3290.50, 3303.05)	(3452.82, 3776.50, 3781.41)	(3008, 3014.98, 3209.94)
BUE	(1036, 1037.80, 1040)	(4644, 4662.59, 4665)	(2333, 2342.20, 2677.81)		(5072, 5093.94, 5101)	(4571.11, 4594.45, 4610)	(2231, 2233.34, 2239)	(3131.73, 3132.28, 3141)	(203, 206.03, 210.55)	(4493, 4513.63, 4514)	(5022, 5043.85, 5052)	(4346, 4355.17, 4365)	(1129.08, 1135.19, 1140)
CCS	(4082, 4105.06, 4108)	(1019.86, 1026.89, 1028)	(3143.46, 3587, 3597.45)	(5072, 5093.94, 5101)		(1042.64, 1044.50, 1068.27)	(2988, 3005.03, 3009)	(2733, 2746.16, 2748)	(5149, 5169.24, 5178)	(1387.38, 1389.86, 1394)	(586.40, 588.50, 590.03)	(1751, 1753.57, 1754.65)	(4880, 4900.87, 4911)
GEO	(3536.41, 3560.69, 3573)	(1779.03, 1779.70, 1781)	(2290.46, 2730.77, 2756)	(4571.11, 4594.45, 4610)	(1042.64, 1044.50, 1068.27)		(2786.94, 2808.85, 2818)	(2941.43, 2954.22, 2960)	(4608.13, 4627.82, 4646)	(330.02, 346.68, 349)	(565, 566.12, 597.73)	(2388.50, 2391.67, 2392)	(4634.54, 4655.19, 4674)
LPB	(1462, 1465.06, 1466)	(2424, 2438.58, 2442.27)	(2067.57, 2163, 2165.29)	(2231, 2233.34, 2239)	(2988, 3005.03, 3009)	(2786.94, 2808.85, 2818)		(1076.49, 1076.99, 1078)	(2361, 2364.07, 2369)	(2857, 2867.27, 2867.84)	(3092, 3106.73, 3112)	(2130, 2136.97, 2141)	(1895, 1899.43, 1906)
LIM	(2510, 2512.58, 2515)	(1870, 1882.69, 1889.74)	(2987.61, 3169, 3171.08)	(3131.73, 3132.28, 3141)	(2733, 2746.16, 2748)	(2941.43, 2954.22, 2960)	(1076.49, 1076.99, 1078)		(3292.81, 3294.91, 3301)	(3127, 3132.38, 3133.96)	(3041, 3052.01, 3055)	(1317, 1325.29, 1327)	(2459, 2464.14, 2472)
MVD	(1069.33, 1075.89, 1078)	(4759, 4774.32, 4780)	(2273, 2280.77, 2633.27)	(203, 206.03, 210.55)	(5149, 5169.24, 5178)	(4608.13, 4627.82, 4646)	(2361, 2364.07, 2369)	(3292.81, 3294.91, 3301)		(4514, 4530.41, 4534.65)	(5075, 5093.62, 5103)	(4486, 4496.23, 4505)	(1339.13, 1339.64, 1341)
PMB	(3457, 3474.94, 3475.89)	(2095.84, 2100.21, 2101)	(2074.20, 2529, 2538.49)	(4493, 4513.26, 4514)	(1387.38, 1389.86, 1394)	(330.02, 346.68, 349)	(2857, 2867.27, 2867.84)	(3127, 3132.38, 3133.96)	(4514, 4530.41, 4534.65)		(878, 878.73, 883)	(2680.02, 2680.52, 2682)	(4649, 4664.37, 4670)
POS	(4000, 4020.63, 4025)	(1533.53, 1538.40, 1541)	(2837.99, 3294.53, 3304)	(5022, 5043.85, 5052)	(586.40, 588.50, 590.03)	(565, 566.12, 597.73)	(3092, 3106.73, 3112)	(3041, 3052.67, 3057.14)	(5075, 5093.62, 5103)	(878, 878.73, 883)		(2235, 2236.43, 2239)	(4978, 4997.12, 5009)
UIO	(3568, 3578.45, 3583)	(728, 730.43, 737.12)	(3452.82, 3778.50, 3781.41)	(4346, 4355.17, 4365)	(1751, 1753.57, 1754.65)	(2388.50, 2391.67, 2392)	(2130, 2136.97, 2141)	(1317, 1325.29, 1327)	(4486, 4496.23, 4505)	(2680.02, 2680.52, 2682)	(2235, 2236.43, 2239)		(3769, 3782.84, 3793)
SCL	(1547.80, 1550.48, 1555)	(4229, 4250.49, 4254)	(3008, 3012.48, 3209.94)	(1129.08, 1135.19, 1140)	(4880, 4900.87, 4911)	(4634.54, 4655.19, 4674)	(1895, 1899.43, 1906)	(2459, 2464.14, 2472)	(1339.13, 1339.64, 1341)	(4649, 4664.37, 4670)	(4978, 4997.12, 5009)	(3769, 3782.84, 3793)	

Figure 1. The shortest route from Santiago de Chile.

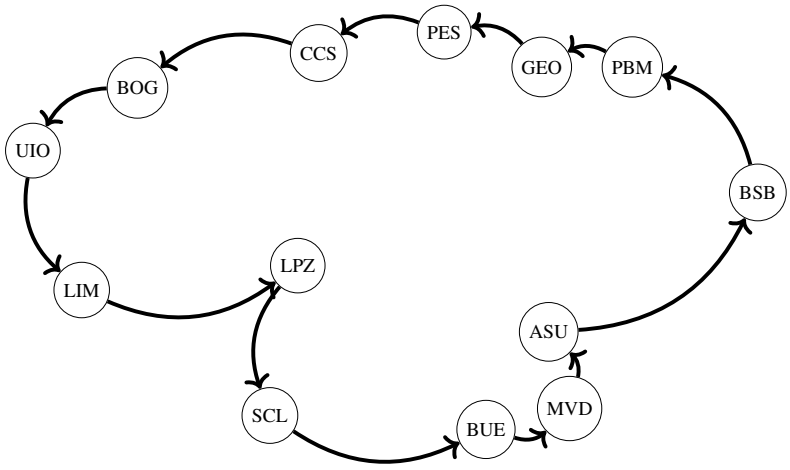


Table 2. The 10 shortest routes to the postman problem in South America.

$N°$	Route	Cost
1	SCL-LPB-LIM-UIO-BOG-CCS-POS-GEO-PBM-BSB-ASU-MVD-BUE-SCL	(13464.3, 13970.6, 14292.7)
2	SCL-BUE-MVD-ASU-BSB-PBM-GEO-POS-CCS-BOG-UIO-LIM-LPB-SCL	(13464.4, 13972.3, 14292.7)
3	SCL-LPB-LIM-UIO-BOG-CCS-POS-GEO-PBM-BSB-ASU-BUE-MVD-SCL	(13632, 14133.5, 14455.7)
4	SCL-MVD-BUE-ASU-BSB-PBM-GEO-POS-CCS-BOG-UIO-LIM-LPB-SCL	(13641.1, 14138.7, 14455.7)
5	SCL-BUE-MVD-ASU-BSB-GEO-PBM-POS-CCS-BOG-UIO-LIM-LPB-SCL	(13993.6, 14491.2, 14787.5)
6	SCL-MVD-BUE-ASU-BSB-GEO-PBM-POS-CCS-BOG-UIO-LIM-LPB-SCL	(14170.4, 14657.6, 14950.5)
7	SCL-BUE-MVD-ASU-LPB-LIM-UIO-BOG-CCS-POS-GEO-PBM-BSB-SCL	(14770.34, 15277.54, 15349.84)
8	SCL-BUE-MVD-ASU-BSB-PBM-GEO-POS-CCS-BOG-UIO-LPB-LIM-SCL	(14843.4, 15350, 15672.7)
9	SCL-MVD-BUE-ASU-LPB-LIM-UIO-BOG-CCS-POS-GEO-PBM-BSB-SCL	(14947.04, 15443.84, 15512.84)
10	SCL-BUE-MVD-ASU-LPB-LIM-UIO-BOG-CCS-POS-PBM-GEO-BSB-SCL	(15299.54, 15796.34, 15844.64)

5. Final remarks

We have presented a total order for triangular fuzzy numbers. Then we have shown properties that fulfill this order with the addition and subtraction operation, after studying these properties we can speak of a structure give for the triple $(\text{TFN}, +, \leq_{OT})$ that we could designate by *ordered Abelian quasi-group* or *ordered Abelian pseudo-group* (see [Clifford 1940, Levi 1942, Everett and Ulam 1945, Birkhoff 1987]) by the proposal of additive sub-inverse. In addition, we include an illustrative example for the Travelling Salesman Problem considering the capitals of those of South America.

Acknowledgments

This work was partially supported by the Brazilian funding agency CNPq (Brazilian Research Council) under Project no 311429/2020-3.

Bibliography

Akyar, E., Akyar, H., and Düzce, S. A. (2012). A new method for ranking triangular fuzzy numbers. *Int. J. Uncertain. Fuzziness Knowl. Based Syst.*, 20(5):729–740.

Asmus.T.C., Dimuro, G. P., and Bedregal, B. R. C. (2017). On two-player interval-valued fuzzy Bayesian games. *Int. J. Intell. Syst.*, 32(6):557–596.

Birkhoff, G. (1987). Lattice-ordered groups. *Selected Papers on Algebra and Topology by Garrett Birkhoff*, 43(2):412.

Bondy, J. and Murty, U. (1976). *Graph theory with applications*. Published by The Macmillan Press Ltd.

Buckley, J. J. (2005). *Fuzzy probabilities: new approach and applications*, volume 115. Springer Science & Business Media.

Clifford, A. H. (1940). Partially ordered Abelian groups. *Annals of Mathematics*, pages 465–473.

Cornelis, C., De Kesel, P., and Kerre, E. E. (2004). Shortest paths in fuzzy weighted graphs. *International journal of intelligent systems*, 19(11):1051–1068.

Dubois, D. and Prade, H. (1978). Operations on fuzzy numbers. *International Journal of Systems Science*, 9(6):613–626.

Everett, C. J. and Ulam, S. (1945). On ordered groups. *Transactions of the American Mathematical Society*, 57(2):208–216.

Klir, G. and Yuan, B. (1995). *Fuzzy Sets and Fuzzy Logic: Theory and Applications*, volume 4. Prentice Hall New Jersey.

Levi, F. W. (1942). Ordered groups. In *Proceedings of the Indian Academy of Sciences-Section A*, volume 16, pages 256–263. Springer India.

Nasseri, S. H. and Behmanesh, E. (2013). Linear programming with triangular fuzzy numbers – a case study in a finance and credit institute. *Fuzzy Information and Engineering*, 5(3):295–315.

Rudin, W. (1964). *Principles of Mathematical Analysis*, volume 3. McGraw-hill New York.

Wang, W. and Wang, Z. (2014). Total orderings defined on the set of all fuzzy numbers. *Fuzzy Sets and Systems*, 243:131–141.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Zumelzu, N., Bedregal, B., Mansilla, E., Bustince, H., and Díaz, R. (2020). Admissible orders on fuzzy numbers. *arXiv preprint arXiv:2003.01530*.

A-Games: using game-like representation for representing finite automata

Cleyton Slaviero[1], **Edward Hermann Haeusler**[2]

[1]Instituto de Ciências Exatas e Naturais
Universidade Federal de Rondonópolis (UFR)
Rondonópolis – MT – Brazil

[2]Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio)
Rio de Janeiro – RJ – Brazil

slaviero@ufr.edu.br,hermann@inf.puc-rio.br

Abstract. *Non-determinism in automata theory allows us to model situations where given an input, one or more outputs are possible. Although this decision regarding which state to chose could be random, there are contexts where this decision is not random, for instance when modeling real life situations. Using game theory, we propose the representation of automata as a game of two players. This game is defined for languages of finite size. We characterize that this representation is suited for both deterministic and non-deterministic automata, and relate the former with perfect information games, and the latter with imperfect information games. We argue that this could help us in explaining concurrency in programming, for instance in novice programming environments.*

1. Introduction

Automata theory studies abstract computing devices [1]. These devices, or machines, have been used in many contexts, and allow us to model and understand interactions in digital circuits; in lexical analysis; software for analyzing large data sets; and software for verifying systems such as communication protocols. For simple problems, such as turning a switch on or off, it is straightforward to define the set of possible paths of the automaton, or the language that it recognizes. However, when modeling more complex systems and using more complex automata, with non-deterministic features for instance, one question arises: how to "decide" if a given transition is chosen over another, given the same input? These kinds of decision problems may arise, for instance, in concurrent systems, which can be modeled by automata such as parallel automata or even Petri Nets [2, 3, 4], or in simple games created by novice programmers. In a previous work [5] we found that the novice programming environments present different approaches to concurrency. Learning concurrency and understanding situations such as synchronization and order-of-execution is a challenge [6] and many bugs regarding concurrency arise when analyzing programs [7]. To this end, our question arises: how could we better understand and tackle the understanding of concurrency? The famous seminal work from Morgenstern and von Neumann [8] defined an area in economic theory called Game Theory. Since then, researchers have been investigating the connection between computer science and game theory [9, 10]. Game theory studies decision problems, in which two (or more)

players need to decide, given their interests, beliefs and knowledge about the other(s), what actions to take. The connection between game theory and computer science has been explored to great extent [9, 10]. Shoham [10] presents a comparison between a work from Kalai in 1995 and current trends in game theory and computer science. From the last two foci the author mentions, "logics of knowledge and belief, and other logical aspects of games" remains a key area to approach game theory and computer science. In this research, we attempt to discuss this relationship between game theory and computer science via automata theory. This is not new: the seminal works from Reif [11, 12] discuss this relationship by means of describing complexity in computing a game using Turing machines. Many subsequent works have been investigating these kinds of relationship, such as the works on algorithmic game theory [13, 14]. In these works, automata are employed to discuss the complexity of strategies in games. In our context, we want to do the opposite: discuss automata and the characteristics of systems they represent by using a game-theoretical approach. Our goal is to investigate how game theory could support the characterization of non-determinism in terms of rational behavior. To this extent, we propose to represent automata as games. In this work we start by showing how games and finite automata could be related. Furthermore, we show that finite deterministic automata are equivalent to perfect information games, whilst finite non-deterministic automata are equivalent to imperfect information games. We finish this paper by showing further works that could be explored from these results. We expect that these new definitions help us in exploring concurrency problems and proposing new ways to explore and solve them.

2. Automata Theory

Automata theory (AT) studies abstract computing devices or "machines" [1]. Automata can be used to model, for instance, in lexical analysis; text corpus analysis; and systems verification. Formally, a Finite State Machine, or Finite Deterministic Automata (FDA), is defined by D = $\langle Q_D, \Sigma, \delta_D, q_0, F_D \rangle$, where:

- Q_D is a non-empty finite set (of states);
- Σ is a finite (non-empty) set of symbols (the alphabet)
- q_0 is the initial state
- F_D is the finite set of final states (or accepting states)
- $\delta_D : Q_D \times \Sigma \to Q_D$ is the possibly partial transition function taking as argument a state and an input and returning a state.

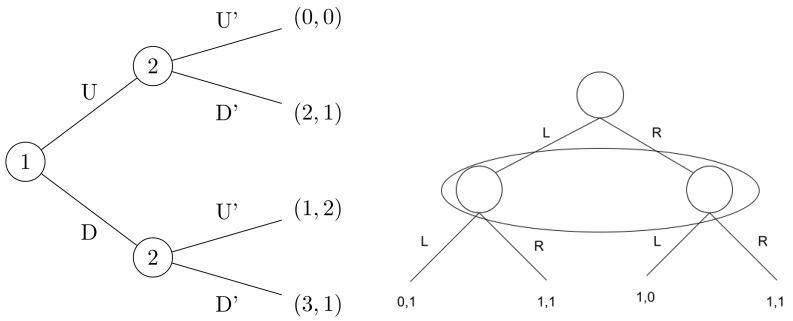
We can also define $\hat{\delta}(q, w)$, $q \in Q_D$ and $w \in \Sigma^*$, and returns $q' \in Q_D$ as the state that automata reaches when it runs over w [1]. From $\hat{\delta}$ we are able to define L(A), which is the language accepted by A as $\{w | \hat{\delta}(q_0, w) \in F_D\}$. [1].

To define a non-deterministic finite automaton (NFA), $\delta : Q_D \times \Sigma \to \mathcal{P}(Q_D)$ is a function which takes a state and an input and returns a set of states.

3. Game Theory

Informally defined, a perfect information game is a game where each player knows the move other player's chose when playing the game. More formally, a (strategic) game is a 3-uple $\langle P, A_i, \succcurlyeq_i \rangle$ where [15]:

- P is a set of players



(a) Extensive form. **(b)** Imperfect Information Game

Figure 1. Examples of games.

- A_i is a set of moves of player i
- $\succcurlyeq_i$ is a preference relation of player i.

A famous example of a game is the Prisoner's Dilemma, in which two prisoners must choose to confess or deflect from a given accusation. In either choice they could do, there is a better or worse payoff considering the other prisoners' decision. In Figure 1a we present this game in extensive form, one type of representation for games. In this picture, preferences are described as numerical values. This follows as a consequence of the utility theory from Von Neumann and Morgenstern [8], in which $x \succcurlyeq y$ iff $U(x) \geqslant U(y)$.

Another class of games is the one where a given player has no information about previous moves from other player(s) [15], which we call imperfect information game. In this class of games, an information set groups possible outcomes the player might be at, after a move from previous players. The current player has no knowledge of which move the previous players performed. Figure 1b show an example of an imperfect information game where the players must choose between left and right during their moves. However, after player one chooses either move, player two is not aware of player's 1 choice.

4. Automata as Games

In order to describe automata as games, we start by providing a game representation for finite deterministic automata. In this section, we describe the general idea and provide examples of automata and the equivalent game.

Definition 4.1 *(A-Game) Let A be a finite automaton $\langle Q_A, \Sigma, \delta_A, q_0, F_A\rangle$ and let G_A be an extensive game with two players, P_I and P_{II} . In this game, P_I plays states from Q_A and P_{II} plays characters from Σ_A^*.*

The game is played as follows. P_I starts by playing q_0, the initial state. In order to proceed, P_{II} must play any character $s \in \Sigma^*$. The game ends when P_I plays a state $q \in F_A$ and P_{II} has no more moves. This will be later defined as a winning strategy. However, we must first define the concept of strategy for these players.

Definition 4.2 *The strategy for each player is the following: if P_I plays $q \in \Sigma_A$ then P_{II} must play some s, such that $\delta(q, s) \neq \emptyset$, or else P_{II} loses.*

It happens that if A is deterministic, then G_A is equivalent to a perfect information game. However, if A as non-deterministic, then $\delta(q, s) = B \subseteq Q_A$, and thus we would

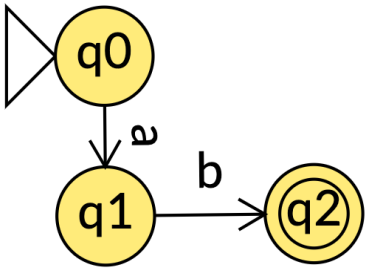


Figure 2. A finite deterministic automaton.

have an information set around P_{II} 's next move after P_I plays $q" \in B$. This allows us to infer that δ defines the information sets for P_{II} . Consider that P_I plays $q" \in B$. Then, P_I must play $s" \in \Sigma$ such that $\delta(q, s')$ is defined and $q \in B$.

In this game, we can also define the strategies of each player. According to the definition from [15], the strategy of a player assigns an action chosen by the player for *every* history in which it is his turn to move. The history of a player is defined by the set of actions he might choose during the game play. For P_I this is equivalent to all combination of states he might chose to confront P_{II} ; regarding P_{II} , these histories is any word $w \in \Sigma^*{}_A$.

Definition 4.3 *Let A be an automaton and $w \in \Sigma^*{}_A$. A strategy for P_{II} defined by w is such that P_{II} chooses symbols from w sequentially, apart from what P_I chooses.*

Another important characteristic of the a-Game is that if we consider a finite automaton, we must consider that the accepted words are those of length n, at most. Thus, in an a-Game we deal with any $L(A)^n$, which is the set of words of length n.

Definition 4.4 *Let $S_I(G_A)^n$ be the set of all games on G_A where P_{II} plays according to the strategy given by Σ^n_A, or all words of size n.*

Informally, an a-Game is a game where player one is the *state writer* and the player two is the *character writer.*. In other words, player one plays any state $q \in Q$, and player two plays any character $s \in \Sigma$. In Figure 3, we show an example of an a-Game. Notice it has the size of the accepted word of the automata in 1b, where there are only three states and two possible transitions. q_2 is the final state, and q_0 is the initial state. The final move from 2, in this example, can be a, b or λ, a flag to denote that P_{II} has no more characters from $\Sigma*$.

4.1. Winning strategies and language acceptance

In order to relate the game and the automata, we must show that any winning strategies in the game is equivalent to the language accepted by A. In Definition 4.2, we unveil the concept of (general) strategy for each player. Also, in Definition 4.3, we describe a strategy for P_{II} and a word w. However, we are interested in those where P_{II} wins over P_I , which are the winning strategies of P_{II} .

Definition 4.5 *A winning strategy in the a-Game for P_{II} is a strategy w of length n, where P_I has no valid move to play and last move is $q_f \in F_A$, or the set of final states of A.*

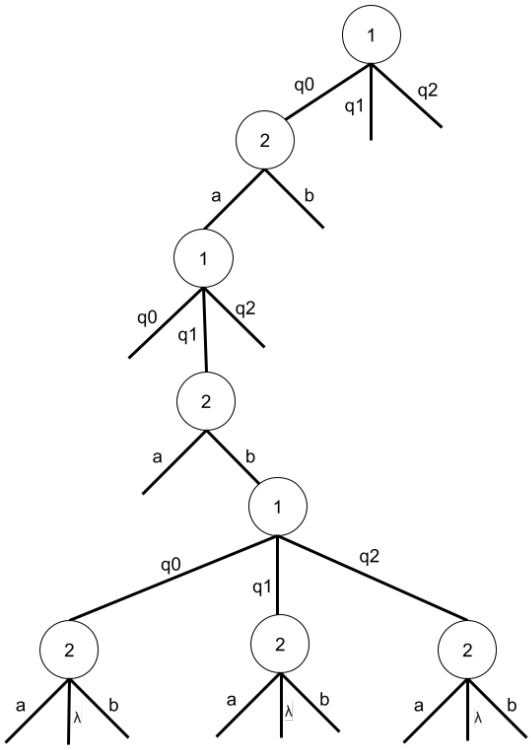


Figure 3. Example of an a-Game from the automata in Figure 2.

This definition allow us to talk about the equivalence between the winning strategies in the a-Game and the language acceptance of automaton A.

Definition 4.6 *Let $S(G_A)^n$ be the wining strategies of G_A^n and the words accepted by the automata A, $L(A)^n$.*

In essence, we need to show that given a winning strategy $s \in S(G_A)^n$, there is an application of $\hat{\delta}$ over a word w of length n, where we end in a final state of the automaton. In other words, w is recognizable by A, or $w \in L(A)^n$.

Theorem 4.1 ($S(G_A)^n = L(A)^n$)

Proof. ($S(G_A)^n \subseteq L(A)^n$) By induction on the size of the strategy. (Basis) The empty strategy. In this case, P_I plays the initial state and the second player plays empty. This means the initial state of A is a final state. Thus, the empty string belongs to $L(A)^n$. (Inductive step) Assume that $S(G_A)^n = L(A)^n$, where n is the length of a string/strategy. Let $(a_I; a_{II}) = s \in S(G_A)^n$ where a_I and a_{II} are the preceding actions from P_I and P_{II} respectively. Since $S(G_A)^n = L(A)^n$, there is a $\hat{\delta}(q, w)$ with $w \in L(A)^n$. Then, let s' be a strategy of length $n + 1$ in the form $(a_I; a_{II}; c)$, $c \in \Sigma$. If $s' \in S(G_A)^n + 1$, then by inductive hypothesis, $\hat{\delta(q, wc)} \in L(A)^{n+1}$. ($S(G_A) = L(A)$.)
($L(A)^n \subseteq S(G_A)^n$) By induction in the size of the string. (Basis) A string $w \in L(A)^1$. In this case, P_I has one move, and P_{II} also plays, which leads to a winning strategy for P_{II} . (Inductive step) Assume that $L(A)^n = S(G_A)^n$, where n is the length of the string. For $L(A)^{n+1}$, we can represent it in the form $w = aw'$. Then, $a \in L(A)$ and has a winning

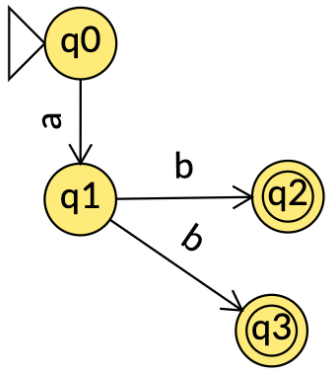


Figure 4. A simple NFA.

strategy, by the inductive step. Since $w' \in L(A) \mid_n$, then $s'_w \in S(G_A)$. $w' \in \Sigma_{L(A)}$ Since $w \in L(A)$ $s \in G(A)$.

QED

5. Ai-Game: A non-deterministic finite automaton Game with imperfection information

We also consider here the case of non-deterministic finite automata. In order to represent it, we observed that the non-determinism from the automata could be equivalent to the concept of information sets available in the imperfect information a-Game concepts. We then present in this section a description of imperfect information a-Game, based on a non-deterministic finite automaton. An a-Game with imperfect information is a game representing a NFA (non-deterministic finite automaton). It can be defined formally as follows:

- A set P with P_I playing $w \in \Sigma$ and P_{II} playing $q \in Q$
- A set of moves $S = \Sigma \times Q$
- A set I of information sets, where each information set is defined as a move from a state in the automaton where there is non-determinism, i.e., $\sigma(q, s) = \{q_1, ..., q_n\}, qi \in Q$
- $\succeq_i$: is a **preference relation** between any given transitions σ_{A_i} and $\sigma_{A_{ii}}$ where $\sigma_{A_i} \succeq \sigma_{A_{ii}}$ iff $d(\sigma_{A_i}) \leq d(\sigma_{A_i ii})$, with $d : Q \to \mathbb{N}$ as a function that given a transition, returns the smaller path from the output state to the initial state of the automata.

5.1. Example of an Ai-Game

Let us show how we can represent a NFA as an imperfect information a-Game. Figure 4 shows a simple non-deterministic automaton, whilst 5 shows the game-like representation of the game. Notice that, for each decision that leads to a non-accepting state, we place an x. The information set, in this case, is placed in the last move. In this case, that last moves that lead to a winning strategy are those that show "ok" at the end. We use λ, a flag to represent that the move where P_I is at a final state.

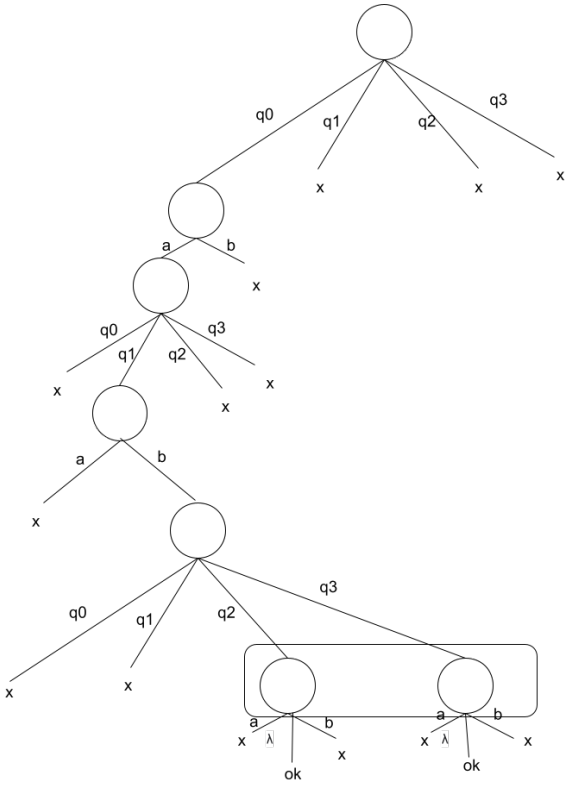


Figure 5. Representing a NFA as a game.

5.2. Winning strategy for Ai-Game NFA

As we discussed before, we also need to demonstrate the equivalence between the results shown in a NFDA and an imperfect information a-game. Intuitively, we find that the deterministic path in the imperfect information A-game is similar to the first A-game we presented. The main difference is that when reaching an information set/non-deterministic transition, P_{II} has no information regarding P_I previous move. However, the same theorem applied to an a-Game, as defined in 4.1, also applies to an Ai-Game.

6. Remarks and future works

Game theory and computer science have a common history of relationship via distinct approaches. In this paper, we attempt to discuss implications of bringing together these two theories, specially, regarding adding the concepts of knowledge and rationality [15] to (non-deterministic) automata. Specially, we are interested whether the concept of rationality can be employed to describe conflict situations in problems modeled by automata. To this end, we provided two distinct representation for different types of games and automata. We aim to explore these equivalences even further, initially exploring the relationship of more general games with more expressive kinds of automaton; and also discuss how rationality and some of its derived properties can be related to properties in these automata. Further work also aims to explore the subset construction algorithm, that allows us to convert a NFA to an FDA. We aim to explore if this relationship could be transposed to the games we created and if game theory could help us understand the pragmatic

perspective of this conversion, for instance analyzing the concept of dominant/dominated equilibrium strategies, and other types of equilibrium, such as Nash Equilibrium, and how it could be related to the language acceptance of (finite) automata.

References

[1] John E Hopcroft, Rajeev Motwani, and Jeffrey D Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA, 2006.

[2] David Park. Concurrency and Automata on Infinite Sequences. In *Proceedings of the 5th GI-Conference on Theoretical Computer Science*, pages 167–183, London, UK, UK, 1981. Springer-Verlag.

[3] Lutz Priese. Automata and Concurrency. *Theoretical Computer Science*, 25, 1983.

[4] Manfred Droste and R. M. Shortt. From Petri nets to automata with concurrency. *Applied Categorical Structures*, 10(2):173–191, 2002.

[5] Cleyton Slaviero and Edward Hermann Haeusler. Exploring Concurrency on Computational Thinking Tools. In *Anais do XIV Simpósio Brasileiro sobre Fatores Humanos em Sistemas Computacionais*, Salvador-BA, 2015.

[6] Yifat Ben-David Kolikant. Learning concurrency: Evolution of students' understanding of synchronization. *International Journal of Human Computer Studies*, 60(2):243–268, 2004.

[7] Shan Lu, Soyeon Park, Eunsoo Seo, and Yuanyuan Zhou. Learning from Mistakes - A Comprehensive Study on Real World Concurrency Bug Characteristics. In *ASPLOS'08*, page 340. Association for Computing Machinery, 2008.

[8] John Von Neumann and Oskar Morgenstern. *Theory of Games And Economic Behavior*. Princeton University Press, 60 edition, 1944.

[9] Joseph Y. Halpern. A computer scientist looks at game theory. *Games and Economic Behavior*, 45(1):114–131, 2003.

[10] Yoav Shoham. Computer science and game theory. *Communications of the ACM*, 51(8):10, 8 2008.

[11] John H. Reif. Universal games of Incomplete Information. In *11th Annual Symposium on Theory of Computing*, pages 288–308, Atlanta, GA, 1979.

[12] John H Reif. The Complexity of Two Player Games of Incomplete Information, 1984.

[13] Tim Roughgarden. Algorithmic game theory. *Communications of the ACM*, 53(7):78, 2010.

[14] Noam. Nisan. *Algorithmic game theory*. Cambridge University Press, 2007.

[15] Martin J. Osborne and Ariel Rubinstein. *A Course in Game Theory.* Number 1. MIT Press Books, 1994.

An Application for Medical Diagnosis Using Correlation Coefficient With Modal Operators and Operator Identifying and Unary

Alex Bertei[1], **Renata H. S. Reiser**[1] **and Luciana Foss**[1]

[1]Centre for Technological Development - Federal University of Pelotas (UFPEL)
96.010-610 - Pelotas - RS - Brazil

{abertei, reiser, lfoss}@inf.ufpel.edu.br

Abstract. *This paper aims to study the Atanassov's correlation coefficient (A-CC) between two of Atanassov's intuitionistic fuzzy sets (A-IFS), obtained as images of intuitionistic fuzzy modal operators. The composition of modal operators are investigated, verifying under which conditions an A-CC preserves the main properties related to conjugate and complement operations performed on A-IFS. In addition, a simulation based on the proposal methodology using operators described above is applied to a medical diagnosis analysis.*

1. Introduction

The expression of uncertainty and imprecision has been widely discussed over the years, generating several extensions to the theory introduced by Zadeh [Zadeh 1965]. Atanassov's intuitionistic fuzzy sets (A-IFS) [Atanassov 1986] are used to explain the situations where there is hesitancy in the information describing the membership and non-membership degrees of elements in fuzzy sets (FS). An A-IFS considers the informations of these degrees and also providing the hesitation (uncertainty) margin related to the Atanassov's intuitionistic fuzzy index (A-IFIx), as reported by [Szmidt et al. 2012] approach. This approach leads to a great numbers of studies, all of them are closely connected with the correlation coefficient (A-CC) [Reiser et al. 2013, Bertei et al. 2016, Bertei and Reiser 2018, Bertei et al. 2021] between two intuitionistic fuzzy sets, mainly those applied to processes of decision-making, such as clustering analysis [Meng et al. 2016], digital image processing and medical diagnosis [Bertei and Reiser 2019].

The A-CC was conceived as a strict association between two variables and defined, in the Pearse correlation coefficient case, as a linear relationship of them, assigning +1 in the case of a positive linear relationship increasing their variable values and, -1 in the case of a negative (decreasing) linear relationship. In general, the correlation coefficient describes how one variable moves in relation to another. A positive correlation indicates that the two are moving in the same direction, achieving a correlation of +1 when they are moving together. A negative correlation coefficient indicates that they move in opposite directions instead. So, the closer an A-CC is to either -1 or 1, the stronger correlation between these two A-IFSs. Therefore, when the correlation between two A-IFS is 0, there is no linear relationship between them.

This article mainly focuses on intuitionistic fuzzy modal (A-IFM) operators [Atanassov 1986] which have been systematically studied by different authors [Atanassov 2012, Dencheva 2004]. In the A-IFS theory, many interpretations for the modality can be achieved, based on the corresponding modal operator expressions.

Some algebraic and characteristic properties of these operators were already examined and discussed in [Bertei and Reiser 2018, Bertei and Reiser 2019]. This article extend the results presented in [Bertei et al. 2021], studying A-CC relating to modal operators as necessity, possibility together with the A-model operator. Based on their analytical expressions, fuzzy data analysis for a problem in medical diagnosis was considered.

This paper is organized as follows: preliminaries is described in Section 2 and 3 presents the foundations on A-CC. In Section 4, this study includes the main results based on A-CC obtained by modal operators. In the Section 5 the study includes the main results based on A-CC obtained by modal operators and Identifying and Unary operator. In the Section 6 is presents an application for the medical diagnosis. Finally, conclusions and further work are discussed in Section 7.

2. Preliminary

Firstly, a brief account on A-IFS is stated. Consider a non-empty and finite universe $\mathscr{U} = \{x_1,\ldots,x_n\}$ and the unitary interval $[0,1] = U$. An A-IFS A_I based on $\mathscr{U}$ is expressed as

$$A_I = \{(x, \mu_{A_I}(x), \nu_{A_I}(x)) : x \in \mathscr{U}\} \tag{1}$$

whenever the membership and non-membership functions $\mu_{A_I}, \nu_{A_I} : \mathscr{U} \to U$ are related by the inequality $\mu_{A_I}(x_i) + \nu_{A_I}(x_i) \leq 1$, for all $i \in \mathbb{N}_n = \{1,2,\ldots,n\}$. An intuitionistic fuzzy index (IFIx) or hesitance degree of an A-IFS A_I is given as

$$\pi_{A_I}(x_i) = 1 - \mu_{A_I}(x_i) - \nu_{A_I}(x_i). \tag{2}$$

And, the set of all above related A-IFS is denoted by $\mathscr{C}(A_I)$. Let $\tilde{U} = \{\tilde{x}_i = (x_{i1}, x_{i2}) \in U^2 : x_{i1} + x_{i2} \leq 1\}$ be the set of all intuitionistic fuzzy values such that $\tilde{x}_i$ is a pair of membership and non-membership degrees of an element $x_i \in \mathscr{U}$, i.e. $(x_{i1}, x_{i2}) = (\mu_{A_I}(x_i), \nu_{A_I}(x_i))$. And, the related IFIx is given as $\pi_{A_I}(x_i) = x_{i3} = 1 - x_{i1} - x_{i2}, \forall i \in \mathbb{N}_n = \{1,2,\ldots,n\}$. The projections $l_{\tilde{U}^n}, r_{\tilde{U}^n} : \tilde{U}^n \to U^n$ are given by:

$$l_{\tilde{U}^n}(\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_n) = (x_{11},x_{21},\ldots,x_{n1}) \qquad r_{\tilde{U}^n}(\tilde{x}_1,\tilde{x}_2,\ldots,\tilde{x}_n) = (x_{12},x_{22},\ldots,x_{n2}). \tag{3}$$

The order relation $\leq_{\tilde{U}}$ on $\tilde{U}$ is defined as: $\tilde{x} \leq_{\tilde{U}} \tilde{y} \Leftrightarrow x_1 \leq y_1$ and $x_2 \geq y_2$. Moreover, $\tilde{0} = (0,1) \leq_{\tilde{U}} \tilde{x} \leq_{\tilde{U}} (1,0) = \tilde{1}$, for all $\tilde{x} \in \tilde{U}$.

Intuitionistic fuzzy negations and **intuitionistic automorphisms** are studied in the following. See more details in [Bustince et al. 2003]. An intuitionistic fuzzy negation (A-IFNs) $N_I : \tilde{U} \to \tilde{U}$ is a function verifying:

$N_I\mathbf{1}$ $N_I(\tilde{0}) = N_I(0,1) = \tilde{1}$ and $N_I(\tilde{1}) = N_I(1,0) = \tilde{0}$;
$N_I\mathbf{2}$ If $\tilde{x} \geq_{\tilde{U}} \tilde{y}$ then $N_I(\tilde{x}) \leq_{\tilde{U}} N_I(\tilde{y})$, $\forall \tilde{x},\tilde{y} \in \tilde{U}$.

In [Bustince et al. 2000], if an IFN N_I also satisfies the involutive property

$N_I\mathbf{3}$ $N_I(N_I(\tilde{x})) = \tilde{x}$, $\forall \tilde{x} \in \tilde{U}$,

N_I is called a strong A-IFN.

According with [Deschrijver et al. 2004, Theorem 3.6], N_I is a strong A-IFNs iff there exists a strong fuzzy negation N on U such that:

$$N_I(x_1,x_2) = (N(N_S(x_2)), N_S(N(x_1))). \tag{4}$$

Thus, N_I is an example of N-representable IFN. Moreover, if N is the standard fuzzy negation $(N(x) = N_S(x) = 1 - x)$ equation (4) can be given as

$$N_{SI}(\tilde{x}) = N_{S_I}(x_1, x_2) = (x_2, x_1). \tag{5}$$

By [Bustince et al. 2004], the complement of an IFS A w.r.t. N_I in (4) is given as

$$\overline{A} = \{(x, N_I(\mu_A(x), \nu_A(x))) : x \in \mathscr{U}\}. \tag{6}$$

And, when $N = N_S$ in Eq. (4), then the complement of an IFS A w.r.t. N_{SI} is expressed as

$$\overline{A} = \{(x, \nu_A(x), \mu_A(x)) : x \in \mathscr{U}\}. \tag{7}$$

The function $f_{N_I} : \tilde{U}^n \to \tilde{U}$ is the N_I-dual operator of $f : \tilde{U}^n \to \tilde{U}$ given as follows

$$f_{N_I}(\tilde{x}_1, \ldots, \tilde{x}_n) = N_I(f(N_I(x_1), \ldots, N_I(\tilde{x}_n))). \tag{8}$$

2.1. Intuitionistic Fuzzy Modal Operators

Following [Atanassov 1983], a pair of operators, the necessity operator given as $\Box : \tilde{U} \to U$, $\Box(x_1, x_2) = (x_1, 1 - x_1)$, and the possibility operator, defined as $\Diamond : \tilde{U} \to U$, $\Diamond(x_1, x_2) = (1 - x_2, x_2)$. These two operators and their properties resemble those of Modal Logic and both can be applied to an A-IFS A_I, transforming it into a fuzzy set (FS), $\Box A_I$ and $\Diamond A_I$.

Definition 2.1. *[Atanassov 1999] Let A_I be an A-IFS. The sets $\Box A_I$-IFS and $\Diamond A_I$-IFS, related to the necessity and possibility modal operators are, respectively, given as*

$$\Box A_I = \{\langle x, \mu_{A_I}(x), 1 - \mu_{A_I}(x)\rangle | x \in \mathscr{U}\} \text{ and } \Diamond A_I = \{\langle x, 1 - \nu_{A_I}(x), \nu_{A_I}(x)\rangle | x \in \mathscr{U}\}. \tag{9}$$

When A_I is a fuzzy set ($\mu_{A_I}(x) = 1 - \nu_{A_I}(x)$) for each $x \in \mathscr{U}$, then $\Box A_I = A_I = \Diamond A_I$.

Proposition 2.1. *[Atanassov 1999, Prop. 1.42] For an A-IFS, the properties hold:*

$$\overline{\Box \overline{A_I}} = \Diamond A_I, \ \overline{\Diamond \overline{A_I}} = \Box A_I, \ \Diamond\Diamond A_I = \Diamond A_I, \ \Box\Box A_I = \Box A_I, \ \Box\Diamond A_I = \Diamond A_I, \ \Diamond\Box A_I = \Box A_I. \tag{10}$$

The operator $\triangle : \tilde{U} \to \tilde{U}$, given as $\triangle(x_1, x_2) = (x_1(1 - x_2), x_2(1 - x_1))$ can be used in Intuitionistic Fuzzy Expert Systems and Intuitionistic Fuzzy Estimations of Expert Knowledge [Atanassov 1999].

Definition 2.2. *[Atanassov 1999, Def. 1.93]The A-IFS $\triangle A_I$ is defined as*

$$\triangle A_I = \{\langle x, \mu_{A_I}(x).(1 - \nu_{A_I}(x)), \nu_{A_I}(x).(1 - \mu_{A_I}(x))\rangle | x \in \mathscr{U}\} \tag{11}$$

The A-IFS $\triangle A_I$ has the properties stated in the following theorem:

Theorem 2.1. *[Atanassov 1999, Theorem 1.94] For every A-IFS A, $\triangle\overline{A_I} = \triangle A_I$.*

In the case of fuzzy sets, we have that $\triangle A_I = \{\langle x, \mu_{A_I}(x)^2, \nu_{A_I}(x)^2\rangle | x \in \mathscr{U}\}$.

Thus, the $\triangle$-operator is similar to Zadeh's idea expressing the *very*-qualifier [Zadeh 1975]. Moreover, if $\nu_{A_I}(x) = 1 - \mu_{A_I}(x)$, $\mu_{A_I}(x)^2 + \nu_{A_I}(x)^2 = 1 - 2\mu_{A_I}(x)(1 - \mu_{A_I}(x)) \leq 1$, i.e. when we have that $\mu_{A_I}(x) > 0$, then $\triangle A_I$ is a proper IFS with the non-determinacy degreee given as $\pi_{A_I}(x) = 2\mu_{A_I}(x)(1 - \mu_{A_I}(x))$.

3. Correlation from A-IFL

Using denotation related to equations (2), (3)a and (3)b:

$$(\mu_{A_I}(x_1), \mu_{A_I}(x_2), \ldots, \mu_{A_I}(x_n)) = (x_{11}, x_{21}, \ldots, x_{n1}) = \mathbf{x}_{i1};$$
$$(\nu_{A_I}(x_1), \nu_{A_I}(x_2), \ldots, \nu_{A_I}(x_n)) = (x_{12}, x_{22}, \ldots, x_{n2}) = \mathbf{x}_{i2};$$
$$(\pi_{A_I}(x_1), \pi_{A_I}(x_2), \ldots, \pi_{A_I}(x_n)) = (x_{13}, x_{23}, \ldots, x_{n3}) = \mathbf{x}_{i3}.$$

and the two corresponding classes of the quasi-arithmetic means are reported below:
(i) the arithmetic mean (AM) related to an A-IFS A_I; and
(ii) the quadratic mean (QM) is performed over the difference between each intuitionistic fuzzy value of an A-IFS A_I and the corresponding arithmetic mean

Thus, the quotient obtained from the product performed of such AM QM extends the A-CC definition to the A-IFS approach.

Definition 3.1. *[Szmidt and Kacprzyk 2012] The A-CC between A_I and B_I in $\mathscr{C}(A_I)$,*

$$\mathsf{C}_I(A_I,B_I) = \frac{1}{3}(C_1(A_I,B_I)+C_2(A_I,B_I)+C_3(A_I,B_I)) \tag{12}$$

is given when $k \in \{1,2,3\}$ based on the following equations:

$$C_k(A_I,B_I) = \frac{\sum_{i=1}^{n}\left(x_{ik}-\frac{1}{n}\sum_{j=1}^{n}x_{jk}\right)\left(y_{ik}-\frac{1}{n}\sum_{j=1}^{n}y_{jk}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_{ik}-\frac{1}{n}\sum_{j=1}^{n}x_{jk}\right)^2\sum_{i=1}^{n}\left(y_{ik}-\frac{1}{n}\sum_{j=1}^{n}y_{jk}\right)^2}}$$

In [Szmidt and Kacprzyk 2012], the correlation coefficient $\mathsf{C}_I(A_I,B_I)$ in Eq. (12) considers both factors, the amount of reliability information expressed by: (i) the membership and non-membership degrees expressed by $C_1(A_I,B_I)$ and $C_2(A_I,B_I)$, respectively; and (ii) the hesitation margins in $C_3(A_I,B_I)$.

Additionally, these expressions just make sense for A-IFS variables whose values vary and avoid zero in the denominator. Moreover, $\mathsf{C}_I(A_I,B_I)$ fulfils the next properties:

(i) $\mathsf{C}(A_I,B_I) = \mathsf{C}(A_I,B_I)$; **(ii)** If $A_I = B_I$ then $\mathsf{C}(A_I,B_I) = 1$; **(iii)** $-1 \leq \mathsf{C}(A_I,B_I) \leq 1$.

Proposition 3.1. *[Bertei et al. 2016, Prop.1] Let N be a strong A-IFNs, A_I and B_I be A-IFS and $\overline{A_I}$ and $\overline{B_I}$ be their corresponding complements. The following holds:*

$$C_1(A_I,\overline{B_I}) = C_2(\overline{A_I},B_I); \quad C_2(A_I,\overline{B_I}) = C_1(\overline{A_I},B_I); \quad C_3(A_I,\overline{B_I}) = C_3(\overline{A_I},B_I). \tag{13}$$

Corollary 3.1. *[Bertei et al. 2016, Corollary.1] Let N be a strong A-IFNs, A_I and B_I are A-IFS and $\overline{A_I}$ and $\overline{B_I}$ be their corresponding complements. The following holds:*

$$\mathsf{C}_I(A_I,\overline{B_I}) = \mathsf{C}_I(\overline{A_I},B_I). \tag{14}$$

4. A-CC Results on Modal Operators

Extending the results presented in [Bertei et al. 2016], the article [Bertei and Reiser 2018] studies the correlation between A-IFS obtained as the image of modal level operators as $\Diamond A_I$-IFS and $\Box A_I$-IFS that are obtained by action of dual and conjugate operators on $\tilde{U}$.

Proposition 4.1. *[Bertei and Reiser 2018, Proposition IV.3.] Let A_I-AIFS and $\Diamond A_I$-IFS given in Eqs. (1) and (9b). The A-CC between A_I-IFS and $\Diamond A_I$-IFS is given as*

$$\mathsf{C}_I(A_I,\Diamond A_I) = \frac{1}{3}\left(C_1(A_I,\Diamond A_I)+1\right), \tag{15}$$

whenever the following holds

$$C_1(A_I,\Diamond A_I) = \frac{\sum_{i=1}^{n}\left(x_{i1}-\frac{1}{n}\sum_{j=1}^{n}x_{j1}\right)\left(-x_{i2}+\frac{1}{n}\sum_{j=1}^{n}x_{j2}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_{i1}-\frac{1}{n}\sum_{j=1}^{n}x_{j1}\right)^2\sum_{i=1}^{n}\left(-x_{i2}+\frac{1}{n}\sum_{j=1}^{n}x_{j2}\right)^2}}.$$

Proposition 4.2. *[Bertei and Reiser 2018, Proposition IV.3.] The correlation coefficient between an A_I-IFS and a $\overline{\Diamond A_I}$-IFS is given as*

$$\mathsf{C}_I(A_I,\overline{\Diamond A_I}) = -\mathsf{C}_I(A_I,\Diamond A_I) \tag{16}$$

Proposition 4.3. *[Bertei and Reiser 2018, Proposition IV.5.] Let A_I-IFS, $\Box A_I$-IFS and $\Diamond A_I$-IFS given by Eqs. 1, 9(a) and (b). The following holds:*

$$\mathsf{C}_I(A_I,\Box A_I) = \mathsf{C}_I(A_I,\Diamond A_I) \tag{17}$$

Proposition 4.4. *[Bertei and Reiser 2018, Proposition IV.7.] Let A_I be an A-IFS. The correlation between A-IFS A_I and $\overline{\Box A_I}$ is given as*

$$\mathsf{C}_I(A_I, \overline{\Box A_I}) = -\mathsf{C}_I(A_I, \Box A_I) \tag{18}$$

Proposition 4.5. *[Bertei and Reiser 2018, Proposition IV.9.] Let A_I-IFS, $\Diamond A_I$-IFS and $\Box A_I$-IFS given as Eqs. 1, 9(a) and (b), respectively. The following holds:*

$$\mathsf{C}_I(\Diamond A_I, \Box A_I) = \frac{1}{3}\left(C_1(A_I, \Diamond A_I) + C_2(A_I, \Box A_I)\right) \tag{19}$$

Proposition 4.6. *[Bertei and Reiser 2018, Proposition IV.11.] Let $\Box A_I$-IFS and $\Diamond A_I$-IFS given by Eqs. 9(a) and (b), respectively. The following holds:*

$$\mathsf{C}_I(\overline{\Diamond A_I}, \Box A_I) = \frac{2}{3}\left(C_2(\overline{\Diamond A_I}, \Box A_I)\right) \tag{20}$$

whenever the following expression holds

$$C_2(\overline{\Diamond A_I}, \Box A_I) = \frac{\sum_{i=1}^{n}\left(x_{i2} - \frac{1}{n}\sum_{j=1}^{n} x_{j2}\right)\left(x_{i1} - \frac{1}{n}\sum_{j=1}^{n} x_{j1}\right)}{\sqrt{\sum_{i=1}^{n}\left(x_{i2} - \frac{1}{n}\sum_{j=1}^{n} x_{j2}\right)^2 \sum_{i=1}^{n}\left(x_{i1} - \frac{1}{n}\sum_{j=1}^{n} x_{j1}\right)^2}}.$$

Proposition 4.7. *[Bertei and Reiser 2018, Prop. IV.13.] For an A-IFS A_I, we have that:*

$$\mathsf{C}_I(\overline{\Diamond A_I}, \overline{\Box A_I}) = \frac{2}{3}(C_1(A_I, \Diamond A_I)) \tag{21}$$

5. A-CC Results on Modal and $\triangle A_I$ Operators

This section reports main results of A-CC related to A-IFS, $\Box A_I$-IFS, $\Diamond A_I$-IFS, and $\triangle A_I$ obtained by the action of dual and conjugate operators on $\tilde{U}$.

Proposition 5.1. *Let $\Diamond A_I - IFS$ and the operator $\triangle A_I$ given in Equations (9b) and (11). The correlation coefficient between $\triangle A_I$ and $\Diamond(\triangle A_I)$ is given as*

$$\mathsf{C}_I(\triangle A_I, \Diamond(\triangle A_I)) = \frac{1}{3}(C_1(\triangle A_I, \Diamond(\triangle A_I)) + 1) \tag{22}$$

whenever the following holds

$$C_1(\triangle A_I, \Diamond(\triangle A_I)) = (-1)\frac{\sum_{i=1}^{n}\left((x_{i1}(1-x_{i2})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j1}(1-x_{j2}))\right)\left((x_{i2}(1-x_{i1})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j2}(1-x_{j1}))\right)}{\sqrt{\sum_{i=1}^{n}\left((x_{i1}(1-x_{i2})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j1}(1-x_{j2}))\right)^2 \sum_{i=1}^{n}\left((x_{i2}(1-x_{i1})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j2}(1-x_{j1}))\right)^2}}$$

Proof. Straightforward from Proposition 4.1. □

Proposition 5.2. *The A-CC between the operator $\triangle A_I$ and $\overline{\Diamond(\triangle A_I)} - IFS$ is given as*

$$\mathsf{C}_I(\triangle A_I, \overline{\Diamond(\triangle A_I)}) = -\mathsf{C}_I(\triangle A_I, \Diamond(\triangle A_I)) \tag{23}$$

Proof. Straightforward from Proposition 4.2. □

Proposition 5.3. *Let $\Box A_I - IFS$ and $\triangle A_I$ be operators given in Equations (9a) and (11). The correlation coefficient between $\triangle A_I$ and $\Box(\triangle A_I)$ is given as*

$$\mathsf{C}_I(\triangle A_I, \Box(\triangle A_I)) = \mathsf{C}_I(\triangle A_I, \Diamond(\triangle A_I)) \tag{24}$$

Proof. Straightforward from Proposition 4.3. □

Proposition 5.4. *The A-CC analysis between $\triangle A_I$ and $\overline{\Box(\triangle A_I)}$ operators is expressed as*

$$\mathsf{C}_I(\triangle A_I, \overline{\Box(\triangle A_I)}) = -\mathsf{C}_I(\triangle A_I, \Box(\triangle A_I)) \tag{25}$$

Proof. Straightforward from Proposition 4.4. □

Proposition 5.5. *Let $\Box A_I - IFS$, $\Diamond A_I - IFS$ and $\triangle A_I$ be operators given in Equations (9a), (9b) and (11). The A-CC between $\Box(\triangle A_I)$ and $\Diamond(\triangle A_I)$ is given as*

$$\mathsf{C}_I(\Box(\triangle A_I), \Diamond(\triangle A_I)) = \frac{1}{3}(C_1(\triangle A_I, \Diamond(\triangle A_I)) + C_2(\triangle A_I, \Box(\triangle A_I))) \tag{26}$$

Proof. Straightforward from Proposition 4.5. □

Proposition 5.6. *Let $\Box A_I - IFS$, $\Diamond A_I - IFS$ and $\triangle A_I$ be operators given in Equations (9a), (9b) and (11), respectively. The next results are verified:*

$$\mathsf{C}_I(\Box(\triangle A_I), \overline{\Diamond(\triangle A_I)}) = \frac{2}{3}(C_2(\Box(\triangle A_I), \overline{\Diamond(\triangle A_I)}) \tag{27}$$

whenever the following expression holds

$$C_2(\Box(\triangle A_I), \overline{\Diamond(\triangle A_I)}) = \frac{\sum_{i=1}^{n}\left((x_{i2}(1-x_{i1})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j2}(1-x_{j1}))\right)\left((x_{i1}(1-x_{i2})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j1}(1-x_{j2}))\right)}{\sqrt{\sum_{i=1}^{n}\left((x_{i2}(1-x_{i1})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j2}(1-x_{j1}))\right)^2 \sum_{i=1}^{n}\left((x_{i1}(1-x_{i2})) - \frac{1}{n}\sum_{j=1}^{n}(x_{j1}(1-x_{j2}))\right)^2}}$$

Proof. Straightforward from Proposition 4.6. □

Proposition 5.7. *Let $\Box A_I - IFS$, $\Diamond A_I - IFS$ and $\triangle A_I$ be operators given in Equations (9a), (9b) and (11), respectively. The following expression holds:*

$$\mathsf{C}_I(\overline{\Box(\triangle A_I)}, \overline{\Diamond(\triangle A_I)}) = \frac{2}{3}(C_1(\triangle A_I, \Diamond(\triangle A_I)) \tag{28}$$

Proof. Straightforward from Proposition 4.7. □

6. MADM - MEDICAL DIAGNOSIS

Previous analytical expressions of modal operator A-CC are applied in developing a method to medical diagnosis (MADM-MD) which is adapted from [Xu 2006]) related to a medical knowledge base, providing a proper diagnosis $D = \{$*Viral fever (VF) , Malaria (Ma) , Typhoid (Ty) , Stomach problem (SP) , Chest problem (CP)*$\}$ for a patient with the given symptoms $S = \{$*temperature (T), headache (H), stomach pain (SPa), cough (C), chest pain (CPa)*$\}$ described in terms of A-IFSs. One methodology is applied where uses necessity, possibility modal operators and the operator $\triangle$ in MADM-MD.

The necessity modal operator $(\Box)$ in the Eq. (9a) together with the operator $(\triangle)$ and $\Box(\triangle T1)$ are applied to values from Table 1 (T1) in [Xu 2006], resulting in values of Table 1. In addition, each symptom is described by its related membership and non-membership degrees. The possibility modal operator $(\Diamond)$ in the Eq. (9b) along with operator $(\triangle)$ and $\Diamond(\triangle T2)$ are applied to values of Table 2 (T2) in [Xu 2006], and symptom results are described in Table 2. In this case study, the set of patients is given as follows: P = $\{$*Al, Bob, Joe, Ted*$\}$. So, we need to seek a diagnosis for each patient p_i, for $i = 1,2,3,4$.

Thus, the A-CC in Eq. (12) is calculated considering Tables 1 and 2, deriving a diagnosis for each patient. The method is performed applying the A-CC between the operators $\Box(\triangle T1)$ and $\Diamond(\triangle T2)$. Such results are listed in Table 3.

Based on the arguments in Table 3, for the method the diagnosis is given as follows: Al suffers from Viral fever, Bob from a stomach problem, Joe from Typhoid, and Ted from Viral ferver. Additionally, one can observe that in Xu [Xu 2006], the diagnosis is the same in three patients (Bob, Joe and Ted) and it is differs in other (Al), in Bertei [Bertei and Reiser 2019]) the diagnosis is similar in Al, Bob and Joe and differs in Ted. The difference in the results are justified by use distinct methodologies.

Table 1. Symptoms characteristic for the diagnoses

	VF	***Ma***	***Ty***	***SP***	***CP***
T	(0.4,0.6)	(0.7,0.3)	(0.21,0.79)	(0.03,0.97)	(0.02,0.98)
H	(0.15,0.85)	(0.08,0.92)	(0.54,0.46)	(0.12,0.88)	(0,1)
SPa	(0.03,0.97)	(0,1)	(0.06,0.94)	(0.8,0.2)	(0.04,0.96)
C	(0.28,0.72)	(0.7,0.3)	(0.08,0.92)	(0.06,0.94)	(0.04,0.96)
CPa	(0.03,0.97)	(0.02,0.98)	(0.01,0.99)	(0.06,0.94)	(0.72,0.28)

Table 2. Symptoms characteristic for the patient

	T	***H***	***SPa***	***C***	***CPa***
Al	(0.98,0.02)	(0.96,0.04)	(0.36,0.64)	(0.96,0.04)	(0.46,0.54)
Bob	(0.2,0.8)	(0.76,0.24)	(0.96,0.04)	(0.37,0.63)	(0.28,0.72)
Joe	(0.98,0.02)	(0.98,0.02)	(0.4,0.6)	(0.44,0.56)	(0.5,0.5)
Ted	(0.96,0.04)	(0.8,0.2)	(0.72,0.28)	(0.94,0.06)	(0.72,0.28)

Table 3. Resulting A-CC of symptoms for each patient

	VF	***Ma***	***Ty***	***SP***	***CP***
Al	**0,562**	0,486	0,399	-0,459	-0,360
Bob	-0,385	-0,422	0,188	**0,545**	-0,265
Joe	0,357	0,143	**0,542**	-0,314	-0,233
Ted	**0,652**	0,651	0,094	-0,373	-0,345

7. Conclusion

In this article, the analytical expressions of A-CC were related to modal operators as necessity and possibility along with the their composition with the A-model operator. The case study presents an alternative to evaluate patients, providing a diagnosis for each patient based on the interpretation provided by A-IFM compositions.

Further work considers the A-CC analysis based on the Atanassov's interval-valued intuitionistic fuzzy sets, including the study of main connectives together with their dual and conjugation operators.

References

Atanassov, K. (1983). Intuitionistic fuzzy sets. vii itkr session. sofia. *Centr. Sci.-Techn. Library of Bulg. Acad. of Sci.*, 1697(84).

Atanassov, K. T. (1986). Intuitionistic fuzzy sets. *Fuzzy Sets Systems*, 20(1):87–96.

Atanassov, K. T. (1999). *Intuitionistic Fuzzy Sets: Theory and Applications*. Studies in Fuzziness and Soft Computing. Physica-Verlag HD, Heidelberg, Germany, Germany.

Atanassov, K. T. (2012). *On Intuitionistic Fuzzy Sets Theory*, volume 283 of *Studies in Fuzziness and Soft Computing*. Springer.

Bertei, A. and Reiser, R. (2018). Correlation coefficient analysis performed on duality and conjugate modal-level operators. In *2018 IEEE International Conference on Fuzzy Systems*.

Bertei, A. and Reiser, R. (2019). Correlation coefficient of modal level operators: An application to medical diagnosis. In *11th International Joint Conference on Computational Intelligence (FCTA 2019)*, pages 1–9.

Bertei, A., Reiser, R. H. S., and Foss, L. (2021). *Correlation Analysis Via Intuitionistic Fuzzy Modal and Aggregation Operators*, pages 163–190. Springer International Publishing, Cham.

Bertei, A., Zanotelli, R., Cardoso, W., Reiser, R., Foss, L., and Bedregal, B. (2016). Correlation coefficient analysis based on fuzzy negations and representable automorphisms. In *2016 IEEE International Conference on Fuzzy Systems*, pages 127–132.

Bustince, H., Barrenechea, E., and Mohedano, V. (2004). Intuicionistic fuzzy implication operators - an expression and main properties. *Uncertainty, Fuzziness and Knowledge-Based Systems*, 12:387–406.

Bustince, H., Burillo, P., and Soria, F. (2003). Automorphisms, negations and implication operators. *Fuzzy Sets and Systems*, 134(2):209 – 229.

Bustince, H., Kacprzyk, J., and Mohedanoi, V. (2000). Tntuitionistic fuzzy generators, application to intuitionistic fuzzy complementation. *Fuzzy Sets and Syst*, 114:485–504.

Dencheva, K. (2004). Extension of intuitionistic fuzzy modal operators ($\boxplus$) and ($\boxtimes$). In *II International IEEE Conference*, volume 3, pages 21–22.

Deschrijver, G., Cornelis, C., and Kerre, E. (2004). On the representation of intuitionistic fuzzy t-norms and t-conorms. *IEEE Transactions Fuzzy Systems*, 1(12):45–61.

Meng, F., Wang, C., Chen, X., and Zhang, Q. (2016). Correlation coefficients of interval-valued hesitant fuzzy sets and their application based on the shapley function. *International Journal of Intelligence Systems*, 31(1):17–43.

Reiser, R., Visintin, L., Benítez, I., and Bedregal, B. (2013). Correlations from conjugate and dual intuitionistic fuzzy triangular norms and conorms. In *2013 Joint IFSA World Congress and NAFIPS Annual Meeting*, pages 1394–1399.

Szmidt, E. and Kacprzyk, J. (2012). A new approach to principal component analysis for intuitionistic fuzzy data sets. In et al., S. G., editor, *CCIS – IPMU 2012*, volume 298, pages 529–538. Springer.

Szmidt, E., Kacprzyk, J., and Bujnowski, P. (2012). Correlation between intuitionistic fuzzy sets: Some conceptual and numerical extensions. In *2012 IEEE International Conference on Fuzzy Systems*. p.1-7.

Xu, Z. (2006). On correlation measures of intuitionistic fuzzy sets. In *Intelligence Data Engineering and Automated Learning – IDEAL 2006*, pages 16–24.

Zadeh, L. (1965). Fuzzy sets. *Information Control*, 8(3):338–353.

Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning. *Information Sciences*, 8(3):199 – 249.

Aplicando Análise Consensual Fuzzy para Tomada de Decisão na Alocação de Recursos em Nuvem Computacionais

Guilherme Schneider[1], **Bruno Moura**[1], **Eduardo Monks**[1], **Adenauer Yamin**[1], **Helida Santos**[2], **Renata Reiser**[1]

[1]Universidade Federal de Pelotas, Pelotas-RS, Brasil

{gbschneider,bmpdmoura,emmonks,adenauer,reiser}@inf.ufpel.edu.br

[2]Universidade Federal do Rio Grande, Rio Grande-RS, Brasil

{helida}@furg.br

Abstract. *This paper explores consensus measures to support to determining the level of use of physical machines in a cloud computing environment. The fuzzy approach considers the uncertainties present in the cloud computing environment and the consensus analysis of the fuzzy sets is based on arithmetic and exponencial means. In the proposal evaluation, a case study aimed at the architecture of the* Framework Int-FLBCC*, which is under development by the research group was designed.*

Resumo. *Este artigo contempla a concepção de uma abordagem que explora medidas de consenso fuzzy, como suporte a problemas de tomada de decisão relacionados à determinação do nível de utilização das máquinas físicas em um ambiente de computação em nuvem. A análise de consenso dos conjuntos fuzzy está baseada nas médias aritméticas e exponenciais. Para avaliar a proposta foi concebido um estudo de caso direcionado a arquitetura do* Framework Int-FLBCC *em desenvolvimento pelo grupo de pesquisa.*

1. Introdução

As infraestruturas computacionais geralmente usadas para execução de aplicações na Computação em Nuvem (CN) utilizam recursos como *data centers* que são conhecidos por consumir grande quantidade de energia elétrica, incrementando o custo operacional de provedores, e contribuindo negativamente com o meio ambiente [Gourisaria et al. 2020]. De acordo com o relatório do Conselho de Defesa dos Recursos Naturais NRDC[1] dos EUA, em 2014, somente os *data centers* nos EUA consumiram uma estimativa de 70 bilhões de quilowatts/hora kWh, representando cerca de 1,8% do consumo total de eletricidade dos EUA [Shehabi et al. 2016]. Com base nestes dados, estima-se que somente os *data centers* dos EUA consumiram aproximadamente 73 bilhões de kWh em 2020.

Neste contexto, a demanda por eficiência energética sem perda de desempenho promove a introdução de novos conceitos com especial destaque na CN, reestruturando a distribuição dos serviços de computação (alocação de servidores, armazenamento, banco de dados) em que não há a necessidade do usuário fazer grande investimento em equipamentos, e ainda, com custo/pagamento associado ao tempo de uso [Nathani et al. 2012].

[1]https://www.nrdc.org/

Destacam-se ainda os principais desafios da CN são: (i) provisionamento automático de serviço; (ii) migração de máquinas virtuais (MV); (iii) consolidação de servidores, (iv) gerenciamento de energia, e ainda, (v) segurança de dados [Zhang et al. 2010].

Considerando estes novos cenários e a investigação em trabalhos relacionados na literatura [Moura et al. 2021], este estudo contempla a migração e consolidação de MV com uma abordagem para utilização eficiente dos servidores físicos do ambiente da CN. O objetivo central deste trabalho é a concepção de uma abordagem que explore as premissas e extensões da Lógica Fuzzy (*Fuzzy Logic - FL*), tratando as incertezas dos especialistas na especificação de suas preferências quanto a definição das funções de pertinência, focando nas medidas de consenso de conjuntos fuzzy e assim, contribuindo no suporte à tomada de decisão para a avaliação de cargas em servidores, e a decorrente alocação de MV, considerando o componente *Int-FLBCC (**Int**erval **F**uzzy **L**oad **B**alancing for **C**loud **C**omputing)* [Moura et al. 2021].

O artigo está estruturado em cinco seções. A Seção 1 trata dos fundamentos contextuais do trabalho. Na Seção 2 apresenta-se os conceitos da Lógica Fuzzy Tipo-2. A Seção 3, expõe a modelagem do componente *Int-FLBCC*, detalhando a base de dados, fuzzificação, base de regras, inferência e defuzzificação. A Seção 4 descreve o método de medidas de consenso entre conjuntos fuzzy concebido para avaliação da modelagem *Int-FLBCC*. E, por fim, na Seção 5, constam as conclusões e trabalhos futuros.

2. Lógica Fuzzy

Lotf Zadeh introduziu a Lógica Fuzzy Tipo-2 (*Type-2 Fuzzy Logic - T2FL*) em 1975 como extensão da FL [Mendel et al. 2014]. Seu surgimento está relacionado com a insuficiência da teoria dos Conjuntos Fuzzy (*Fuzzy Sets - FS*) tradicionais na modelagem das incertezas inerentes à definição das funções de pertinência dos antecedentes e consequentes em um Sistema de Inferência Fuzzy (FIS) [Mendel 2003].

Seja o conjuto-universo $\chi \neq \emptyset$. Um conjunto fuzzy $X = \{(u, \mu_X(u)) : \mu_X(u) \in [0,1], u \in \chi\} \in \mathcal{F}_\chi$ está definido pela função $\mu_A : \chi \to [0,1]$, determinando o valor fuzzy $\mu_X(u)$ que indica o grau de pertinência de $u \in \chi$ em X. O conjunto de todos os valores fuzzy será indicado por $\mathcal{F}_\chi$, onde $\emptyset, \chi \in \mathcal{F}_\chi$, sendo $\mu_\emptyset(u) = 0$ e $\mu_\chi(u) = 1$, $\forall u \in \chi$, respectivamente. Sendo o conjunto de todos os conjuntos fuzzy indicado por $\mathcal{F}_\chi$.

Na extensão desta abordagem multi-valorada, considera-se a Lógica Fuzzy Intervalar Tipo-2 (*Interval Type-2 Fuzzy Set Logic* - IT2FL). Particularmente, a semântica considerada provê a interpretação do grau de pertinência intervalar do elemento $u \in \chi$ em um conjunto fuzzy A, como um valor no intervalo de pertinência $\mu_A(x) \subseteq [0,1]$. Nesta interpretação, não é possível precisar o valor exato, apenas fornecemos limites (superior e inferior) correspondendo aos extremos do seu intervalo de pertinência.

Esta interpretação se refere a imprecisão dos especialistas para alcançar um consenso quando da atribuição dos graus de pertinência aos múltiplos atributos considerados no sistema modelado. Neste contexto, uma das métricas que avalia a imprecisão nas diferentes atribuições de especialistas está identificada como o diâmetro do correspondente grau intervalar de pertinência.

Definição 1 *[Karnik and Mendel 1998] Seja $\chi \neq 0$ um universo. Um T2FS A é caracterizado por uma Função de Pertinência do Tipo-2 (*Membership Function Type-2 - T2MF*)*

onde $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$, $x \in \chi$ *e* $u \in J_x \subseteq [0,1]$. *Para cada T2FS* $\tilde{A}$ *tem-se*

$$\tilde{A} = \{((x,u), \mu_{\tilde{A}}(x,u)) \,|\, \forall x \in \chi, \forall u \in J_x \subseteq [0,1]\}.$$

Um T2FS atribui a um elemento no universo χ um mapeamento $A(x) : [0,1] \rightarrow [0,1]$, e pode ser definido como $\{(x, A(x,t)) : x \in \chi, t \in [0,1]\}$ quando $A(x,.) : [0,1] \rightarrow [0,1]$ é dado como $A(x,t) = A(x)(t)$, para cada $x \in \chi$, $t \in [0,1]$. Em particular, $A(x)$ é um número real em $[0,1]$, para cada $x \in \chi$. E ainda, se $\mu_{\tilde{A}}(x) = 1$ então A é um conjunto fuzzy intervalar $A(x) = \{(u,1) : u \in J_x \subseteq [0,1]\}$, $\forall x \in \chi$. [Mendel et al. 2006]

Observa-se que os conjuntos fuzzy valorados por intervalos [Gehrke et al. 1996] é um caso particular de T2FS. Seja A um T2FS $A(x) = [\underline{A}(x), \overline{A}(x)]$, $\forall x \in \chi$. Além disso, sejam $A, B \in T2FS$. Para todo $x \in \chi$, tem-se os operadores:
(i) Complemento: $A_C(x) = [1 - \overline{A}(x), 1 - \underline{A}(x)]$;
(ii) União: $A(x) \cup B(x) = [\max(\underline{A}(x), \underline{B}(x)), \max(\overline{A}(x), \overline{B}(x))]$;
(iii) Intersecção: $\mu_{A \cap B}(x) = [\min(\underline{A}(x), \underline{B}(x)), \min(\overline{A}(x), \overline{B}(x))]$.

Denotando, $A(x) = X$, $B(x) = Y$, $\forall x \in \chi$, U como o conjunto de todos os subintervalos reais no intervalo unitário $[0,1]$ e $\mathbb{U}$ como o conjunto dos valores fuzzy intervalares, a ordem "$\leq$"parcial em $\mathbb{U}$ é a *Ordem Produto* [Klement et al. 2004] dada como: $X \leq Y \Leftrightarrow \underline{X} \leq \underline{Y} \wedge \overline{X} \leq \overline{Y}$.

As funções que qualificam as intersecções e uniões fuzzy são modeladas neste trabalho por normas e conormas triangulares, respectivamente. Segundo [Mendel et al. 2006] e considerando os intervalos em $\mathbb{U}$, tem-se que:

- Uma norma triangular (t-norma) intervalar é uma função $\mathbb{T} : \mathbb{U}^2 \rightarrow \mathbb{U}$ que satisfaz as propriedades de comutatividade, associatividade, monotonicidade e tem o $\mathbb{1} \in U$ como elemento neutro. Alguns exemplos de t-normas mais utilizadas são: Intersecção Padrão, Produto Algébrico, Intersecção Drástica, Lukasiewicz e Nilpotente Mínimo.
- Uma conorma triangular (t-conorma) intervalar é uma função binária $\mathbb{S} : \mathbb{U}^2 \rightarrow \mathbb{U}$ que satisfaz as propriedades de comutatividade, associatividade, monotonicidade e tem o $\mathbb{0} \in U$ como elemento neutro. Alguns exemplos de t-conormas são: União Padrão, Soma Probabilística, União Drástica, Lukasiewicz e Nilpotente Máximo.

Um sistema baseado em T2FL pode estimar funções de entrada e saída, por meio do uso de heurísticas e técnicas intervalares. A seguir, os blocos que constituem um Sistema de Inferência Fuzzy Tipo-2 (*Type-2 Fuzzy Inference System* - T2FIS) são descritos:
(1) Interface de Fuzzificação: O processo de fuzzificação baseado em T2FS é realizado de acordo com a natureza e definição de um FST2, associando um valor de entrada com uma função intervalar e não simplesmente com um único valor em U. Em outras palavras, é inserido no mecanismo de inferência a incerteza relacionada as funções de pertinência de entrada. Assim, para cada entrada $A(x)$ um vetor de entrada $\mathbf{x} = (x_1, x_2, \ldots, x_n) \in \chi^n$ quando $n \in \mathbb{N}^*$ está relacionado a um par de vetores em $\mathbb{U}^n$ obtidos da seguinte forma:

$$\left(\overline{A}(x_1), \overline{A}(x_2), \ldots, \overline{A}(x_n)\right), \left(\underline{A}(x_1), \underline{A}(x_2), \ldots, \underline{A}(x_n)\right).$$

(2) Base de Regras (*Rule Base - RB*): constituída por regras que classificam as Variáveis Linguísticas (VL) de acordo com os T2FS valorados por intervalos;

(3) Unidade de Decisão Lógica: realiza as operações de inferência entre os dados de entrada e as condições impostas na RB, gerando a ação a ser realizada no T2FIS;
(4) Defuzzificação: nesta fase, são consideradas duas principais etapas, que são elas:

(i) **Redutor de Tipo**: responsável por reduzir T2FS em FS, ao buscar o melhor FS que representa o T2FS deve satisfazer a seguinte premissa: quando toda a incerteza desaparecer, o resultado do T2FIS reduz-se a um FIS [Wu and Nie 2011].
(ii) **Defuzzificação**: o T2FIS usa a média dos pontos limites $\underline{y}$ e $\overline{y}$ da saída $B(x)$:

$$y = \frac{\underline{Y} + \overline{Y}}{2} = \frac{\underline{B}(x) + \overline{B}(x)}{2}, \forall x \in \chi, \tag{1}$$

onde os valores $\underline{y}$ e $\overline{y}$ são calculados via método iterativo de Karnik e Mendel (algoritmo KM), ou obtido através do uso de um método convencional, como o centroide, no valor final da inferência.

3. Int-FLBCC: Modelagem do Sistema Fuzzy Tipo-2

O *Int-FLBCC* é responsável por verificar o Nível de Utilização (U) da máquina física relacionado ao balanceamento de carga na CN, considerando uma RB atuando em três etapas: Fuzzificação, Inferência e Defuzzificação retornando como saída U de cada máquina física do ambiente da CN, a partir do tratamento das variáveis Poder Computacional (PC), Custo de Comunicação (CC) e *Random Access Memory* (RAM).
Definição das Funções de Pertinência, transformando as VLs relativas a cada uma das variáveis de incerteza em FST2, através do estudo das variáveis junto a especialistas. Aplica-se a forma trapezoidal na representação gráfica das suas funções de pertinência. Nesta trabalho, a leitura das configurações aplicadas no ambiente de CN simulado é realizada na mensuração dos três atributos PC, CC e RAM. Estes valores são aplicados a uma escala padrão adotada considerando o intervalo $[0;10]$, veja Tabela 1:

Tabela 1. Parâmetros da Escala Padrão

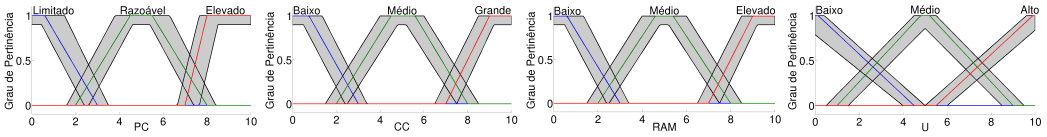
Parâmetro	Descrição
h_i	representando o $host(i)$ do ambiente da nuvem;
MM	Máximo de MIPS disponível no host i considerando todos os *Processing Elements* (PE);
$UtoB$	representando a utilização de largura de banda do host i;
$UtoR$	o uso de memória RAM no host i;
$MaxPC$	valor total em MIPS do melhor host do ambiente da nuvem;
$MinCC$	o menor custo de comunicação no ambiente da nuvem;
$MaxRAM$	o valor de RAM do melhor host.

Na obtenção dos correspondentes graus de pertinência tem-se os parâmetros:

$$PC = (h_i(MM)/MaxPC * 10) \tag{2}$$
$$CC = ((10 * h_i(UtoB))/MinCC) \tag{3}$$
$$RAM = (h_i(UtoR)/MaxRam) * 10 \tag{4}$$

associadas as T2FS das Figuras 1(a), 1(b), 1(c) e 1(d), modelando as respectivas variáveis PC, CC, RAM e U. A seguir, os termos linguísticos (TL) para definição dos FST2:
(i) FST2 da variável PC são os seguintes: "Limitado" (PCL), "Razoável" (PCR) e "Elevado" e considerando (PCE - como melhor caso). Sendo $PC = a$ e $a \in [0;10]$, obtém-se as FPT2 apresentadas em modo gráfico na Figura 1(a).

(ii) FST2 da variável CC são: "Pequeno" (CCP - melhor caso), "Médio" (CCM) e "Grande" (CCG). Sendo $CC = b$ e $b \in [0;10]$, têm-se as FPT2 da Figura 1(b).
(ii) FST2 da variável RAM são: "Baixo" (RAMB - melhor caso), "Médio" (RAMM) e "Grande" (RAMG). E, para $RAM = c$ e $c \in [0;10]$, têm-se as FPT2 da Figura 1(c).
(iii) FST2 da variável U usados nesse caso são: "Baixa" (UB), "Média" (UM) e "Alta" (UA). Sendo $U = d$ e $d \in [0;10]$. O nível de utilização U das máquinas também é adaptado para escala padrão conforme mostra a Figura 1, de (a) até (d).



(a) (b) (c) (d)

Figura 1. PC, CC, RAM e U na escala padrão

No ambiente da CN, a incerteza e imprecisão no nível de uso das máquinas físicas decorre da flutuação do poder computacional, largura de banda e memória disponível no momento da execução das aplicações submetidas pelos usuários. Considera-se um algoritmo online aplicando abordagem da T2FS que percorre as máquinas físicas disponíveis na arquitetura da CN obtendo o nível de uso a cada iteração. Seguem-se as demais etapas:
Fuzzificação, ocorrendo o mapeamento dos valores de entrada não fuzzy (já ajustados para escala observada) para o domínio fuzzy.
Base de Regras, considerando três fatores: (i) as VL nomeiam os FS, tornando a modelagem do sistema mais próxima do mundo real; (ii) são utilizadas conexões lógicas do tipo "AND" para criar a relação entre as variáveis de entrada; (iii) as implicações são do tipo *modus ponens* (modo afirmativo):

$$\text{Se } \text{“}x_1 \text{ é } A_1\text{”} \; E \; \text{“}x_2 \text{ é } A_2\text{”} \; E \; \text{“}x_3 \text{ é } A_3\text{”} \text{ então “}y \text{ é } B\text{”}$$

Inferência, modelando as operações entre os FS, combinação dos antecedentes das regras e implicações via o operador *modus ponens generalizado*. Esta fase ocorre em três etapas:
(i) Aplicação da Operação Fuzzy "AND" através do método MIN (mínimo);
(ii) Aplicação do Método de Implicação Fuzzy, utilizando o método MIN (mínimo);
(iii) Aplicação do Método de Agregação Fuzzy, utilizando o método MAX (máximo).
Defuzzificação transformação da região resultado da inferência, a técnica utilizada foi o Centro da Área. Esse método calcula o centroide (x) da área composta pela saída T2FIS definido pela seguinte fórmula:

$$u = \frac{\sum_{i=1}^{N} u_i \mu_{OUT}(u_i)}{\sum_{i=1}^{N} \mu_{OUT}(u_i)}$$

4. Medidas de Consenso

Medidas de consenso são estudadas em contextos de tomada de decisão [Beliakov et al. 2014] como um fator chave em qualquer problema de tomada de decisão em grupo [Tsuchiya and Hiramoto 2018], abordando o uso de agregações [Oliveira et al. 2021]. Em [Beliakov et al. 2014, Definição 3] foi realizado um estudo baseado: (i) na unanimidade, interpretando o consenso completo; (ii) no consenso mínimo, sempre que as entradas se situam em extremos (0 e 1) no intervalo unitário.

Definição 2 *A função* $C : \bigcup_{i=1}^{\infty}[0,1]^n \to [0,1]$ *define uma medida de consenso fuzzy (FCM) sempre que as duas seguintes propriedades são satisfeitas:*

$(C1)$ $x_1 = \ldots = x_n \Rightarrow C(x_1,...,x_n) = 1$, $\forall x_1,\ldots x_n \in [0,1]$ *(unanimidade);*
$(C2)$ $C(0,1) = C(1,0) = 0$ *(consenso mínimo para* $n = 2$*).*

Seja $\chi = \{u_1,\ldots,u_{100}\}$ e $x_i = \mu_X(u_i) \in [0,1]$, $\forall i \in \mathbb{N}_1 00$, os valores fuzzy X. Dos resultados apresentados em [Oliveira et al. 2021], definem-se as medidas de consenso, via funções de agregação estendida. No caso, as médias aritmética e exponencial:

$$C_{AM}(X) = \frac{1}{n}\sum_{i=1}^{n} 1 - |x_{(i)} - x_{(n-i+1)}| \quad \text{and} \quad C_{\exp_\alpha}(X) = \frac{1}{\alpha}\ln\sum_{i=1}^{n}\frac{e^{1-|x_{(i)}-x_{(n-i+1)}|}}{n}.$$

Numa segunda abordagem, para análise de consenso entre dois ou mais conjuntos fuzzy, as propriedades $C1$ e $C2$ são estendidas de $[0,1]$ para $\mathcal{F}_\chi$.

Definição 3 *Para* $\chi = \{x_1,\ldots,x_m\} \in \mathcal{F}_\chi$*, a função* $\mathcal{C} : \bigcup_{n=1}^{\infty}(\mathcal{F}_\chi)^n \to [0,1]$ *é uma medida de consenso de conjuntos fuzzy (*$\mathcal{L}_{\mathcal{F}_\chi}$*-FSCM) em* $\mathcal{F}_\chi$ *se satisfaz as condições:*
$\mathcal{C}1$*:* $\mathcal{C}(X,\ldots,X) = 1$*,* $\forall X \in \mathcal{F}_\chi$*;*
$\mathcal{C}2$*:* $\mathcal{C}(X_\chi, X_\emptyset) = \mathcal{C}(X_\emptyset, X_\chi) = 0$*,* $\forall u \in \chi$*.*

Proposição 1 *Seja* $C : \bigcup_{n=1}^{\infty}[0,1]^n \to [0,1]$ *seja um* $\mathcal{L}_{[0,1]}$*-FCM. Função* $\mathcal{C}_C : \bigcup_{n=1}^{\infty}(\mathcal{F}_{\chi_m})^n \to [0,1]$ *na Eq.(5) é uma* $\mathcal{L}_{\mathcal{F}_\chi}$*-FSCM:*

$$\mathcal{C}_C(X_1,\ldots,X_n) = \frac{1}{m}\sum_{i=1}^{m} C(\mu_{X_1}(u_i),\ldots,\mu_{X_n}(u_i)). \tag{5}$$

Definição 4 *[Beliakov et al. 2014, Definição 11, 12] Considerando a média aritmética da distância entre os pares, a função* $C^d_{SK} : \bigcup_{i=1}^{\infty}[0,1]^n \to [0,1]$*, é definida pela expressão:*

$$C^d_{SK}(x_1,\ldots,x_n) = 1 - \frac{2}{n^2}\sum_{\forall i,j | i\neq j}^{n} d(x_i,x_j); \tag{6}$$

é uma medida de consenso que satisfaz a monotinicidade em relação a maioria, e $\mathcal{L}_{[0,1]}$*-RDF definida por* $d : [0,1]^2 \to [0,1]$*, onde* $d(x,y) = (x-y)^2$*.*

Definição 5 *A função* $C_{Tastle} : \bigcup_{i=1}^{\infty}[0,1]^n \to [0,1]$ *dada por*

$$C_{Tastle}(x_1,\ldots,x_n) = 1 + \frac{1}{n}\sum_{i=1}^{n}\log_2(1 - |x_i - \bar{x}|) \tag{7}$$

define uma medida de consenso relacionada à média aritmética sobre o operador logaritmo aplicada à diferença entre 1 *e a* $\mathcal{L}_{[0,1]}$*-RDF* $d : [0,1]^2 \to [0,1]$ *definida como* $d(x,y) = |x-y|$*. Ambos os operadores* C^d_{SK} *e* C_{Tastle} *verificam [Beliakov et al. 2014, Definição 11] em relação a* $\mathcal{L}_{[0,1]}$*-RDF d.*

Cada método de consenso C_M é executado sobre os dados relativos a VL (B, M, A) relacionadas aos conjuntos fuzzy de *PC*, *CC* e *RAM*, incluindo também U, a variável de saída. Usamos $\alpha = 1$ e $n = 100$ para realizar a análise de consenso fuzzy.

Os resultados para a análise do caso pontual são apresentados na Tabela 2 e Figura 2. Na função $\mathcal{C}_{AM}$ o menor valor de consenso foi de 0.5000, e o maior 0.9938. Já na avaliação da função $C_{\exp_\alpha}$ o menor valor foi 0.5560, e maior 0.9938. No caso da função $\mathcal{C}_{SK}$ o menor valor de consenso foi de 0.3060, e o maior 0.6020. E por último, na execução de $\mathcal{C}_{Tastle}$ 0.2375 e 0.4988, o menor e o maior valor respectivamente.

Os melhores resultados foram atingidos com a aplicação da função $\mathcal{C}_{AM}$ e $C_{\exp_\alpha}$ na avaliação das funções de pertinência que definem os termos linguísticos médios em todas as variáveis.

Tabela 2. Resultado Numérico

X	$\mathcal{C}_{AM}$	$C_{\exp}$	$\mathcal{C}_{SK}$	$\mathcal{C}_{Tastle}$
PC_L	0.6600	0.7245	0.6020	0.4988
PC_M	0.9800	0.9803	0.3060	0.2519
PC_H	0.5100	0.5956	0.3061	0.2375
RAM_L	0.6351	0.7070	0.5533	0.4530
RAM_M	0.9938	0.9938	0.5533	0.2837
RAM_H	0.5900	0.6717	0.4811	0.3874
PC_L	0.6351	0.7070	0.5533	0.4530
PC_M	0.9938	0.9938	0.3457	0.2837
PC_H	0.5900	0.6717	0.4811	0.3874
U_L	0.5200	0.5755	0.5435	0.4425
U_M	0.9800	0.9801	0.4940	0.4114
U_H	0.5000	0.5560	0.5533	0.4530

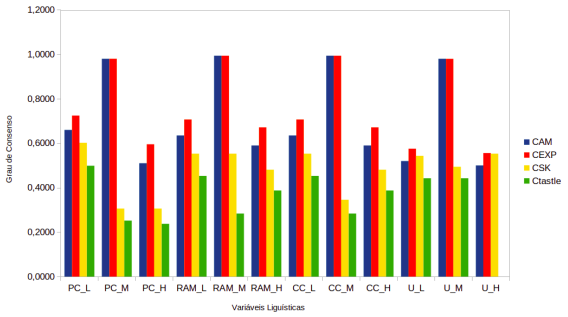


Figura 2. Resultados Gráfico

5. Conclusão

Neste artigo foi desenvolvida uma proposta de avaliação do grau de consenso para apoio à decisão na determinação do nível de utilização de máquinas físicas da CN. A análise de consenso fuzzy foi desenvolvida sobre os dados relativos aos conjuntos fuzzy referentes ao PC, CC, RAM e U. Foi investigado o problema de mensuração do grau de consenso, não apenas comparando valores fuzzy, mas de forma mais abrangente se reportando aos conjuntos fuzzy associados aos atributos do sistema de inferência.

O objetivo central foi validar a decisão de consenso para a tomada de decisão dentre a opinião de vários especialistas, considerando diferentes critérios, e dando enfoque nesta fase do trabalho para análise através das funções de pertinência tipo-1.

Os resultados obtidos mostraram bons níveis de consenso na modelagem do projeto de *Int-FLBCC*, provendo maior confiabilidade para o tratamento das incertezas presente no ambiente da CN.

Como trabalhos futuros pretende-se aplicar os métodos $\mathcal{C}_{AM}$, $C_{\exp_\alpha}$, $\mathcal{C}_{SK}$ e $\mathcal{C}_{Tastle}$ para análise do consenso, levando em consideração a modelagem do projeto por funções de pertinência tipo-2, e ainda, empregando métodos de penalidade [Elkano et al. 2018], visando ampliar a avaliação tanto na precisão quanto da incerteza presente no ambiente.

Referências

Beliakov, G., Calvo, T., and James, S. (2014). Consensus measures constructed from aggregation functions and fuzzy implications. *Knowl.-Based Syst.*, 55:1–8.

Elkano, M., Galar, M., Sanz, J. A., Schiavo, P. F., Pereira Jr, S., Dimuro, G. P., Borges, E. N., and Bustince, H. (2018). Consensus via penalty functions for decision making in ensembles in fuzzy rule-based classification systems. *Appl. Soft Comput.*, 67:728–740.

Gehrke, M., Walker, C., and Walker, E. (1996). Some comments on interval valued fuzzy sets. *Int. Journal of Intelligent Systems*, 11(10):751–759.

Gourisaria, M. K., Samanta, A., Saha, A., Patra, S. S., and Khilar, P. M. (2020). An extensive review on cloud computing. In *Data Engineering and Communication Technology*, pages 53–78.

Karnik, N. N. and Mendel, J. M. (1998). Introduction to type-2 fuzzy logic systems. In *1998 IEEE Int. Conf. on Fuzzy Systems Proc.*., volume 2, pages 915–920 vol.2.

Klement, E., Mesiar, R., and Pap, E. (2004). Triangular norms. position paper I: basic analytical and algebraic properties. *Fuzzy Sets and Systems*, 143(1):5–26.

Mendel, J., Hagras, H., Tan, W.-W., Melek, W. W., and Ying, H. (2014). *Introduction to type-2 fuzzy logic control: theory and applications*. John Wiley & Sons.

Mendel, J. M. (2003). Fuzzy sets for words: a new beginning. In *Fuzzy Systems, 2003. FUZZ '03. The 12th IEEE Int. Conf. on*, volume 1, pages 37–42.

Mendel, J. M., John, R. I., and Liu, F. (2006). Interval type-2 fuzzy logic systems made simple. *IEEE Trans. Fuzzy Systems*, 14(6):808–821.

Moura, B. M., Schneider, G. B., Yamin, A. C., Santos, H., Reiser, R. H., and Bedregal, B. (2021). Interval-valued fuzzy logic approach for overloaded hosts in consolidation of virtual machines in cloud computing. *Fuzzy Sets and Systems*.

Nathani, A., Chaudhary, S., and Somani, G. (2012). Policy based resource allocation in iaas cloud. *Future Generation Computer Systems*, 28(1):94–103.

Oliveira, L., De Moura, R. C., Schneider, G. B., Pilla, M. L., Yamin, A. C., Reiser, R. H. S., and Bedregal, B. R. C. (2021). Toward a fuzzy logic-based consensus analysis in hybrid memory management. In *2021 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6.

Shehabi, A., Smith, S., Sartor, D., Brown, R., Herrlin, M., Koomey, J., Masanet, E., Horner, N., Azevedo, I., and Lintner, W. (2016). United states data center energy usage report.

Tsuchiya, Y. and Hiramoto, N. (2018). Measuring consensus and dissensus: A generalized index of disagreement using conditional probability. *Inf. Sci.*, 439-440:50–60.

Wu, D. and Nie, M. (2011). Comparison and practical implementation of type-reduction algorithms for type-2 fuzzy sets and systems. In *FUZZ-IEEE*, pages 2131–2138. IEEE.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.

Compilação e Execução de código da linguagem QML no Computador Quântico da IBM

João Gabriel da Cunha Schittler, Juliana Kaizer Vizzotto

[1]Curso de Ciência da Computação – Universidade Federal de Santa Maria (UFSM)
Caixa Postal – Santa Maria – RS – Brasil

{jgschittler,juvizzotto}@inf.ufsm.br

Abstract. *This paper descibes an ongoing project that aims to implement a compiler for the quantum programing language QML that generates quantum circuits, and then execute these compiled circuits on a IBM quantum computer using the Qiskit package.*

Resumo. *Esse artigo descreve um projeto em andamento que propõe implementar um compilador para circuitos quânticos para a linguagem de programação quântica QML e então executar os circuitos compilados no computador quântico da IBM (*IBM Quantum Computer*) usando o pacote Qiskit.*

1. Introdução

A computação quântica tem sua origem no trabalho de Richard Feynman [Feynman 1982], onde o autor trata questões sobre simular sistemas físicos quânticos com a utilização de computadores clássicos. Desta maneira, Feynman propôs utilizar fenômenos da mecânica quântica, como superposição e emaranhamento, para resolver problemas mais rapidamente que na computação clássica. Essa questão levantada por Feynman levou à algumas propostas para um modelo quântico de computação, mas a realização de que realmente pode-se ter algum ganho veio somente alguns anos depois com os resultados de Grover[Grover 1996] e Shor [Shor 1997]. Grover propôs um algoritmo quântico para executar busca em registros desordenados, com um ganho quadrático na complexidade temporal comparando com qualquer outro algoritmo clássico. O trabalho de Shor definiu um algoritmo quântico de tempo polinomial para fatorar números inteiros. Para esse problema somente se conhece algoritmos clássicos com tempo exponencial. Pode-se dizer que esses foram os primeiros resultados que geraram um grande interesse de pesquisadores da ciência de computação.

Principalmente com o objetivo de investigar a área da programação quântica e proporcionar o desenvolvimento e estudo de algoritmos quânticos, as linguagens de programação quânticas de alto nível têm sido desenvolvidas nos últimos anos. A linguagem funcional quântica QML [Altenkirch and Grattage 2005] é um exemplo. QML é baseada nas construções de alto nível das linguagens funcionais convencionais e integra computações quânticas reversíveis e irreversíveis, usando lógica linear de primeira ordem para os *weakenings*. Os programas *estritos* não apresentam decoerência quântica e consequentemente preservam superposições e emaranhamento, que são essenciais para o paralelismo quântico. Além disso, os autores apresentam uma maneira natural de compilar os termos funcionais da linguagem em circuitos quânticos. Em [Grattage and Altenkirch 2005] os autores apresentam a implementação de um compilador em Haskell para QML.

O computador quântico da International Business Machines (IBM) e sua plataforma IBM Quantum Experience (IBM-QE) [IBM] proporcionam o uso de um computador quântico em nuvem que pode ser acessado de forma remota por qualquer um.

Com objetivo de investigar os potenciais de linguagens de programação quânticas de alto nível e algoritmos quânticos e também estimular esse ramo de pesquisa na área de Ciência da Computação, o presente trabalho propõe implementar um compilador para QML considerando sua semântica de circuitos quânticos, que possa ser executado no computador quântico da IBM, através da integração com o pacote Qiskit [Qis].

2. Fundamentação Teórica

2.1. Computação Quântica

Computação quântica (CQ)[Nielsen and Chuang 2011], considerando um ponto de vista algorítmico, apresenta um novo modelo de computação, o qual incorpora novas maneiras de projetar e estruturar algoritmos utilizando as características da física quântica. A unidade básica de informação clássica computável é o bit, um sistema físico clássico binário. Em computação quântica, a unidade básica de informação é representada pelo bit quântico ou qubit, um sistema físico quântico binário. O qubit é comumente representado como uma superposição de estados, através da notação *braket*[1] de Dirac: $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$.

Os valores associados às bases α e β representam as amplitudes de probabilidade relacionadas a cada base do qubit ($|0\rangle$ e $|1\rangle$). O qubit pode ser definido como um vetor em um espaço vetorial complexo (espaço de Hilbert), tal que $|\alpha|^2 + |\beta|^2 = 1$.

Formalmente, a combinação de dois ou mais estados quânticos pode ser obtida usando uma operação de produto tensorial ($\otimes$). Se $q = \alpha|0\rangle + \beta|1\rangle$ e $p = \gamma|0\rangle + \delta|1\rangle$ são dois qubits não relacionados. Ao aplicar o produto tensorial temos $q \otimes p = \alpha\gamma|00\rangle + \alpha\delta|01\rangle + \beta\gamma|10\rangle + \beta\delta|11\rangle$.

O processamento de informação quântica é realizado por operadores, que possibilitam a evolução de um sistema quântico. Ele define que um sistema quântico isolado no estado $|\phi\rangle_1$ evolui para $|\phi\rangle_2$ através da aplicação de uma operação unitária $|\phi\rangle_2 = U|\phi\rangle_1$. As operações quânticas tem como característica serem reversíveis, quando se conhece as operações que foram aplicadas ao qubit, é possível retornar ao seu estado inicial.

Outro tipo de operação sobre os bits quânticos é a *medição*, esta uma operação não reversível. Ela é obtida com um colapso na função de onda de um sistema quântico. Dessa forma ao aplicar a medição do valor 0 em um sistema quântico da forma $\alpha|0\rangle + \beta|1\rangle$ obtemos $|0\rangle$ com $|\alpha|^2$ de probabilidade associada.

2.2. Linguagem QML

Uma das maiores dificuldades no desenvolvimento de novos algoritmos quânticos, principalmente considerando pesquisadores da área da Ciência da Computação, é o quão baixo nível esses programas são representados, normalmente descritos como circuitos quânticos. A linguagem de programação funcional quântica

[1]O nome *braket* surge através de uma convenção em que um vetor de coluna é chamado "ket" e sua notação é demonstrada por $|\ \rangle$ e um vetor de linhas é chamado "bra" e tem como notação $\langle\ |$.

QML[Altenkirch and Grattage 2005] traz uma abordagem de alto nível, para auxiliar o desenvolvimento desses algoritmos, contendo estruturas de controle de alto nível como variáveis tipadas, funções e as *keywords* **let**,**if-then-else**.

A semântica dessa linguagem pode ser interpretada como uma sequência de FQCs (*finite quantum computations*), que são representadas com a seguinte quíntupla $\mathbf{FQC} = \{a, h, b, g, \phi\}$, onde a representa o número de qbits da entrada, h representa o tamanho da heap inicial, b representa o número de qbits da saída, g representa o tamanho de lixo necessário e ϕ representa uma operação unária reversível.

2.3. IBM Quantum

A empresa IBM criou um serviço chamado de IBM Quantum Experience [IBM], que, entre outras funções, fornece uma interface gráfica para a criação e execução de circuitos quânticos, e disponibiliza uma interface de comunicação via linguagem de programação Python, chamada de Qiskit [Qis], para executar circuitos programados em código. Além disso, o IBM-QE disponibiliza tutoriais e guias de conceitos sobre CQ, que deixa a execução de algoritmos quânticos bem mais acessível.

Os componentes disponíveis para a criação dos circuitos na interface gráfica e pela biblioteca Qiskit são: portas quânticas, como Hadamard, portas lógicas clássicas e outras operações, como medição de qbit.

O pacote Qiskit permite que o programador declare circuitos quânticos, as portas quânticas dos circuitos, registradores clássicos e quânticos, e a interface de comunicação com os serviços da IBM. No código, a ordem em que o programador declara as operações do circuito é a ordem em que elas vão ser executadas no IBM-QE. Após declarar o circuito que vai ser executado, ele pode ser enviado a um computador quântico da IBM.

3. Implementação de um Compilador para a Linguagem de Programação Quântica QML e Execução no IBM-QE

O presente trabalho propõe implementar um compilador para a linguagem de programação quântica QML, considerando sua semântica de circuitos quânticos, que possam ser executados em computadores quânticos da IBM, através da integração com a biblioteca Qiskit.

A compilação do código da linguagem QML vai passar pelos passos padrão de um compilador: análise léxica, sintática e semântica e geração de código. A linguagem de programação que será utilizada para implementação do compilador é a linguagem C++. Pretende-se utilizar a ferramenta LEX para a análise léxica e a ferramenta YACC para a análise sintática e semântica.

O processo da compilação, de forma geral, vai funcionar da seguinte maneira:

- Similar ao que acontece na linguagem QML, cada procedimento/função vai ser expressado em código por uma estrutura que vai conter as informações da quíntupla FQC. Depois que todas as operações do procedimento forem lidas, um circuito representando essa função será criado.
- Quando uma função é chamada dentro de outra função ou pela main, o circuito dessa função é colocado no lugar dessa chamada, e a indexação dos qubits desse

procedimento se ajusta de acordo com os qubits do contexto que chamou o procedimento, de forma que não existam conflitos.
- Depois que todas as funções forem compiladas, a main do programa vai gerar o circuito final que vai ser escrito no arquivo de saída do compilador.

Está sendo estudado alguma maneira de reutilizar qubits auxiliares de procedimentos, de forma que não causem "de-coerência" quântica com os outros qubits, para que os programas compilados usem a menor quantidade de qubits possível.

Os circuitos quânticos gerados pelo compilador vão estar em um formato que pode ser facilmente lido pelo programa que fará a comunicação com o IBM Quantum para então ser executado. O formato do arquivo deve conter os seguintes componentes:

- Um *Header* inicial, contendo informações que precisam ser lidas antes do circuito, como o número de qubits total do circuito.
- A lista de operações do circuito, juntamente com os qbits envolvidos em cada uma.

Por exemplo, um arquivo gerado pelo compilador contendo um programa quântico, seria como abaixo:

- 2
- Had 1
- NOT 2
- CNOT 0 1
- Measure 0
- Measure 1

Na sequência, a arquitetura proposta conterá um programa implementado na linguagem Python, que fará a leitura desse arquivo gerado, montará o circuito quântico e realizará a comunicação com o IBM-QE, usando o pacote Qiskit.

A proposta descrita neste artigo contempla o trabalho de conclusão do curso de Ciência da Computação da Universidade Federal de Santa Maria (UFSM) do primeiro autor. O trabalho está em fase de revisão bibliográfica e delineamento da arquitetura. A implementação da arquitetura aqui proposta ocorrerá nos meses de outubro de 2021 até fevereiro de 2022.

References

Ibm quantum website. `https://quantum-computing.ibm.com/`. acessado em 24/09/2021.

Qiskit website. `https://qiskit.org/`. acessado em 24/09/2021.

Altenkirch, T. and Grattage, J. (2005). A functional quantum programming language. In *20th Annual IEEE Symposium on Logic in Computer Science*.

Feynman, R. P. (1982). Simulating physics with computers. 21(6):467–488.

Grattage, J. and Altenkirch, T. (2005). A compiler for a functional quantum programming language. Manuscript.

Grover, L. K. (1996). A fast quantum mechanical algorithm for database search. In *Proceedings of STOC-1996*, pages 212–219. ACM.

Nielsen, M. A. and Chuang, I. L. (2011). *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, USA, 10th edition.

Shor, P. W. (1997). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509.

Consolidando a Modelagem do Jogo Aventura Espacial

Yuri da Silva Rosa[1], **Renata Hax Sander Reiser**[1],
Simone André da Costa Cavalheiro[1], **Luciana Foss**[1], **André Rauber Du Bois**[1]

[1]Centro de Desenvolvimento Tecnológico – Universidade Federal de Pelotas (UFPEL)
Rua Gomes Carneiro, 1 – CEP 96010-610 – Pelotas – RS – Brasil

{ydsrosa,reiser,simone.costa,lfoss,dubois}@inf.ufpel.edu.br

Abstract. *Playful and unplugged activities have been proposed as an improved alternative enabling the insertion of methodological innovation to increase learning in elementary education, through the application of computational concepts and techniques. In such context, the consolidation of the Space Adventure activity is described, presenting the modelling of its final stage named as Task 5. In this stage, the game execution promotes learning via simulation of a scientific space journey. The concepts of abstraction, decomposition, pattern recognition are worked on, and the search for solutions for energy efficiency and correction evaluation of data organization is also encouraged.*

Resumo. *Atividades lúdicas e desplugadas têm sido propostas como alternativas viáveis economicamente e com inserção de inovação metodológica para incremento da aprendizagem na Educação Fundamental, via aplicação de conceitos e técnicas da computação. Neste contexto, descreve-se a consolidação da atividade Aventura Espacial, apresentando a modelagem da sua etapa final, identificada como Tarefa 5. Nesta etapa, a execução das jogadas promove aprendizagem via simulação de uma jornada científica espacial. São trabalhados os conceitos de abstração, decomposição, reconhecimento de padrões e estimulam-se ainda a busca de soluções para eficiência energética e avaliação da correção na organização de dados.*

1. Introdução

Atividades lúdicas e desplugadas têm sido propostas como alternativas viáveis economicamente, mas com inserção de inovação metodológica para incremento da aprendizagem na educação fundamental, via aplicação de conceitos e técnicas computacionais [Bell et al. 2015, Jagušt et al. 2018].

O presente trabalho visa consolidar a atividade denominada "Aventura Espacial", que faz parte de uma estratégia para disseminar a importância do Pensamento Computacional [Wing 2006] na Educação Básica, no contexto do projeto Explorando o Pensamento Computacional para a Qualificação do Ensino Fundamental (ExpPC). O projeto ExpPC tem como objetivo geral fazer a ponte entre a academia e a comunidade no tema Pensamento Computacional. Além disso, objetiva sensibilizar a comunidade sobre a importância de se trabalhar no Ensino Básico a metodologia baseada na aprendizagem algorítmica, inerente às atividades da Computação. Para tanto, no site do projeto, são disponibilizados planos de ensino e de aula de atividades que trabalham diferentes habilidades e conceitos relacionados ao PC.

A atividade "Aventura Espacial" foi concebida como um jogo desplugado que propõe uma sequência didática, visando desenvolvimento de habilidades cognitivas para alunos do quarto ano do Ensino Fundamental, onde a aprendizagem estará fundamentada na metodologia do Pensamento Computacional. Os conceitos e habilidades abordadas nesta atividade são estruturas de dados homogêneas e estáticas, abstração, decomposição, reconhecimento de padrões e avaliação. As estruturas de dados são usadas para modelar componentes da nave e planejar as rotas para as viagens interplanetárias.

A estruturas consideradas são:

(i) vetores, modelando tanto os planos de voo, como sequências de instruções que determinam as manobras da nave partindo de um planeta e chegando em outro, quanto o consumo de energia armazenado em baterias da nave.

(ii) matrizes, promovendo correlação entre duas classes de estruturas de dados homogêneas. Como neste caso, uma representação tabular descreve a relação entre os tipos de rochas e os planetas onde foram coletadas, modelando o compartimento da nave destinado ao armazenamento de rochas. Além disso, utiliza-se outra matriz para sintetizar as menores distâncias entre cada par de planetas, auxiliando no planejamento de rotas mais eficientes em termos de gasto de energia.

Estas estruturas de dados vêm sendo estudadas [Kalile 2019] e são relevantes abstrações para representar dados relacionados a diferentes problemas, mesmo em áreas distintas da Computação.

Este artigo reporta a última etapa desta atividade, descrevendo sistematicamente o jogo que visa aprendizagem dos conceitos e habilidades mencionados via simulação de uma jornada científica espacial. Para descrever a Tarefa 5 desta atividade, foram necessários combinar os conceitos e métodos didáticos já descritos nas tarefas[1] anteriores [ExpPC 2013]. O restante deste artigo está organizado como segue. A seção 2 resume a concepção das tarefas da atividade. A seção 3 detalha o jogo Aventura Espacial. Na seção 4 são discutidas as habilidades do PC abordadas. Por fim, a seção 5 apresenta as considerações finais.

2. Concepção e Descrição das Tarefas Iniciais da Atividade

A proposta da atividade didática foi estruturada em cinco tarefas e conta a história da personagem Alex, uma cientista que viaja em uma nave espacial pesquisando amostras de rochas dos nove planetas do sistema solar, partindo do planeta Terra [Rosa et al. 2021]. O mapa do sistema solar é representado por um *grid*, onde a nave movimenta-se de uma célula para outra ao se deslocar pelo mapa.

As tarefas iniciais da atividade estão brevemente descritas na sequência:

Tarefa 1, introduz os conceitos dos vetores como dimensão, posição e valor. Em cada posição do vetor são armazenados símbolos de navegação ($\rightarrow$, $\leftarrow$, $\downarrow$, $\uparrow$) para representação das rotas entre planetas. Por exemplo, a Figura 1 representa uma rota entre dois planetas, onde a nave deve se deslocar duas células para cima, uma para à esquerda, outra para cima e, finalmente, uma à esquerda.

Tarefa 2, introduz operações sobre vetores, como: cálculo de rotas reversas, invertendo

[1]As tarefas anteriores estão disponíveis em `https://wp.ufpel.edu.br/pensamentocomputacional/planos/`.

os valores e as posições dos símbolos de navegação do vetor; criação de rotas compostas, gerando um novo vetor de rota com a composição sequencial dos símbolos de navegação das rotas intermediárias.

Tarefa 3, trabalha a comparação de vetores e introduz o vetor que modela a bateria, o qual contém em suas posições 1's ou 0's para quantificar a energia na bateria da nave. A Figura 2 apresenta um vetor de bateria de tamanho 10 com nível 8 de carga de energia. Para que uma viagem seja viável, é necessário verificar se há bateria suficiente, comparando o nível de carga no vetor e o tamanho da rota. Caso haja energia suficiente, após realizar a viagem, deve-se fazer a atualização dos valores do vetor de bateria, zerando as posições não nulas correspondentes ao tamanho da rota.

Tarefa 4, faz uso de uma matriz para mapear todas as menores distâncias entre os planetas. Esta matriz servirá como consulta aos estudantes durante os momentos do jogo. Além da matriz de distâncias entre os planetas, introduz-se uma matriz que representa a relação entre os planetas e a quantidade de amostras encontradas de cada tipo de rocha.

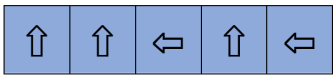


Figura 1. Vetor rota

Figura 2. Vetor bateria

3. Descrição do Jogo Aventura Espacial (Tarefa 5)

Nesta tarefa, tem-se como objetivo usar as estruturas introduzidas nas tarefas anteriores modelagem algumas estruturas e artefatos usados nas viagens espaciais em busca de amostras de rochas. Nas tarefas anteriores, são vencidas as etapas necessárias para planejar e executar as viagens, como descritas nos itens marcados do checklist ilustrado na Figura 3. Este checklist representa toda a estratégia de preparação para a decolagem da nave, a qual simboliza o começo do jogo.

Figura 3. Preparo para Decolagem

Figura 4. Papéis dos Tripulantes

Inicialmente, são definidos os papéis de cada jogador nas equipes. Para estimular a interação e cooperação entre os discentes, são previstos diferentes papéis para cada um dentro da equipe. Um mesmo componente pode ainda assumir mais de um papel na tripulação da nave, conforme ilustrado na Figura 4 e descrito a seguir:

- Primeiro Capitão: decide, com a sua equipe, o destino da viagem e traça as rotas para suas aventuras. Além disso, ele deve calcular a rota de retorno à Terra.
- Segundo Capitão: ajuda o primeiro capitão a traçar as rotas e as executa.

- Engenheiros da nave: responsável por atualizar a localização da nave dentro da rota, atualizar nível de energia da bateria e consertar problemas que possam surgir na nave.
- Explorador: responsável por sortear a carta que determina os resultados da exploração para busca/coleta de rochas ao chegar em um planeta, bem como, possíveis manutenções que devem ocorrer na nave. O explorador deve informar aos respectivos responsáveis (tesoureiro ou engenheiros) as atualizações que devem ser realizadas.
- Tesoureiro: responsável pelo registro/relatório descrevendo as amostras de rochas encontradas/coletadas.

Após a definição do papel de cada aluno, os aplicadores (monitores, professores) devem orientar a largada da partida. Inicialmente, o primeiro capitão, apresentará três destinos para sua equipe. O planeta mais votado entre a equipe será eleito como o primeiro destino. E ambos, o primeiro e o segundo capitães, traçam a rota com partida no planeta Terra. A rota deve ser construída levando em consideração o tamanho da bateria (que define o maior trecho a ser realizado) e a melhor eficiência em termos de consumo de energia. Para isso, os capitães devem fazer uso da matriz das menores distâncias entre os planetas. A rota planejada deve ser registrada tanto no mapa do sistema solar quanto no vetor de rota.

Antes de iniciar a viagem, toda a equipe deve verificar se a rota está correta (leva ao planeta desejado) e é viável (o nível de energia da bateria é suficiente para chegar ao planeta de destino e retornar). Caso haja algum problema, a equipe deve definir os ajustes necessários para a decolagem. Quando estiver tudo certo com o planejamento da viagem, a rota será passada ao Segundo Capitão, para que ele simule os movimentos da nave, seguindo os símbolos de navegação armazenados no vetor de rota.

Ao chegar primeiro planeta da rota, o vetor de rota será entregue aos engenheiros que deverão atualizar tanto o nível de energia na bateria quanto o indicador do planeta em que a nave se encontra nesta fase do jogo. Além disso, o explorador deverá escolher uma carta dentre as dispostas daquele planeta e apresentar suas informações ao grupo, encaminhando aos respectivos responsáveis para as devidas providências. Neste momento, se a nave apresentar algum problema, o mesmo deverá ser solucionado pelos engenheiros. Assim como, se houver amostras encontradas, o tesoureiro deverá atualizar na matriz de amostras.

Figura 5. Carta de Coleta

Figura 6. Carta de Coleta e Bônus

As cartas referentes a cada planeta podem indicar os tipos e as quantidades de amostras coletadas na exploração atual, bem como, definir bônus (como, por exemplo, aumento no nível de energia da bateria) ou problemas técnicos (como, por exemplo, perda de informações armazenadas nos vetores ou necessidade de alteração de rota). As Figuras 5 e 6, ilustram alguns exemplos de cartas que estão associadas a um planeta.

Depois de todas as atualizações realizadas, os capitães devem traças a rota de retorno ao planeta Terra, calculando a rota reversa da viagem de ida. Ao retornar à Terra, os jogadores devem reportar-se ao primeiro momento da tarefa, onde o capitão escolhe junto de sua equipe o próximo destino para prosseguir com a aventura. O jogo se encerra após retornar ao planeta Terra e a equipe já ter obtido pelo menos uma amostra de cada tipo de pedra dos planetas agendados na rota inicial.

4. Discutindo Habilidades do Jogo Aventura Espacial

Nas etapas anteriores, aplicaram-se abstrações mapeadas pelos conceitos de vetores e matrizes, incluindo conceito de decomposição na estruturação das rotas.

Nesta última etapa, onde ocorrem as jogadas, trabalha-se também outros conceitos, como avaliação de eficiência, quantificando a energia disponível na nave e previsão do consumo no percurso da rota.

Além disso, tem-se a avaliação de correção do plano de navegação. Ao construir os vetores de navegação, os alunos devem simular um teste verificando se a sequência de instruções leva de fato do planeta de origem ao de destino. A identificação de padrões é aplicada na busca pela menor rota entre dois planetas, analisando os vetores de navegação. As menores rotas não devem usar símbolos inversos, ou seja, evitando ciclos que aumentam o percurso. Assim, se uma rota usa uma seta para cima, não podem aparecer setas para baixo, e vice-versa (de forma análoga, se usar uma seta para à esquerda, não podem aparecer setas para à direita, e vice-versa).

Desta forma, diferentes conceitos do PC são abordados no decorrer da atividade com referência a Tarefa 5, como uma forma de viabilizar ou melhorar as viagens espaciais, tornando-as seguras e eficientes para o jogo Aventura Espacial.

5. Conclusão

Este artigo descreveu a Tarefa 5 da atividade lúdica e desplugada Aventura Espacial, que visa incentivar a aprendizagem e o desenvolvimento do Pensamento Computacional junto às séries iniciais do Ensino Fundamental. A proposta buscou trabalhar conceitos de abstração, decomposição e identificação de padrões. Outros aspectos relevantes também foram incentivados na proposta de jogo lúdico apresentada, como a busca por eficiência energética e a avaliação da correção na organização de dados.

Como trabalho futuro, deseja-se criar cursos remotos síncronos para serem divulgados junto à Secretaria Municipal de Educação e Desporto para a formação dos professores da rede municipal de Ensino Básico, sensibilizando os profissionais sobre a importância de se trabalhar as habilidades do PC no Ensino Fundamental. Busca-se ainda investir em tutoriais de formação, os quais serão disponibilizados para a comunidade escolar.

Referências

Bell, T., Witten, I. H., and Fellows, M. (2015). *Computer Science Unplugged: an enrichment and extension programme for primary-aged students*.

ExpPC (2013). Explorando o pensamento computacional desde o ensino fundamental. `https://wp.ufpel.edu.br/pensamentocomputacional/`. UFPel.

Jagušt, T., Krzic, A. S., Gledec, G., Grgić, M., and Bojic, I. (2018). Exploring different unplugged game-like activities for teaching computational thinking. In *2018 IEEE Frontiers in Education Conference (FIE)*, pages 1–5. IEEE.

Kalile, P. A. (2019). ProgramaÇÃo como uma ferramenta no ensino de matrizes. In *XXIII Encontro Brasileiro de Estudantes de Pós-Graduação em Educação Matemática*. UNICSUL.

Rosa, Y. S., Reiser, R. H. S., Cavalheiro, S. A. C., Foss, L., and Bois, A. R. D. (2021). Aventura espacial: Proposta de atividade para o desenvolvimento do pensamento computacional. In *Anais do XXVII Workshop de Informática na Escola*, pages 1–10, Porto Alegre, RS, Brasil. SBC.

Wing, J. (2006). Computational thinking. *Commun. of the ACM*, 49(3):33–35.

Description of Command and Control Networks in Coq*

Guilherme G. F. da Silva[1], Edward Hermann Haeusler[1], Cláudia Nalon[2]

[1]Pontifical Catholic University of Rio de Janeiro, Rio de Janeiro, RJ, Brazil

[2]University of Brasília, Brasília, DF, Brazil

guilhermegfsilva@gmail.com, hermann@inf.puc-rio.br, nalon@unb.br

Abstract. *A command and control (C2) system can be defined as a group of individuals organised hierarchically in which higher-ranking individuals can issue directions to their subordinates with a certain goal in mind. We present a model for representation of command and control networks in the Coq proof assistant based on a tree data structure. Our model utilises Coq's implementation of data structures and includes examples of how to define functions and properties that may be relevant in a C2 system, as well as examples of some of the tests we have performed to verify the correctness and usability of our model. The examples given here are simple, but constitute basic blocks that can later be used in more realistic formalisations.*

1. Introduction

The NATO terminology database [Nato Terminology Office 2021] defines *command* as: 'The authority vested in an individual of the armed forces for the direction, coordination, and control of military forces', and *control* as: 'The authority exercised by a commander over part of the activities of subordinate organisations, or other organisations not normally under his command, that encompasses the responsibility for implementing orders or directives.' A Command and Control (C2) network harmoniously integrates both concepts with the aim of defining ways to successfully accomplish a mission. The network is often organised as an adaptive hierarchical team of agents (or coalitions) whose members collaborate over ever changing situations, applying common tactics and protocols in order to achieve some desired outcome. Not surprisingly, those networks have been widely used in military organisations as a way to minimise occurrences of unforeseen or emerging properties within a complex system [Development, Concepts and Doctrine Centre 2017]. The validation of C2 networks is, therefore, crucial not only for ensuring the full realisation of strategic goals, but also to give warranties that the accomplishment of a mission is not costly; particularly, it aims at reducing risks and minimising the loss of lives.

There are some reports in the literature on C2 formal validation. In general they fall into two approaches. One is to formalise the network as a model and to verify dynamic properties using a temporal logic model-checker; the other is to formally prove relevant properties, using an Interactive Theorem-Prover (ITP), or even an Automatic Theorem-Prover (ATP). For instance, [Wang 2012, Chapter 8] describes the use of stochastic Petri-Nets to formalise the validation of C2 networks applying the first

*This work was partially funded by the project FINEP 2904/20 - Sistema de Sistemas de Comando e Controle (S2C2).

approach. [van de Pol et al. 1998] proves some very basic properties using the interactive prover PVS [Owre 2020] together with a model-checker embedded in that system to validate models generated from the specification. The latter seems to be one of the first works integrating both approaches, but it is restricted to the validations of instances/configurations of particular C2 networks. Due to the widespread popularity of model-checking and its apparently easier end-user modelling, there are more reported approaches using model-checkers than logical formalisations, or even about the formal mathematics of C2. We however believe that the theorem-proving approach is better than model-checking to ensure the absence of non-normative features, as required in the validation of C2 networks.

This work is part of a project that aims to formally develop and validate a methodology to ensure that C2 networks are correct with regard to non-emerging properties as much as possible. As far as we know, this is the first project that is based on theorem-proving with the focus on normative specification of C2. We started the formalisation using Coq [The Coq Team 2021] and the very basic blocks of C2 networks. This article reports the achievements obtained for these building blocks of C2.

2. Representation of a Network in Coq

Firstly, let us establish what networks mean in this context. We have defined a network as a group of nodes, with each node representing an individual in a C2 system. A valid network has at least one node. The hierarchy between these individuals is represented by a connected and acyclic graph, i.e. a tree. The root of the tree represents the leader in our C2 system, with each edge indicating the relation between individuals and their direct subordinates.

We have established that each individual in the network (other than the leader) has only one direct superior. However, an individual can have any number of direct subordinates. Additionally, every network has one and only one leader, and a network must possess a minimum of two nodes and one edge. Figure 1 shows an example of a graph representing such a system, where individual 1 is the leader, 2 and 3 are her direct subordinates, and so on.

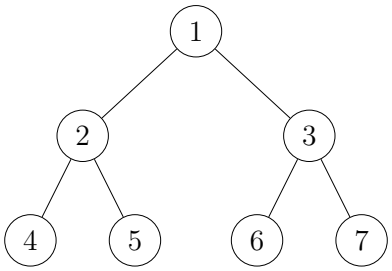


Figure 1. Directed graph representing the hierarchy in a command and control network

2.1. Defining a Network

Now, let us describe how to represent these C2 graphs in Coq. To do so, we need to define a data structure. Coq provides us with the means to do so via the Structure operator. The

Coq code containing the main body of the structure is defined as net in our source code, which can be found at [Silva, Guilherme 2021]. This net object represents a command and control network as a structure type containing a single leader node and a set of nodes subordinated to this leader. These subordinates, in turn, can have their own subordinates, and so on. The objects and functions that make up this structure are as follows.

We define the number of nodes in our model as a variable named `nodes`, which is a natural number. By definition, nodes in our model are numbered individually starting from 1 without skipping any number, so the value of nodes will also always be equal to the highest node value in a particular network. A structure with a value of 10 assigned to the nodes field, for example, will have a total of 10 nodes numbered from 1 to 10. The largest network we have used when testing our model has a total of 31 nodes.

leader is a natural number value which tells us the index of the node which is the network's leader, equivalent to the root of the graph.

superior tells us which nodes are direct subordinates of which. This field is a list of pairs of natural numbers representing our graph, with each pair representing a single edge of the graph via the indices of two nodes (parent and child). We assume that the numbers contained within these pairs are consistent with the node values defined by nodes. For example, for the network shown in Figure 1, our list of edges would be represented in Coq as the list `(1,2)::(1,3)::(2,4)::(2,5)::(3,6)::(3,7)::nil`.

These two are the fields that must be given as parameters when creating an instance of the structure, as we will see later. The remaining fields are the functions our structure will use.

second-in-command is a function which tells us which node in the network is the second-in-command of the current leader and the one that should replace the current leader if necessary. It is defined as the first subordinate of the leader node, as we will describe in more detail ahead.

parent and **children** are functions that receive a single node (natural number) as an argument and, respectively, return the index of the superior/parent node or a list of indices indicating the children/subordinates of the node.

is_parent and **is_parent_bool** are two similar functions that tell us if two given nodes are parent and child to each other in the graph.

node_level tells us the level of a node in the hierarchy. By definition, the leader should have a level value of 1, its direct subordinates should have a level of 2, and so on.

We detail the implementation of these functions in the next section.

2.2. Network Functions

Second_in_command. This function, as stated, tells us which node is considered the highest-ranking subordinate of the current leader and the one that should be made the leader if the current one needs to be replaced. We define the second-in-command node as the first node to appear in the list of edges as a direct subordinate of the leader node. The function that returns this node is defined in our code as **get_second**. This function receives two parameters, edges and leader. This is consistent with how we defined **second_in_command** in the structure, i.e. that

it is always **get_second** applied to two arguments, the list of edges and the leader value.

What this function does is recursively search through the list of edges (a, b), comparing the first number in each one (the parent node, or a) with the value of leader until it finds a match. When that happens, the second value of the pair (the child node, or b) is returned. If the value does not match, we call **get_second** again recursively on the remaining edges, defined here as edges'. If we reach the end of the list without finding any matches, we return a default value of 0 indicating that no valid second-in-command node was found.

Parent. This function receives one node and needs to tell us its parent node. Once again, we find the value by searching through the list of edges recursively, this time comparing the node value with the child node in each edge. The **get_parent** function in our code does this. Since our model already assumes that the network is defined with each node having only one parent, there is no need to search through the rest of the list after a match is found. Should the function finish searching the list without finding an edge whose target node matches the given value, it returns 0 by default, indicating that the node has no parent. This should happen only when the value given is the leader, i.e. the root node.

Children. This function operates similarly to parent. However, since a node can have any number of children, this function needs to return a list of natural numbers. The **get_children** function returns this list. Once again, the function works by recursively calling itself to search through the list of edges, this time comparing the given value with the parent node, a, in each edge (a, b). If a match is found, we append the child value b to the list that will be our final product and continue searching via recursion. If there is no match, we do a recursion without appending anything to the list. At the end of the run, we will have searched through every edge and have the complete list of children of the given node. If the node has no children, an empty list value nil will be returned.

Is_parent and is_parent_bool. These functions take two nodes a and b as input and return whether a is a parent of b. Coq has two different types to represent this, proposition (Prop) or boolean (bool). Prop defines the constants for truth and falsity in the metalanguage whereas bool is a defined type similar to the one used in programming languages. For comparison's sake, we have included those two different functions, one for each of these types. The two are mostly similar but with some differences in which operators are used. Both functions work by searching through the edge list for an element in which both of the values, a and b, match the given parent and child. A value of "false" is only returned if the function searches through the entire list without finding any matches.

Node_level. This function is a bit more complex. It needs to tell us the level of a node in the hierarchy. The leader's level is by definition 1, while the level of its direct subordinates is 2, and so on.

To tell the level of a node, we need to count how many levels separate it from the leader. We do this by first searching the edge list for the pair (a, b) where b is the value of our target node. Once we have found it, we increment a counter by 1 and call a recursion to find the level of a, its parent node. The recursion stops at the node representing the leader. Our counter will then let us know how many levels separate the target node from the leader. In summary, we search backwards

starting from our target node and count the number of levels toward the root.
The issue here is that since we do not know how the edges are ordered, we need to run through the list multiple times. In other words, this is a function with $O(n^2)$ complexity in which we need to search the list at least n times to guarantee we will have the value we want. Doing this requires more than one level of recursion, as shown in the get_level function in our code [Silva, Guilherme 2021].

2.3. Defining a Network Instance

Now that we have talked at length about our structure and its functions, we can create an actual instance of a network to see some of them in use. In Figure 2, we revisit the network example shown earlier in Figure 1, and we can see that it has three different levels of hierarchy.

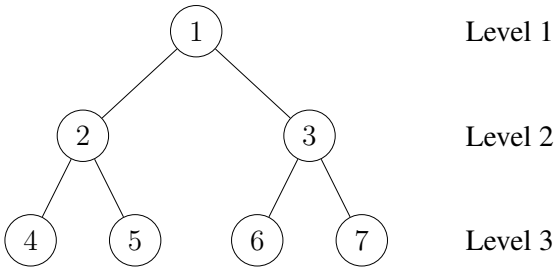


Figure 2. Example of a network with seven individuals and three hierarchy levels.

By creating this network in Coq as the object `net_1`, we can try computing the functions we had previously defined on it and confirm that the results we get are the expected ones. Table 1 shows some examples of this, with the left column showing Coq's Compute command being applied to `net_1` with the corresponding functions and nodes as parameters, and the right column showing the corresponding output displayed by Coq.

Coq Input	Coq Output
Compute is_parent_bool `net_1` 1 2.	= true : bool
Compute is_parent_bool `net_1` 2 3.	= false : bool
Compute node_level `net_1` 1.	= 1 : nat
Compute node_level `net_1` 2.	= 2 : nat
Compute node_level `net_1` 3.	= 2 : nat
Compute node_level `net_1` 4.	= 3 : nat
Compute parent `net_1` 2.	= 1 : nat
Compute children `net_1` 1.	= [2; 3] : list nat

Table 1. Examples of Coq operations on a network instance.

2.4. Defining Properties

In this next part, we will talk about how to use the appropriate tools in Coq to define properties that a network and its elements must have. We exploit properties like these as a basis when building and proving theorems related to command and control systems.

To start with a simple example, let us define the property that "in any network, the leader must be one of its elements". As mentioned before, the list of nodes in a network is represented by a single natural number telling us how many nodes there are, with the assumption that they are all individually numbered from 1 to the stated value. Therefore, a value of 10 in this field, for example, tells us that we have a network with 10 nodes numbered 1 to 10. The leader is also represented by a natural number. Thus, in order to define that the leader is always a valid node, all we need to do is inform Coq that its index is contained in that interval. This can be done in Coq via the simple definition line

```
Definition leader_is_in_net := forall n : net, leader n <= nodes n.
```

Another property we can define is the affirmation that no node can be its own superior. To do this, we will use the superior element, which, as already shown, is a list of pairs of numbers representing each edge of the graph, i.e. the indices of a parent node and child node. Basically, what we want to say is that none of these pairs contain the same number twice. This can be done via the definition line:

```
Definition no_self_superior := forall (n : net) (i : nat),
  fst (nth i (superior n) (0,0)) <> snd (nth i (superior n) (0,0)).
```

These are a few examples of basic properties that any network in our model must have. Properties like these can be used as a stepping stone for defining more complex properties as well as for constructing theorems and proofs.

3. Dynamic Networks

One thing we have not yet explored in our Coq model is the fact that a C2 network may need to undergo changes in its organisation over time. For example, an individual who has been removed from the network may need to be replaced or have its subordinates transferred to another.

We can implement these changes into our model by writing functions that can be applied to a network to alter the configuration of its elements. Consider the network shown previously in Figures 1 and 2, for example. This network has node 1 as its leader and 2 and 3 as the leader's direct subordinates. According to the function described in Section 2.2, node 2 is the one considered this network's second-in-command. So let us consider what might happen if node 1 were removed and needed to be replaced with its second-in-command, i.e. 2. Naturally, we need to account for node 1's other direct subordinates (in this case, node 3). One way to handle this is to have the subordination of the other nodes transferred directly to 2, the node that succeeds 1. Figure 3 shows the resulting network that we expect from this transformation.

To define a way for our model to make these changes on its own, we can write a function that creates a new network with leadership handed down to the second-in-command. We have dubbed this function **next_leader**. The **next_leader** function creates

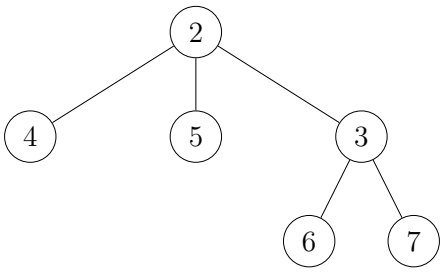


Figure 3. Resulting network after removal of node 1 and transferal of leadership to node 2 in the network initially shown in figure 1.

a new network object with one less node, the second-in-command of the original network as the leader, and a new group of edges defined by another function named change_node.

For example, suppose we want to define a new network `net_2` by applying **next_leader** to `net_1`. We can then apply Coq's evaluation functions to `net_2` and verify that it has the properties expected of the network shown in Figure 3. As you can see in Table 2, Coq identifies node 2 as the leader, 3 as the second-in-command, 6 as the total number of nodes, and the five edges of the network in Figure 3 .

Coq Input	Coq Output
Compute leader `net_2`.	= 2 : nat
Compute nodes `net_2`.	= 6 : nat
Compute second_in_command `net_2`.	= 3 : nat
Compute superior `net_2`.	= [(2, 3); (2, 4); (2, 5); (3, 6); (3, 7)] : list (nat * nat)

Table 2. Coq operations applied to object `net_2`, defined as (next leader `net_1`.)

We can proceed to define more networks by applying this or any other rearrangement function to `net_1` or `net_2`. More functions like this one can easily be defined by following the same principles used for this one. For example, we could expand this succession function into one that replaces any given node in a network (rather than just the leader) with its highest-ranking subordinate; we could define a function that transfers all the subordinates of node a to node b, or one that simply adds a new node as a subordinate to one already in the network.

One thing that should be noted when we remove nodes from a network like this is that we need to assert that the resulting network still has a minimum of two nodes and one edge connecting them. A single node is not a valid network as it has no edges and no way to designate a node as second-in-command.

4. Issuing Commands

We have talked about network hierarchy and reorganisation in our model, but have yet to cover arguably the most important part of a command and control system, which is the commands themselves. A command, as we define here, is an instruction given by a node

to its subordinate(s) telling them what to do. One way we can handle commands is to assign a numeral value to each node that indicates what it is currently doing.

Returning to our structure definition, take a look at the state field, which is a list of pairs of natural numbers. Each pair in this list contains the number of a node and another number representing its current state. For example, suppose that we want our network to have the initial state of all its nodes as "idle". We can choose the value 1 to represent this state and define the network as follows.

```
Definition net_1 : net := Build_net 7 1
 ((1 , 2)::(1 , 3)::(2 , 4)::(2 , 5)::(3 , 6)::(3 , 7)::nil)
 ((1 , 1)::(2 , 1)::(3 , 1)::(4 , 1)::  (5 , 1)::(6 , 1)
  ::(7 , 1)::nil).
```

Now, we can establish commands that change the current state of one or more nodes. Let us define the value 2 as representing the state "move". Suppose that we want node 3 to order all of its direct subordinates to move. All we need to do is establish a way to change the state of every node that is a subordinate of node 3 to "2".

5. Conclusion

We hope that the examples of structures, functions and properties described here can be of assistance to Coq developers in search of a general model for a command and control system or any system in which the concept of hierarchy may be relevant, as well as developers simply seeking some insight into how Coq operates.

We also intend to continue the development of this model where possible by expanding it to include more complex functions and properties, particularly ones based on the ones already established here.

References

Development, Concepts and Doctrine Centre (2017). Future of command and control. *Joint Concept Notes*, 17(2).

Nato Terminology Office (2021). Natoterm. `https://nso.nato.int/natoterm/content/nato/pages/home.html?lg=en`. Last visited: 17-Sep-2021.

Owre, S. (2020). PVS 7.1, The Prototype Verification System. `https://pvs.csl.sri.com/`.

van de Pol, J., Hooman, J., and Jong, E. (1998). Formal requirements specification for command and control systems. In *In Proceedings of the Conference on Engineering of Computer Based Systems*, pages 37–44. IEEE.

Silva, Guilherme (2021). Source code. `http://github.com/GGFSilva/CommandAndControl/`.

The Coq Team (2021). Coq 8.13. `https://coq.inria.fr/`.

Wang, J. (2012). *Timed Petri Nets: Theory and Application*. The International Series on Discrete Event Dynamic Systems. Springer US.

Desenvolvimento de uma Ferramenta para Teste Diferencial de Compiladores Usando Códigos Gerados Aleatoriamente*

Bruno Schafaschek Coelho[1], **Luiz Felipe Krauz**[1], **Samuel da Silva Feitosa**[1]

[1] Instituto Federal de Santa Catarina (IFSC) – Caçador – SC – Brasil

{bruno.sc11, luiz.k1999}@aluno.ifsc.edu.br, samuel.feitosa@ifsc.edu.br

Abstract. *It is notable that software projects are becoming more complex and extensive, which depend mostly on the compiler or interpreter of the programming language chosen for their development. In this sense, how is it possible to guarantee the quality and reliability of these development tools? One of the possibilities is to run tests exhaustively, identifying and correcting errors, until we have some certainty that the code produced is free of bugs. In this way, this project proposes the development of a tool that applies generated randomly test cases and compares the results with different Java compilers.*

Resumo. *É notável que os projetos de software estão se tornando mais complexos e extensos, os quais dependem em grande parte do compilador ou interpretador da linguagem de programação escolhida para o seu desenvolvimento. Neste sentido, como é possível garantir a qualidade e confiabilidade dessas ferramentas de desenvolvimento? Uma das possibilidades é executar testes exaustivamente, identificando e corrigindo erros, até que se tenha certa segurança de que o código produzido esteja livre de* bugs*. Sendo assim, este projeto propõe o desenvolvimento de uma ferramenta que aplique casos de teste gerados aleatoriamente e compare os resultados com os diferentes compiladores de Java.*

1. Introdução

Atualmente Java está entre as linguagens de programação mais utilizadas no mundo [Cass 2019], sendo uma linguagem orientada a objetos e fortemente tipada, que vem sendo aplicada em projetos de diversos portes. Por ser utilizada em projetos grandes e em sistemas críticos, torna-se importante garantir que não ocorram erros de execução, uma vez que qualquer problema poderia causar danos irreversíveis.

Sendo assim, para minimizar possíveis erros em projetos de software é de grande importância que diversos testes sejam executados. Isso não é diferente na área de desenvolvimento de compiladores. Soma-se a isso a existência de diferentes implementações (compiladores e máquinas virtuais) para a linguagem Java. Entretanto, somente através da utilização de testes conduzidos por pessoas não é possível garantir eficácia e qualidade, já que os casos de testes podem ficar incompletos devido a própria limitação humana [Palka et al. 2011]. Além disso, é praticamente impossível que uma equipe de testes consiga testar efetivamente dezenas de milhares de linhas de códigos.

Neste contexto, este projeto busca responder ao seguinte questionamento: *É possível detectar bugs em compiladores Java com maior cobertura de código, através*

*Este trabalho encontra-se em fase de desenvolvimento.

de teste diferencial, usando programas gerados aleatoriamente? Estão previstas duas etapas para obter esta resposta. Primeiro, é proposto a adequação do gerador de códigos aleatórios [Kraus et al. 2021] desenvolvido previamente pelos autores deste artigo, o qual é capaz de construir programas automaticamente a partir de classes e interfaces pré-existentes, para atuar no contexto de teste diferencial de compilares [McKeeman 1998]. Segundo, será desenvolvida uma ferramenta capaz de testar propriedades em diferentes implementações da linguagem Java, procurando possíveis diferenças entre elas, as quais indicariam uma falha no processo de desenvolvimento. Para este trabalho, serão consideradas as propriedades de compilação e comportamento.

O restante do texto está estruturado da seguinte forma: Na Seção 2 encontram-se os trabalhos relacionados, a Seção 3 apresenta a técnica de teste diferencial de compiladores, a Seção 4 é onde se define a geração de código aleatório, e, por fim, as Seções 5 e 6 apresentam os resultados parciais e os resultados esperados, respectivamente.

2. Trabalhos Relacionados

A biblioteca QuickCheck [Claessen and Hughes 2000] é uma das precursoras na área de testes baseados em propriedades, a qual utiliza dados gerados aleatoriamente como entrada. Utilizando esta biblioteca, Palka et al. [Palka et al. 2011] desenvolveu um gerador de código Haskell para efetuar o teste diferencial do compilador desta linguagem. Este trabalho serve de inspiração para a proposta deste artigo, o qual considera a linguagem Java, ao invés do Cálculo Lambda, envolvendo uma complexidade maior em termos de construções sintáticas e semânticas.

Outros trabalhos na área de teste diferencial de compiladores também foram desenvolvidos para outras linguagens de programação, buscando utilizar maneiras de otimizar e melhorar a qualidade dos testes. Como exemplo, pode-se citar os trabalhos de McKeeman [McKeeman 1998] que utilizou esta abordagem para testes na linguagem C, e Ofenbeck et al. [Ofenbeck et al. 2016] que fez uso desta abordagem para testar o compilador de C++, dentre outros.

3. Teste Diferencial de Compiladores

O teste diferencial de compiladores, tem como foco principal a utilização de casos de teste (que podem ser gerados aleatoriamente) para comparar dois ou mais sistemas [McKeeman 1998]. Os compiladores devem passar por uma bateria de testes, onde ambos devem receber os mesmos parâmetros de entrada, e caso ocorra uma incongruência nos testes, como um *crash* ou um *loop* por exemplo, isto indicará que existe um *bug* em um dos compiladores, tendo como probabilidade maior um erro no compilador não oficial [Ofenbeck et al. 2016].

O maior desafio no teste diferencial é garantir a qualidade dos testes, já que é necessário que parâmetros eficientes sejam utilizados. Neste sentido, a utilização de testes gerados de forma aleatória podem auxiliar no processo, uma vez que tendem a ter uma cobertura maior do código base do compilador. Ainda assim, é possível que sejam relatados falsos positivos, uma vez que, dependendo da situação, mesmo com resultados diferentes entre as execuções, ambas podem estar corretas.

4. Geração de Código Aleatório

O mecanismo de geração de código Java aleatório já desenvolvido [Kraus et al. 2021] utiliza uma série de bibliotecas, que permitem deste a extração de informações de classes pré-existentes usando *Java Reflection*[1], a composição de ASTs e compilação para código Java através do *JavaParser*[2], e a especificação de testes baseados em propriedades com o framework *Jqwik*[3].

A geração do código é orientada por um tipo válido, ou seja, primeiro são extraídas informações sobre classes e interfaces, e definido de forma aleatória um tipo primitivo ou definido pelo usuário, para que seja produzida uma expressão ou diretiva seguindo esta definição. Recursivamente são gerados acessos a atributos públicos, invocação de métodos, construção de objetos, diretivas condicionais ou de repetição, etc. Para apresentar parte do mecanismo de geração, abaixo é apresentado um trecho de código que demonstra a geração de diretivas em termos gerais (condicionais, repetição e expressões), e um método que apresenta a geração de diretivas condicionais.

```
1  @Provide
2  public Arbitrary<Statement> genStatement() {
3     //Gerar possiveis Statements
4     return Arbitraries.oneOf(genIfStmt(), genWhileStmt(), genExpressionStmt));
5  }
6  @Provide
7  public Arbitrary<IfStmt> genIfStmt() {
8      Arbitrary<Expression> e = mCore.genExpression(PrimitiveType.booleanType());
9      return e.map(exp -> new IfStmt(exp, genStatement().sample(), genStatement().sample()
           ));
10 }
```

Como pôde ser percebido no código acima, o método *genStatement* realiza a seleção de forma aleatória de uma possível diretiva válida. No método *genIfStmt* é gerada uma diretiva condicional, onde primeiramente é gerada uma expressão com o tipo *booleano* através da chamada do método *genExpression*, para então combinar com duas novas diretivas (*genStatement*) para os casos verdadeiros e falsos da expressão condicional. Esse mecanismo de geração será adaptado para produzir entradas válidas para a ferramenta de teste diferencial de compiladores Java, a qual está sendo desenvolvida neste projeto.

5. Resultados Parciais

Este é um projeto que encontra-se em desenvolvimento, sendo que parte importante deste já foi desenvolvida [Kraus et al. 2021]. A geração de programas para servir de entrada para os casos de teste está concluída, podendo sofrer pequenas adaptações para a aplicação no contexto descrito neste texto. A Figura 1 apresenta o fluxo de execução da ferramenta para teste diferencial de compiladores que está sendo desenvolvida.

O fluxo de execução inicia com o recebimento de uma série de classes / interfaces Java (*imports*), dos quais serão extraídas as informações sobre os tipos definidos pelo usuário, como os nomes de classes ou interfaces, seus atributos, métodos e construtores, etc. Estas informações servem como base para o mecanismo de geração de código aleatório, o qual é responsável por produzir milhares de programas diferentes, os quais

[1] https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/package-summary.html

[2] www.javaparser.org

[3] www.jqwik.com

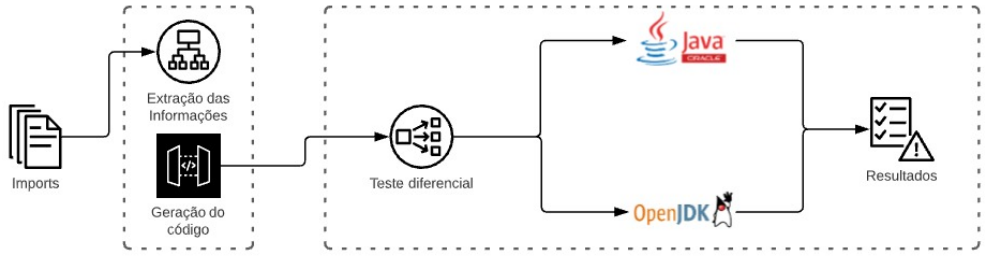


Figura 1. Fluxo de execução proposto.

servirão como entrada (casos de teste) para o instrumento de teste diferencial de compiladores.

Na etapa final, que ainda deve ser desenvolvida, os casos de teste serão compilados por diferentes compiladores e executados individualmente nas suas respectivas máquinas virtuais. Estatísticas de compilação e execução (comportamento) de ambos os compiladores/JVMs serão coletadas e comparadas para verificar a existência de inconsistências, o que será um indicativo de possíveis erros nas ferramentas em estudo. Por exemplo, diversos programas gerados aleatoriamente serão compilados em diferentes implementações da linguagem Java, verificando a propriedade de compilação. De forma similar, os mesmos programas serão executados nas diferentes implementações, e os resultados (estado e output) serão comparados, verificando assim a propriedade de comportamento. Em ambas as verificações, caso existam divergências entre as ferramentas testadas, um indicativo de erro será reportado. A partir da análise dos erros detectados pela ferramenta, serão abertos relatórios de *bug* para os respectivos compiladores. Como referência, o compilador de Java da Oracle será utilizado como implementação oficial para os testes de outros compiladores.

6. Resultados Esperados e Contribuições

Este projeto consiste no estudo de teste diferencial de compiladores, mais especificamente aqueles referentes à linguagem Java, através da execução de casos de teste gerados de forma aleatória. A hipótese levantada para este trabalho é que através da utilização destes casos de teste é possível ter uma maior cobertura dos casos implementados nos compiladores, permitindo a detecção de *bugs* antes da liberação destes sistemas para os usuários finais (programadores), evitando a introdução de erros em cascata em projetos de software.

Esta proposta pretende contribuir para o avanço desta área de pesquisa, fornecendo uma ferramenta de software livre para a execução de teste diferencial de compiladores Java através de código gerado aleatoriamente. Além disso, o mecanismo de geração de código e a metodologia proposta neste projeto também podem ser aproveitadas para outros escopos ou outras linguagens de programação.

Referências

Cass, S. (2019). As principais linguagens de programação de 2019. IEEE Spec.

Claessen, K. and Hughes, J. (2000). Quickcheck: A lightweight tool for random testing of haskell programs. *SIGPLAN Not.*, 35(9):268–279.

Kraus, L. F., Schafaschek, B., Ribeiro, R. G., and da Silva Feitosa, S. (2021). Synthesis of random real-world java programs from preexisting libraries. In *25th Brazilian Symposium on Programming Languages*, SBLP'21, page 108–115, New York, NY, USA. Association for Computing Machinery.

McKeeman, W. M. (1998). Differential testing for software. *DIGITAL TECHNICAL JOURNAL*, 10(1):100–107.

Ofenbeck, G., Rompf, T., and Püschel, M. (2016). Randir: Differential testing for embedded compilers. In *Proceedings of the 2016 7th ACM SIGPLAN Symposium on Scala*, SCALA 2016, page 21–30, New York, NY, USA. Association for Computing Machinery.

Palka, M. H., Claessen, K., Russo, A., and Hughes, J. (2011). Testing an optimising compiler by generating random lambda terms. In *Proceedings of the 6th International Workshop on Automation of Software Test*, AST '11, pages 91–97, New York, NY, USA. ACM.

Estudos sobre o algoritmo de Grover e sua implementação*

Liliana Souza do Carmo[1], **Gisele Bosso de Freitas**[1]

[1] Centro de Ciências Exatas, Naturais e Tecnológicas (CCENT)
Universidade Estadual da Região Tocantina do Maranhão (UEMASUL)
Imperatriz, MA – Brasil

liliana.uemasul@gmail.com, giseleuemasul@gmail.com

Abstract. *The Quantum Computation is based on the principles of Quantum Mechanics. Unlike classical computers, quantum ones have parallelism capabilities by default, which makes their processing speed for solving problems much higher. For this performance, the use of certain algorithms is necessary. In this text, we present some studies on the Grover algorithm, with the aim of knowing and learning how to implement it in quantum simulators, in addition to analyzing and interpreting the results obtained.*

Resumo. *A Computação Quântica se baseia nos princípios da Mecânica Quântica. Ao contrário dos computadores clássicos, os quânticos possuem capacidade de paralelismo como padrão, o que torna sua velocidade de processamento para resolução de problemas muito maior. Para esse desempenho, o uso de determinados algoritmos faz-se necessário. Neste texto, apresenta-se alguns estudos sobre o algoritmo de Grover, com o objetivo de conhecer e aprender a implementá-lo nos simuladores quânticos, além de analisar e interpretar os resultados obtidos.*

1. Introdução

Os computadores quânticos são projetados para terem mais eficiência do que os computadores clássicos, podendo ser bastante utilizados em áreas como criptografia, otimização, simulação de sistemas quânticos e resolução de grandes sistemas de equações lineares. Um exemplo disso é o algoritmo de Grover [Grover 1996], um dos primeiros algoritmos quânticos a serem apresentados, é utilizado para realizar busca em banco de dados não-ordenado e forma a base da computação quântica.

A generalização do algoritmo de Grover, também chamada de estimativa de amplitude [Brassard et al. 2002], é um algoritmo quântico importante porque serve de base para outros que foram propostos posteriormente [Jordan 2021]. Estimar a amplitude está na base da maioria dos algoritmos quânticos conhecidos relacionados aos problemas de colisões e propriedades de gráficos.

Com este trabalho, deseja-se estudar o algoritmo de Grover, sua generalização e aprender a implementá-lo no Qiskit, além de analisar e interpretar os resultados obtidos com esta implementação, a escolha da utilização do simulador Qiskit foi por ser um dos mais conhecidos e utilizados além de estar disponível de forma gratuita atualmente.

Para a implementação do algoritmo de Grover, utiliza-se o Qiskit (Quantum Information Science Kit) [Qiskit 2021], um Software Development Kit (SDK) open-source

*Trabalho Concluído.

desenvolvido pela IBM para construção, otimização e execução de circuitos e algoritmos em computação quântica. Pode-se compará-lo a um pacote da linguagem Python, porque permite uma integração com o código Python que facilita muito a construção de algoritmos, dada a simplicidade da sintaxe e a comum necessidade de se executar parte dos algoritmos em conjunto com recursos de computação clássica, para resolver problemas de otimização, química quântica, física, aprendizado de máquina e finanças [Javadi-Abhari and Gambetta 2021].

2. Fundamentos e metodologia

Para a realização deste trabalho, iniciamos com estudos dos conceitos básicos de computação quântica, seguida do algoritmo de Grover e de estudos sobre simulação em circuitos quânticos, mais especificamente no Qiskit (IBM, 2021), em seguida implementar algoritmos e analisar seus funcionamento e resultados.

Um qubit representa um estado quântico ($|s\rangle = a\,|0\rangle + b\,|1\rangle$, com $a, b \in C$ e $a^2 + b^2 = 1$) e está numa superposição de zeros e uns num determinado instante.

Em um computador quântico, para que o processamento da informação aconteça, utiliza-se as portas lógicas quânticas, que são fundamentais para a construção de circuitos e que atendem a condições de normalização e implementam operações inversíveis [de Castro 2007]. Alguns exemplos de portas lógicas são:

- NOT quântica: é representada por *X* (troca o bit quântico), ou seja,

$$X\,|0\rangle = |1\rangle \;\; e\; X\,|1\rangle = |0\rangle\; X(\alpha\,|0\rangle + \beta\,|1\rangle) = \alpha X\,|0\rangle + \beta X\,|1\rangle = \alpha\,|1\rangle + \beta\,|0\rangle \tag{1}$$

- *Z*: não altera o estado $|0\rangle$, mas muda o sinal do estado $|1\rangle$ para $|-1\rangle$:

$$Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$$

- *H* (porta de Hadamard): Conhecida como porta H ou raiz quadrada da porta NOT.

$$H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

É uma porta lógica que tem como saída, os dois estados fundamentais, isto é

$$|0\rangle \Rightarrow \frac{(|0\rangle + |1\rangle)}{\sqrt{2}}; |1\rangle \Rightarrow \frac{(|0\rangle - |1\rangle)}{\sqrt{2}} \tag{2}$$

Pode-se considerar como circuitos quânticos, um mecanismo que é constituído de vários tipos de portas quânticas. À vista disso, é demonstrado a seguir a superposição de dois estados fundamentais de um computador quântico, representado pelo estado genérico $|s\rangle$. Para isso, é aplicado o produto tensorial entre duas portas lógicas quânticas, porta Hadarmard e dois estados quânticos, $|0\rangle$ e $|1\rangle$,

$$|s\rangle = (H \otimes H)(|0\rangle\,|0\rangle) \tag{3}$$

reescrevendo-os:

$$|s\rangle = H^{\otimes 2}\,|00\rangle \tag{4}$$

aplicando o produto tensorial:

$$|s\rangle = H\,|0\rangle \otimes H\,|0\rangle \tag{5}$$

substituindo o valor que correspondem a $H\,|0\rangle$, temos que:

$$|s\rangle = \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \otimes \frac{(|0\rangle + |1\rangle)}{\sqrt{2}} \tag{6}$$

resolvendo o produto tensorial, temos:

$$|s\rangle = \frac{1}{2}(|00\rangle + |01\rangle + |10\rangle + |11\rangle) \tag{7}$$

Representando, em uma base de quatro ou mais, têm-se:

$$|s\rangle = \frac{1}{2}(|0\rangle + |1\rangle + |2\rangle + |3\rangle) \tag{8}$$

Diante disso, equação [7], é constatado a superposição de estados, através de dois estados quânticos e a atuação de duas portas lógicas.

3. Algoritmo de Grover

A habilidade do computador quântico em comparação ao computador clássico é de fato notável, principalmente quando se trata da velocidade em realizar pesquisas em um determinado banco de dados. Um exemplo disso, pode ser retratado pelo algoritmo de Grover [Qiskit 2021], que usa o fenômeno de paralelismo quântico para encontrar soluções para um problema de busca não-ordenada. Nesse contexto, considere uma lista com N elementos, dos quais é necessário localizar somente um, tendo em vista, que esse elemento possui características exclusivas, que será denominado de X, conforme se apresenta em [Qiskit 2021].

Agora, imagine que cada elemento seja uma moeda de mesmo valor, ou seja, um contador $N-1$ moedas de valor 10, salvo o item X, nesse caso, a única moeda de valor 5, lembrando que esses valores foram escolhidos aleatoriamente. Essa situação se encontra na figura [1].

Figura 1. Lista com N moedas. Fonte: [Qiskit 2021]

Para resolver esse problema utilizando um computador clássico será preciso analisar, aproximadamente, $\frac{N}{2}$ ou quem sabe, olhar uma moeda por vez; já no computador quântico, o mesmo problema poderá ser solucionado a partir da amplificação de amplitude de Grover, o qual haverá uma economia de tempo considerável em relação ao computador clássico, [Qiskit 2021]. Esse modo de amplificar a amplitude é dado pelo crescimento da probabilidade de encontrar a posição do elemento X. Na figura [2], é exposto a amplitude

do objeto a ser encontrado, que será maior do que os demais, e então, resultará no item procurado.

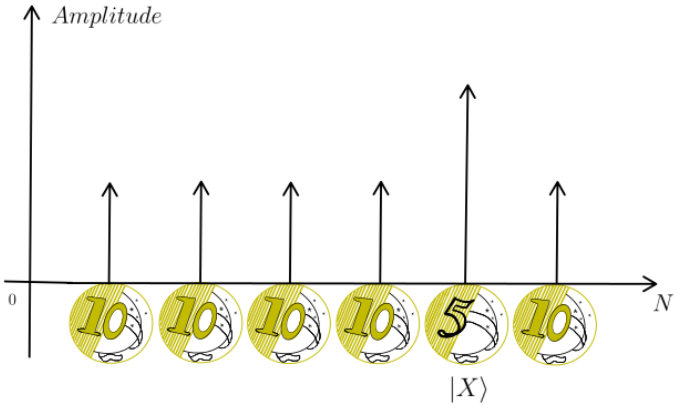


Figura 2. Representação: Amplificação de Amplitude. Fonte: [Qiskit 2021]

Em conformidade ao exemplo da moeda, é retratado na seção [4], a atuação do algoritmo de Grover em busca do estado $|X\rangle = |11\rangle$, o mesmo exibido na equação [7], em que foi usado 2 qubits, [Qiskit 2021]. É importante ser ressaltado, que a quantidade de "elementos", N consistirá em $N = 2^n$, sendo que n é o número de qubits. Portanto, N será igual a 4 estado. Precisamente como descrito na demonstração algébrica de superposição de estados em [8].

4. Implementação do algoritmo de Grover

Para implementar esse algoritmo num simulador quântico, utiliza-se a linguagem Python juntamente com a biblioteca Project Jupyter [Jupyter 2021] [1]. Por conseguinte, importa-se a biblioteca Qiskit[2] instanciamos essa classe em uma variável, tal como $circuit = QuantumCircuit\,()$. A biblioteca Quiskit exige um valor inteiro para determinar a quantidade de qbits necessários no circuito. O valor empregado na entrada dessa variável compreende a quantidade de qubits que serão usados nesse algoritmo, nesse caso, dois qubits os quais foram adicionados no objeto da variável. Em seguida, é escrito o algoritmo para desenhar essa representação, figura [3].

```
from qiskit import *
%matplotlib inline
circuit = QuantumCircuit(2)
circuit.draw(output = 'mpl')
```

Então, aplica-se a porta Hadamard no qubit 0 e no qubit 1. Para apresentar em forma de figura o circuito contruído, até o momento, utiliza-se o comando $circuit.draw(output = 'mpl')$ da classe Quiskit, como na figura [3].

```
circuit.h([0,1])
circuit.draw(output = 'mpl')
```

[1]Essa implementação do algoritmo de Grover no Qiskit, está baseada no guia, disponível em [Qiskit 2021].

[2]Para download do Qiskit, basta acessar o site `https://qiskit.org/` e seguir todos os passos apresentados pelo mesmo.

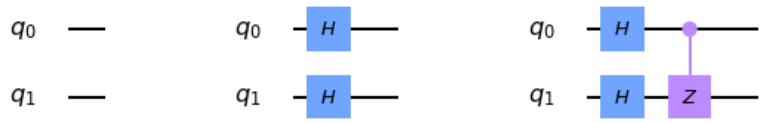


Figura 3. Podemos ver, da esquerda para a direita, a representação de dois Qubits, da aplicação das portas Hadamard e Controlada z.

Insere-se a porta controlada z, representada por cz, que será o oráculo, que tens como propósito "amplificar a amplitude", como visto no exemplo da moeda, figura [3].

```
circuit.cz(0,1)
circuit.draw(output = 'mpl')
```

Em seguida, é aplicado a porta Hadamard e a porta Z, e ainda foi adicionado uma barreira, a fim de exibir um diagrama ordenado, figura [4] e por fim, realizou-se uma medida em todos os qubits através de $circuit.measure\_all()$, figura [5].

```
circuit.h([0,1])
circuit.z([0,1])
circuit.barrier()
circuit.cz(0,1)
circuit.barrier()
circuit.h([0,1])
circuit.draw(output = 'mpl')
```

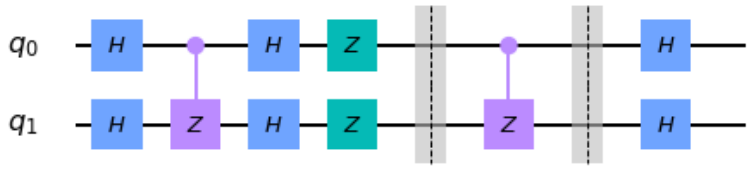


Figura 4. Aplicação de Portas Quânticas no Circuito.

```
circuit.measure_all()
circuit.draw(output = 'mpl')
```

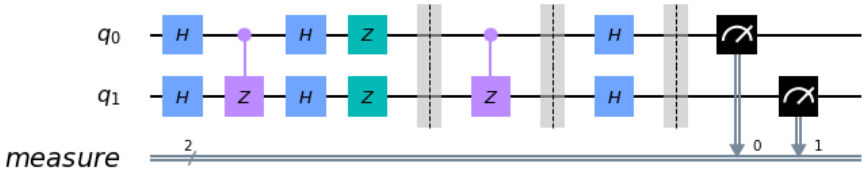


Figura 5. Medição no Circuito Quântico.

Através do simulador $qasm\ simulator$, que faz parte da lista de simuladores de grande performance da IBM, frequentemente aplicado em modelagens comuns de circuitos quânticos, [IBM 2021], obtém-se todos os estados possíveis com suas respectivas

probabilidades. E portanto, encontrou-se o estado $|11\rangle$ com probabilidade de 100%, figura [6]. [3]

```
simulator = Aer.get_backend('qasm_simulator')
result = execute(circuit, backend = simulator, shots = 1).result()
counts = result.get_counts()
from qiskit.tools.visualization import plot_histogram
plot_histogram(counts)
```

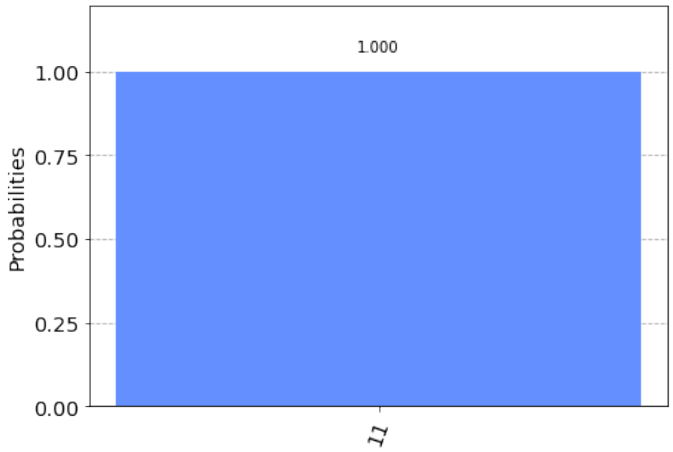


Figura 6. Representação do Estado Procurado $|11\rangle$ com probabilidade igual a 1.

5. Considerações finais

A importância de estudar o algoritmo de Grover reside no fato de que, além de ser um dos primeiros algoritmos da literatura e, portanto, servir de base para uma série de outros que foram propostos posteriormente, o algoritmo de Grover é considerado ótimo, ou seja, ele apresenta a menor complexidade, dentre todos os possíveis algoritmos que poderiam resolver o mesmo tipo de problema. Isso significa que qualquer algoritmo que acessa o banco de dados usando um dado operador matemático, este deve ser aplicado pelo menos tantas vezes quanto o algoritmo de Grover [Bennett et al. 1997], que executa uma busca em um conjunto de dados em tempo $\sqrt{N}$.

Dado o caráter probabilístico do algoritmo de Grover, a sua utilização quando se trata de um conjunto de dados não estruturado e muito grande traz agilidade à busca de forma que o resultado, certamente, tem alta probabilidade de ser o correto.

Referências

Bennett, C. H., Bernstein, E., Brassard, G., and Vazirani, U. (1997). Strengths and weaknesses of quantum computing. *SIAM journal on Computing*, 26(5):1510–1523.

Brassard, G., Hoyer, P., Mosca, M., and Tapp, A. (2002). Quantum amplitude amplification and estimation. *Contemporary Mathematics*, 305:53–74.

[3]Para uma melhor compreensão, segue o código fonte dessa implementação: `https://github.com/lianafis/Algoritmo_Grover/blob/f2bac3baaa13b085adcec967d7b2e1874e18f5b3/Grover.ipynb`

de Castro, L. N. (2007). Fundamentals of natural computing: an overview. *Physics of Life Reviews*, 4(1):1–36.

Grover, L. K. (1996). *A fast quantum mechanical algorithm for database search*.

IBM (2021 (Acessado: Setembro 30, 2021)). Ibm quantum simulators. Disponível em: `https://quantum-computing.ibm.com/services/docs/services/manage/simulator/#qasm`.

Javadi-Abhari, A.; Nation, P. and Gambetta, J. (2019 (Acessado: Setembro 23, 2021)). Qiskit – write once, target multiple architectures. Disponível em: `https://www.ibm.com/blogs/research/2019/11/qiskit-for-multiple-architectures/`.

Jordan, S. (2021 (Acessado: Setembro 23, 2021)). Quantum algorithm zoo, microsoft quantum. Disponível em: `https://quantumalgorithmzoo.org/`.

Jupyter, P. (2021 (Acessado: Abril 15, 2021)). Disponível em: `https://jupyter.org/`.

Qiskit (2020 (Acessado: Março 23, 2021)). Grover's algorithm. Disponível em: `https://qiskit.org/textbook/ch-algorithms/grover.html`.

Gramers: Agentes Pedagógicos para uma plataforma de jogos baseada em Gramática de Grafos*†

Júlia Veiga da Silva[1], Braz Araujo da Silva Junior[1], Luciana Foss[1], Simone André da Costa Cavalheiro[1]

[1]Laboratório de Fundamentos da Computação – Universidade Federal de Pelotas (UFPel)
CEP 96.010-610 – Pelotas – RS – Brasil

{jvsilva, badsjunior, lfoss, simone.costa}@inf.ufpel.edu.br

Abstract. *This paper proposes pedagogical agents to GrameStation, a Graph Grammar-based game engine. In GrameStation, creating a game corresponds to specifying a Graph Grammar. Although intuitive, the specification of a Graph Grammar may not be trivial to people who do not have previous knowledge. It makes the pedagogical agents for the platform necessary to help users during their experience. Therefore, four agents, called gramers, are proposed. Examples of the interaction of gramers in the platform are illustrated.*

Resumo. *Este artigo apresenta a proposta de agentes pedagógicos para o GrameStation, um motor de jogos baseado em Gramática de Grafos. No GrameStation, a criação de um jogo corresponde à especificação de uma gramática. Ainda que intuitiva, a especificação de uma Gramática de Grafos pode não ser trivial àqueles que não possuem conhecimento prévio. Isto torna necessária a presença de agentes pedagógicos, com o objetivo de auxiliar os usuários durante sua experiência. Neste sentido, quatro agentes, denominados gramers, são propostos. Exemplos da interação dos gramers na plataforma são ilustrados.*

1. Introdução

Em relação aos jogos digitais, ensinar a novos usuários as mecânicas e regras de um jogo pode ser desafiador. Assim, tutoriais são recursos frequentemente utilizados para auxiliar nesse processo. Para Gohl (2016), um tutorial divertido e fundamentado representa um importante recurso ao jogador, uma vez que problemas relacionados à mecânica do jogo, por exemplo, podem impedir seu progresso durante a partida, gerando frustração e, consequentemente, a perda de interesse.

Os jogos educacionais, além do entretenimento, visam o ensino de conteúdos específicos e o desenvolvimento de habilidades [Abt 1987]. Nesses jogos, a presença de personagens animados – os agentes – é comumente observada. De maneira geral, os agentes são responsáveis por apresentar tutoriais a respeito do funcionamento dos jogos, bem como guiar o usuário por fases/etapas, interagindo por meio de mensagens, alertas e elogios [Alencar e Netto 2017].

*Trabalho concluído.

†O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001, do MCTIC/CNPq (Rede Sacci), da FAPERGS, do CNPq, da SMED/Pelotas e da PREC e PRPPG/UFPel.

Diversos trabalhos propõem agentes para a apresentação de tutoriais. Santana et al. (2017), por exemplo, apresentam um agente robô desenvolvido para ser vinculado a ferramentas de ensino on-line, cujo objetivo é mediar videoaulas acerca do desenvolvimento de jogos digitais. Já Helms et al. (2015) propõem agentes que visam contribuir com o aprendizado da Matemática no Ensino Médio, auxiliando na resolução de equações algébricas e reforçando conteúdos apresentados em sala de aula de maneira lúdica.

Neste contexto, o GrameStation (GS)[1] é um motor de jogos baseado em Gramática de Grafos (GG), uma linguagem utilizada na Engenharia de Software para especificar sistemas e verificar propriedades. No GS, a especificação de um jogo corresponde à especificação de uma GG, com a definição de grafos e regras. Entretanto, ainda que a compreensão de uma GG seja intuitiva, sua especificação pode não ser trivial a quem não está habituado a esse formalismo, considerando-se, portanto, a necessidade de um tutorial para a utilização da plataforma. Assim, com o intuito de auxiliar a compreensão da manipulação/criação das GGs no GS, são propostos quatro agentes pedagógicos, denominados gramers, com a intenção de resolver esse problema.

O artigo está organizado como segue. A Seção 2 expõe a fundamentação teórica acerca de tutoriais e agentes pedagógicos, bem como a definição de GG e a apresentação do GS. A Seção 3 descreve a metodologia e proposta deste trabalho: agentes pedagógicos para o GS. A Seção 4 apresenta exemplos da interação dos agentes na plataforma. A Seção 5 discorre sobre as considerações finais e trabalhos futuros para esta proposta.

2. Fundamentação Teórica

Nesta seção são apresentados os principais conceitos abordados neste artigo.

2.1. Tutoriais

De acordo com White (2014), um tutorial é qualquer componente de um jogo digital destinado a ensinar alguém a jogar. Neste sentido, para Sweetser e Wyeth (2005), a importância de um tutorial aumenta de acordo com a complexidade de determinado jogo. Além disso, o aprendizado por meio desse recurso não deve ser entendiante, mas fazer parte da diversão.

Visando a criação de tutoriais eficazes, Gohl (2016) estabelece um conjunto composto por 16 diretrizes para o desenvolvimento de tutoriais envolventes, entre elas: 1) Envolver os jogadores e chamar sua atenção no início do jogo; 2) Fornecer uma quantidade apropriada de dicas; 3) Ensinar antecipadamente as habilidades que devem ser utilizadas posteriormente ou logo antes de serem necessárias pela primeira vez; 4) Apresentar os objetivos prioritários com clareza e antecedência; 5) Fornecer desafios em um ritmo adequado para não frustrar os jogadores; 6) Proporcionar uma experiência divertida e positiva durante o aprendizado do jogo e domínio dos desafios; 7) Dar suporte na recuperação de erros e não penalizar os jogadores, repetidamente, pela mesma falha; 8) Fornecer feedbacks sobre o progresso no jogo; 9) Prover um feedback positivo imediato para a primeira ação do jogador.

2.2. Agentes Pedagógicos

O termo "agente" é utilizado para denotar desde simples processos de hardware/software, até entidades capazes de realizar tarefas complexas. Entre as propriedades básicas de um

[1]Disponível em: https://wp.ufpel.edu.br/pensamentocomputacional/gramestation-pt/.

agente estão a autonomia, comunicação, mobilidade e flexibilidade. Já a noção de agentes pedagógicos passou a ser observada na literatura em virtude da utilização do paradigma de agentes em sistemas desenvolvidos para fins educacionais [Giraffa 1999].

Os agentes pedagógicos, portanto, atuam como tutores ou companheiros virtuais, auxiliando os estudantes no processo de aprendizagem [Giraffa 1999]. Giraffa (1999) divide os agentes pedagógicos em dois tipos: *goal-driven* (guiados por objetivos) e *utility-driven* (guiados por sua utilidade no ambiente). Agentes *goal-driven* são tutores que interagem com o usuário, realizando atividades em conjunto. Já agentes *utility-driven* realizam tarefas ligadas a atividades pedagógicas, como auxiliar o estudante a encontrar programas específicos, arquivos, diretórios, entre outros. Segundo Gratch et al. (2002), a presença de agentes em ambientes educacionais motiva os estudantes e, consequentemente, amplia o efeito da aprendizagem, dado que sua atenção é facilmente cativada por meio dos recursos visuais.

2.3. Gramática de Grafos

Uma GG é uma generalização das gramáticas de Chomsky, substituindo *strings* por grafos [Ehrig et al. 1973]. Em uma GG, os estados de um sistema são representados como grafos (definidos por vértices e arestas com origem e destino estabelecidos) e seus eventos (transições entre estados) por regras de transformação de grafos. Dessa forma, é possível realizar a descrição visual de sistemas complexos por meio de suas características e comportamentos.

Formalmente, uma GG é definida por um grafo tipo, cuja função é restringir os elementos (vértices e arestas) permitidos em um sistema; um grafo inicial (tipado sobre o grafo tipo) que especifica o estado inicial do sistema; e um conjunto de regras que definem os possíveis comportamentos do sistema. Sendo assim, a partir da especificação de uma GG, é possível simular esse sistema a fim de verificar estados alcançáveis e detectar e evitar estados indesejados.

2.4. GrameStation: Motor de Jogos Baseado em GG

O GS é um motor de jogos baseado em GG desenvolvido na plataforma Unity, na linguagem de programação C#. A plataforma possibilita o desenvolvimento de habilidades associadas ao Pensamento Computacional (PC) que, por sua vez, é definido como um processo de resolução de problemas fundamentado na Ciência da Computação [Wing 2006]. Para Wing, o PC é um processo de propósito geral que não deve ser explorado somente por profissionais da computação, mas por todos. Silva Junior (2020) relaciona características das GGs a diversas habilidades do PC, como coleta, análise e representação de dados, decomposição de problemas, abstração, algoritmos e processos, simulação e paralelismo.

No GS, o grafo tipo corresponde a uma área de declaração, o grafo inicial refere-se à organização do jogo ao início da partida e as regras representam as possíveis ações a serem realizadas pelo jogador. A plataforma é estruturada em módulos, dividindo-se em: Game Tutorial (treinamento), Game Builder (criação) e Game Player (execução). No primeiro módulo, o usuário é conduzido na criação de um jogo – o Jogo da Velha – no formato de uma GG. Já no segundo, o usuário é capaz de criar seus próprios jogos, especificando as GGs. Por fim, no terceiro módulo, o usuário consegue executar seus jogos simulando as GGs, ou seja, selecionando regras e as aplicando, de maneira adequada, no grafo inicial.

A primeira etapa de especificação de uma GG no GS refere-se à importação de recursos externos (como imagens) a serem incluídos no jogo, seguida da criação dos grafos tipo e inicial e das regras. Na especificação dos elementos do grafo tipo, algumas propriedades devem ser definidas para cada um deles: para os vértices, deve-se definir nome, aparência, cor, posição (horizontal e vertical) e tamanho; já para as arestas são definidos nome, aparência, cor, vértice-origem, vértice-destino e tamanho. Os elementos dos demais grafos herdam essas propriedades ao terem seus tipos definidos, com exceção do nome (para vértices e arestas) e da posição (apenas para vértices), também definidos nos outros grafos. Na plataforma, ainda, existem formulários que registram os valores definidos para cada uma dessas propriedades, para cada grafo da gramática. A Figura 1 ilustra, à esquerda, o grafo tipo do Jogo da Velha, responsável por definir todos os elementos permitidos na especificação adequada do jogo, definindo seus tipos e possíveis relações. Os vértices $\times$ e $\bigcirc$ representam as marcações dos dois jogadores da partida, enquanto o $\square$ corresponde às subdivisões do tabuleiro – denominadas "casas" – nas quais os jogadores colocam suas respectivas marcações. O $\triangle$, por sua vez, representa o vazio, utilizado para indicar uma casa que não está marcada por um vértice. As arestas amarela, vermelha e cinza indicam, respectivamente, a relação dos vértices $\times$, $\bigcirc$ e $\triangle$ com uma determinada casa. Por fim, as demais arestas sinalizam a relação entre casas nas direções estabelecidas pelo jogo: horizontal (azul), vertical (laranja), diagonal principal (rosa) e diagonal secundária (verde). O grafo inicial, ao centro, indica a organização dos elementos do jogo no início da partida. Visto que o tabuleiro do Jogo da Velha deve iniciar com nove casas desocupadas, é atribuída, a cada casa, a relação com o elemento vazio. As relações entre as casas nas direções estabelecidas também são indicadas.

As regras "Marcar $\times$" (à direita, acima) e "Marcar $\bigcirc$" (à direita, abaixo), por sua vez, especificam as possíveis ações dos jogadores em seus respectivos turnos. Para a aplicação dessas regras, uma condição deve ser atendida: existir uma casa desocupada no tabuleiro. Após a aplicação, um efeito é produzido: o vértice $\times$ ou o vértice $\bigcirc$ é marcado nessa casa e o vértice $\triangle$ é eliminado. Além dessas, são definidas as regras responsáveis por indicar o momento em que um dos jogadores vence a partida. Para isso, deve-se obter a sequência de três elementos iguais em uma das quatro direções permitidas. Sendo assim, são especificadas 8 regras: duas possibilidades de marcação ($\times$ e $\bigcirc$) para quatro possibilidades de direção (horizontal, vertical, diagonal principal e diagonal secundária).

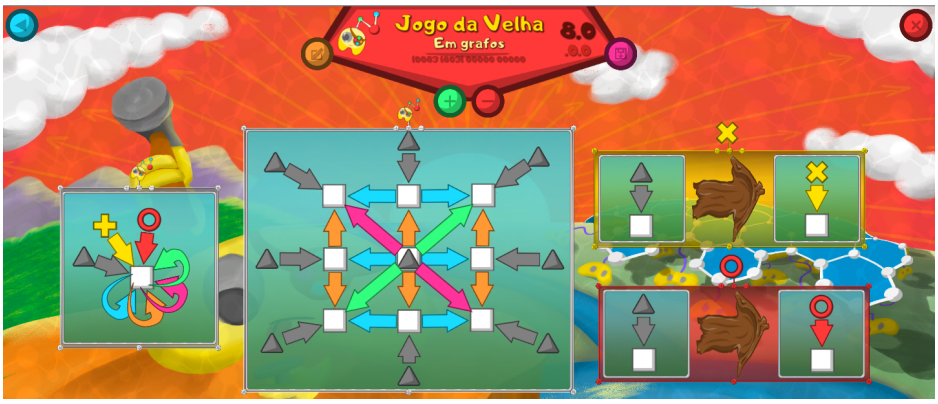


Figura 1. Grafo tipo (à esquerda), grafo inicial (ao centro) e regras (à direita)

A Figura 2 ilustra as regras que definem a vitória do jogador com marcação × na horizontal (na parte superior da figura) e do jogador com marcação ○ na vertical (na parte inferior da figura). Após a criação do jogo, é possível executá-lo e jogá-lo selecionando as regras especificadas e as aplicando no grafo inicial (Figura 3).

Figura 2. Regras "Vitória × Horizontal" (acima) e "Vitória ○ Vertical" (abaixo)

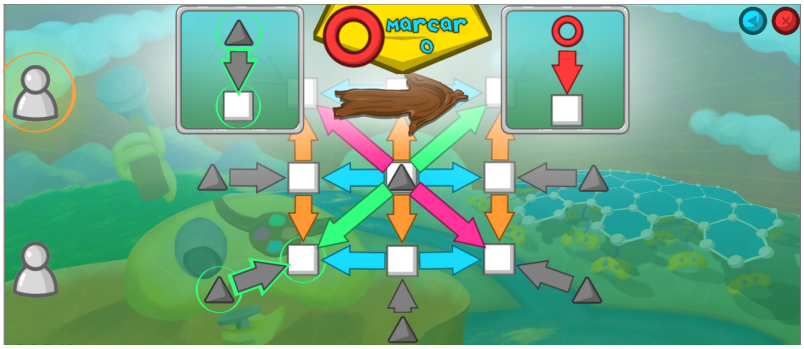


Figura 3. Aplicação da regra "Marcar ○" durante a execução do jogo

3. Agentes Pedagógicos no GS: Metodologia e Proposta

A primeira etapa deste trabalho consistiu na familiarização com o formalismo de GG e com a ferramenta. A partir das primeiras experiências, por meio da criação de jogos, foram identificadas dificuldades que, posteriormente, guiaram o processo de proposta deste trabalho, como: (1) dúvidas em relação aos termos e ações da plataforma, além dos termos e conceitos relacionados à GG – como grafo, tipo e morfismo; (2) dificuldade na definição das relações entre os elementos (estabelecidas por arestas em uma GG), as quais independem de posição ou coordenada; e (3) falta de percepção quanto às restrições impostas pelo grafo tipo, o qual exige que os demais grafos do sistema possuam, apenas, elementos e relações previamente declarados. Como resultado dessas experiências, são propostos quatro agentes pedagógicos *goal-driven* denominados gramers, que auxiliam o jogador em diferentes contextos ao longo da experiência na plataforma.

Gruly (Figura 4), a primeira gramer do GS, é proposta para contornar as dificuldades elencadas de maneira geral. Com a aparência de um *headset* de realidade virtual, apresenta o "mundo dos grafos" ao usuário, o orientando por meio de tutoriais – da execução

de ações na plataforma, à criação/manipulação de um jogo como GG. Uma abordagem frequentemente utilizada em jogos é a apresentação do tutorial em uma área de treinamento específica, na qual a mecânica do jogo é ensinada [Sweetser e Wyeth 2005]. Gruly, portanto, está localizada em uma área específica do GS, destinada à execução de tutoriais. No tutorial proposto, o usuário é guiado por Gruly a fim de realizar a criação passo a passo do Jogo da Velha – o que permite a visualização das estruturas que compõem as GGs à medida que o jogo é especificado. Para atender a problemática (1), Gruly está presente desde a tela inicial – informando sobre as funcionalidades disponíveis no GS (como criar, editar e executar um jogo) –, até a área de tutoriais, onde são apresentados os principais conceitos de GG. Já as dificuldades apontadas em (2) e (3), são facilmente contornadas por meio da criação gradual do jogo com o auxílio das explicações da gramer a cada nova etapa.

A escolha do Jogo da Velha para o tutorial considerou, além da simplicidade e popularidade, o fato do jogo apresentar-se de maneira distinta da convencional em uma GG – tanto pelo modo como os elementos são localizados, quanto em relação à forma que a matriz do jogo é "construída". Além disso, o jogo oferece a possibilidade de destacar pontos, durante a implementação, que não são óbvios àqueles que não estão habituados a esse formalismo, principalmente em relação à definição da relação entre elementos.

Conforme apontado na subseção 2.3, as GGs – tal como o GS – contêm diversos termos próprios. Logo, para habituar o usuário a essas expressões – e complementar as problemáticas (1) e (2) – propõe-se o gramer Graz (Figura 4). Com a aparência de um *joystick* de console de videogame, carrega um glossário de A a Z com palavras referentes às GGs e ao GS, bem como suas definições formais e informais, possibilitando ao usuário consultar/estudar os termos durante sua experiência. De certo modo, o jogador é introduzido a uma nova linguagem e, portanto, deve ser apresentado ao vocabulário da mesma.

Dra. Grafoss (Figura 4), gramer com a aparência do console portátil Game Boy, utiliza os elementos de um determinado grafo como fita e, quando solicitada, carrega suas informações no formulário, apresentando-as ao jogador. Esse recurso auxilia a edição de elementos pois facilita o acesso a dados específicos. Assim, o usuário é capaz de saber, exatamente, o atributo que deseja alterar.

De acordo com o modelo GameFlow, proposto por Sweetser e Wyeth (2005), entre os 8 elementos fundamentais para uma experiência divertida em jogos está o feedback. Segundo os autores, os jogadores devem receber feedback imediato sobre suas ações, além de sempre estarem cientes sobre seu status ou pontuação. Neste sentido, propõe-se a gramer Grimone (Figura 4). Também com a aparência de um *joystick* de console de videogame, sua função é exibir mensagens de acertos e erros na aplicação de regras (*matches*) e *warnings* (mensagens de aviso) durante a execução do jogo.

4. Exemplos de Interação dos Gramers no GS

O módulo Game Tutorial é destinado a usuários não habituados com o formalismo de GG. Dessa maneira, a gramer Gruly conduz o usuário para a criação de todos os grafos necessários para a execução adequada do Jogo da Velha, comunicando-se por meio de caixas de diálogo. Para a especificação do grafo tipo, por exemplo, Gruly descreve todos os elementos que o compõem, bem como o papel de cada um para o funcionamento do

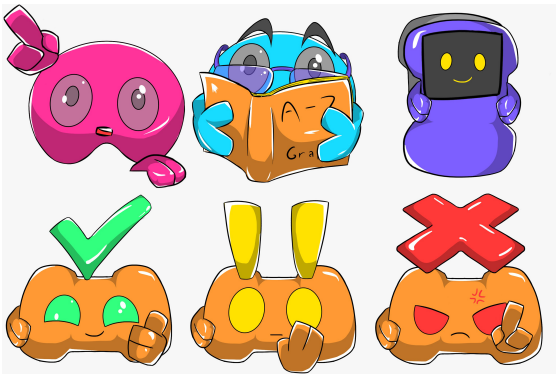


Figura 4. Gramers, da esquerda para a direita: Gruly, Graz, Dra. Grafoss (em cima) e Grimone (embaixo)

jogo. As explicações são apresentadas à medida que tarefas – como arrastar elementos para áreas indicadas na tela – são solicitadas, de modo que, ao final do tutorial, o usuário tenha criado completamente todos os grafos do jogo.

O módulo Game Builder é destinado à criação de jogos. Neste ambiente, o gramer Graz está localizado na parte superior da tela. Sua função é exibir ao jogador, por meio de caixas de diálogo, as definições formais e informais a respeito de elementos já especificados. Ao selecionar, por exemplo, o ícone relativo ao grafo tipo, Graz apresenta as seguintes informações: informalmente, o grafo tipo define os diferentes tipos de vértices e arestas que podem aparecer em uma especificação. Todos os demais grafos de uma GG são compostos por instâncias de vértices e arestas do grafo tipo; formalmente, um grafo G é uma tupla G = (V_G, E_G, src_G, tgt_G), onde V_G é um conjunto de vértices, E_G, um conjunto de arestas e src_G, tgt_G : $E_G \rightarrow V_G$ são funções totais que definem, respectivamente, o vértice de origem e destino de cada aresta.

Ainda no Game Builder, a gramer Dra. Grafoss também encontra-se na parte superior da tela. Sua função é exibir as informações de um determinado elemento pertencente a um grafo. Ao selecionar, por exemplo, um dos vértices □ do grafo inicial (Figura 1, ao centro), Dra. Grafoss o utiliza como fita – já que possui a aparência de um Game Boy – e exibe um formulário com todas as informações do elemento. Neste caso nome, aparência, cor, posição horizontal, posição vertical e tamanho.

Por fim, no módulo Game Player, o jogador consegue reproduzir o jogo criado no módulo anterior. Neste módulo, a gramer Grimone é a responsável por interagir com o usuário. Ao selecionar a regra "Marcar ○" (Figura 3), por exemplo, e aplicá-la corretamente, a gramer é exibida na parte inferior da tela junto a uma mensagem positiva ao jogador. No entanto, ao tentar aplicar a próxima regra – "Marcar ×", por exemplo – no mesmo vértice, Grimone é exibida junto a uma mensagem de erro, uma vez que a condição para a aplicação dessa regra é a existência de uma casa desocupada no tabuleiro.

5. Considerações Finais

Este trabalho apresenta a proposta da utilização de agentes pedagógicos no GrameStation, motor de jogos baseado em Gramática de Grafos. Embora visual, a especificação de uma Gramática de Grafos pode não ser trivial. Dessa forma, espera-se que a presença de

agentes tutores contribua tanto com o processo de compreensão desse formalismo, quanto com a criação/manipulação dessas gramáticas na plataforma.

O próximo passo deste trabalho refere-se à finalização da implementação dos agentes na plataforma. Após, a fase de testes será iniciada. Serão realizados testes de usabilidade, com foco na interface de software e na experiência do usuário. A primeira avaliação será realizada com estudantes de cursos de computação, divididos em dois grupos: o primeiro utilizará a plataforma com a presença dos gramers e o segundo utilizará a plataforma sem a presença dos gramers. Posteriormente, deseja-se realizar os mesmos testes com professores da Educação Básica e, mais a frente, com os estudantes. A partir dos testes espera-se analisar, com maior aprofundamento, os aspectos técnicos relevantes em relação ao desenvolvimento dos agentes e avaliar sua utilização a fim de evidenciar sua adequação ao propósito inicial.

Referências

Abt, C. C. (1987). *Serious Games*. University Press of America.

Alencar, M. A. S. e Netto, J. F. (2017). Melhorando a Colaboração em um Ambiente Virtual de Aprendizagem usando um Agente Pedagógico Animado 3D. *Brazilian Symposium on Computers in Education (Simpósio Brasileiro de Informática na Educação - SBIE)*, 28(1):1417–1426.

Ehrig, H., Pfender, M., e Schneider, H. J. (1973). Graph-grammars: An algebraic approach. In *IEEE Annual Symposium on Foundations of Computer Science*, pages 167–180.

Giraffa, L. M. M. (1999). *Uma arquitetura de tutor utilizando estados mentais*. PhD thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brasil.

Gohl, L. (2016). Usability guidelines for the creation and evaluation of functional and engaging tutorials in computer games. Trabalho de Conclusão de Curso, Universidade de Ciências Aplicadas de Hamburgo, Hamburgo, Alemanha.

Gratch, J., Rickel, J., Andre, E., Cassell, J., Petajan, E., e Badler, N. (2002). Creating Interactive Virtual Humans: Some Assembly Required. *IEEE Intelligent Systems*, 17(4):54–63.

Helms, F., Loreto, A., Adamatti, D., Buss, C., e Ferreira, A. (2015). Modelando um sistema tutor multiagentes para auxiliar na aprendizagem da matemática. *Scientia Plena*, 11(8).

Santana, A., Silva, T. R., e Aranha, E. (2017). Um Modelo de Tutor Virtual para Aulas Baseadas em Missões de Programação de Jogos Digitais. *Workshops do Congresso Brasileiro de Informática da Educação - CBIE*, 6(1):1059–1068.

Silva Junior, B. A. (2020). GGasCT: Bringing Formal Methods to the Computational Thinking. Master's thesis, Universidade Federal de Pelotas, Pelotas, RS, Brasil.

Sweetser, P. e Wyeth, P. (2005). GameFlow: A Model for Evaluating Player Enjoyment in Games. *ACM Computers in Entertainment*, 3(3):3–3.

White, M. M. (2014). *Learn to Play: Designing Tutorials for Video Games*. CRC Press.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3):33–35.

Implicações Representáveis por Funções Overlap e Grouping

Cecilia Botelho[1], **Alessandra Galvão**[1],
Adenauer Yamin [1], **Helida Santos** [2], **Renata Reiser** [1]

[1]PPGC/CDTEC/UFPEL, Pelotas – RS – Brasil
[2]C3/FURG, Rio Grande – RS – Brasil
{argalvao,cscbotelho,reiser,adenauer}@inf.ufpel.edu.br
helida@furg.br

Abstract. *QL- and D-implications are usually generated by strong negations together with t-norms and t-conorms, which are restrictive, as they demand properties such as associativity and the neutral element. In order to simplify and provide more flexibility in the conditions defining constructive methods to generate such implications, we consider dual aggregations as overlap and grouping functions. Some examples illustrate the proposal methods.*

Resumo. *As implicações de QL e D são geralmente geradas por negações fortes e t-normas e t-conormas, agregadores que exigem propriedades como associatividade e elemento neutro. A fim de simplificar e dar mais flexibilidade nas condições que definem os métodos construtivos para gerar tais implicações, consideramos a negação fuzzy máxima e as construções duais de funções overlap e grouping. Alguns exemplos ilustram os métodos propostos.*

1. Introdução

No âmbito da lógica fuzzy, funções de implicação são um elemento importante de sua constituição. Definir e utilizar funções de implicação que consigam representar diferentes cenários em sistemas de inferência fuzzy é um desafio ainda em aberto, existindo diferentes classes de implicações representáveis. Dentre elas, destacam-se as classes QL- e D-implicações [Dimuro et al. 2019]. O objetivo desse artigo é explorar métodos construtivos de gerar implicações via conceitos de funções overlap e grouping, focando nas QL- e D-implicações, com vistas a simplificar e aportar mais flexibilidade a sua estrutura.

Este artigo apresenta na Seção 2 o conceito de negação e exemplos. Na Seção 3, estudam-se funções de agregação. E, na sequência descrevem-se as QL- e D-implicações, bem como os métodos de construir tais conectivos a partir de negações, funções grouping e overlap. Ao final, tem-se a conclusão e descrição da continuidade da pesquisa.

2. Negação fuzzy

Definição 1. *Uma função* $N\colon [0,1] \to [0,1]$ *é uma negação fuzzy se:* $\mathbf{N1}$ *é decrescente, i.e.,* $N(x) \leq N(y), \forall y \leq x$*, e* $\mathbf{N2}$ *satisfaz* $N(0) = 1$ *e* $N(1) = 0$ *(condições de fronteira).*

E ainda, N é forte se é involutiva ($N(N(x)) = x, \forall x \in [0,1]$), como a negação padrão $N_S(x) = 1 - x$. Mas, um contra-exemplo para a involução é a negação máxima:

$$N_{\top}(x) = \begin{cases} 0, \text{se } x = 0; \\ 1, \text{caso contrário.} \end{cases} \tag{1}$$

Sejam $F\colon [0,1]^n \to [0,1]$ e a negação forte $N\colon [0,1] \to [0,1]$. A função N-dual de F, indicada por $F^N : [0,1]^n \to [0,1]$, é definida pela expressão:

$$F^N(x_1, x_2, \ldots, x_n) = N(F(N(x_1), N(x_2), \ldots, N(x_n))). \tag{2}$$

3. Funções de Agregações

Na Teoria dos Conjuntos, um operador de agregação caracteriza-se por agregar uma tupla de objetos que pertençam a um determinado conjunto, em um único objeto desse mesmo conjunto. No contexto da Lógica Fuzzy, os n valores em cada tupla são números reais valorados em $[0,1]$. Frequentemente, deve-se impor condições sobre uma função de agregação A, compatíveis com as aplicações na área. Recentemente, [Mesiar and Komornikova 1997] propuseram propriedades para operadores de agregação.

Um operador de agregação é uma função $A : [0,1]^n \to [0,1]$, que satisfaz:

A1 $A(x) = x, \forall x \in [0,1]$ (Identidade);
A2 $A(0,0,\ldots,0) = 0$ e $A(1,1,\ldots,1) = 1$ (Condições de Contorno);
A3 $A(x_1,\ldots,x_n) \leq A(x'_1,\ldots,x'_n)$ se $x_i \leq x'_i$, $\forall x_i, x'_i \in [0,1], i \in \mathbb{N}_n$. (Crescente).

Por **A1**, se A tem aridade-1, então coincide com a função identidade. Nas condições de contorno, atenta-se para o comportamento do agregador no pior e no melhor caso. Em **A2** parece ser fundamental na definição de operadores de agregação. Em [Mayor and Trillas 1986] foi proposta uma extensão para essa condição básica, apresentando condições fundamentais para um operador de agregação:

$$A(x,0) = A(1,0) \cdot x, \text{ e } A(x,1) = (1 - A(1,0)) \cdot x + A(1,0), \forall x \in [0,1].$$

Seguem outras propriedades para agregações fuzzy [Grabisch et al. 2011].

A4 $A(A(x_1,x_2),x_3) = A(x_1,A(x_2,x_3)), \forall x_1,x_2,x_3 \in [0,1]$ (Associatividade).

A5 $A(x_1,\ldots,x_n) \leq A(x'_1,\ldots,x'_n)$ se $x_i \geq x'_i$, $\forall i \in \mathbb{N}_n$, $x_i \in [0,1]$ (Decrescente).

A6 Seja $N_n = \{1,2,\ldots,n\}$ e $\sigma : N_n \to N_n$ uma σ-permutação, então:

$$A(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(n)}) = A(x_1, x_2, \ldots, x_n), \forall x_i \in [0,1], i \in \mathbb{N}_n \text{(Simetria)}.$$

A7 $\exists e \in [0,1]\colon A(x_1,\ldots,x_{i-1},e,x_{i+1},\ldots,x_n) = A(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n), \forall x_i \in [0,1], i \in \mathbb{N}_n$ (e-Neutralidade).

A8 Seja $\{x_{1i}\}_{i\in\mathbb{N}}$ é uma sequência convergente, então

$$\lim_{n\to\infty} A(x_{1i}, x_2, \ldots, x_n) = A(\lim_{n\to\infty} x_{1i}, x_2, \ldots x_n), \forall x_i \in [0,1], i \in \mathbb{N}_n \text{(Continuidade)}.$$

A9 $A(A(x_1,x_2),A(x_3,x_4)) = A(A(x_1,x_3),A(x_2,x_4)), \forall x_i \in [0,1], i \in \mathbb{N}_4$ (Bissimetria).

A10 $\exists a \in [0,1]\colon A(x_1,\ldots,a,\ldots,x_n) = a, \forall x_i \in [0,1], i \in \mathbb{N}_n$ (a-Absorvência).

A11 $\exists x \in [0,1]\colon A(x,x,\ldots,x) = x$ (Idempotência).

A12 $\min_{i=1}^n(x_i) \leq A(x_1,x_2,...,x_n) \leq \max_{i=1}^n(x_i)$, $\forall x_i \in [0,1], i \in \mathbb{N}_n$ (Compensação).

A13 $A(\alpha x_1,\ldots,x_n) = \ldots = A(x_1,\ldots,\alpha x_n)$, $\alpha \in [0,1]$, $x_i \in [0,1]$, $i \in \mathbb{N}_n$ (α-Migrativa).

3.1. Normas e Conormas Triangulares

A seguir, seguem conceitos, propriedades e exemplos de (co)normas triangulares [Klement and Navara 1999].

Definição 2. *Uma norma triangular (t-norma) é uma função* $T\colon [0,1]^n \to [0,1]$*, monotônica, associativa, simétrica e tem elemento* 1*-neutro.*

Definição 3. *Uma conorma triangular (t-conorma) é uma função* $S\colon [0,1]^n \to [0,1]$*, monotônica, associativa, simétrica e tem elemento* 0*-neutro.*

Exemplo 1. *Seguem exemplos de t-normas e t-conormas duais.*

As funções $T_M, S_M\colon [0,1]^2 \to [0,1]$*,* $T_M(x,y) = \min(x,y)$ *e* $T_M(x,y) = \max(x,y)$ ***são denominadas t-norma do mínimo** e **t-conorma do máximo**, respectivamente.*

As funções $T_P, S_P\colon [0,1]^2 \to [0,1]$*, dadas por* $T_P(x,y) = x.y$ *e* $S_P(x,y) = x + y - xy$ ***são denominadas t-norma do produto** e **t-norma da soma algébrica**, respectivamente.*

As funções $T_L, S_L\colon [0,1]^2 \to [0,1]$*, dadas por* $T_L(x,y) = \max(x + y - 1, 0)$ *e* $S_L(x,y) = \min(x + y, 1)$ ***são denominadas t-norma e t-conorma de Łukasiewicz**, respectivamente.*

3.2. Funções Overlap

Considera-se a generalização de funções overlap binárias [Bedregal et al. 2013] e [Bustince et al. 2010], para o contexto n-dimencional.

Definição 4. *[Gómez et al. 2016, Def 3.1] Uma função overlap n-dimensional* $O\colon [0,1]^n \to [0,1]$ *satisfaz, além das propriedades* $\mathbf{A3}$*,* $\mathbf{A6}$ *e* $\mathbf{A7}$*, as seguintes condições:*

(A14) $O(x_1, \ldots, x_n) = 0$ *se, e somente se* $\prod_{i=1}^{n} x_i = 0$*;*

(A15) $O(x_1, \ldots, x_n) = 1$ *se, e somente se* $\prod_{i=1}^{n} x_i = 1$*;*

Observe que na Definição 4, uma função overlap $O\colon [0,1]^n \to [0,1]$ satisfaz $\mathbf{A14}$, e de forma equivale, satisfaz a propriedade $\mathbf{A10}$ para 0-absorvência. Ou seja, a função overlap O tem 0 como elemento absorvente. E, a seguir, funções 0-overlap e 1-overlap são também estendidas para o cont]exto n-dimensional.

A Tabela 1 reporta da literatura, uma listagem das expressões de funções $O\colon [0,1]^n \to [0,1]$, e sua classe dentro dos operadores overlap.

Definição 5. *Considere uma função* $O\colon [0,1]^n \to [0,1]$ *que verifica todas as condições da Definição 4, exceto a* $\mathbf{A14}$*, a qual substituída por:*

(A14') $O(x_1, \ldots, x_n) = 0$ *se* $\prod_{i=1}^{n} x_i = 0$*;*

então, tem-se que O *é uma **função** 0**-overlap** n**-dimensional**. Ainda, se a função* $O\colon [0,1]^n \to [0,1]$ *verifica todas as condições da Def. 4, exceto* $\mathbf{A15}$*, a qual é substituída pela seguinte expressão:*

(A15') $O(x_1, \ldots, x_n) = 1$ *se* $\prod_{i=1}^{n} x_i = 1$*;*

então, tem-se que O *é uma **função** 1**-overlap** n**-dimensional**.*

Ao ampliar e combinar os conceitos de funções 0-overlap e 1-overlap, o conceito de funções overlap geral é definido como segue:

Definição 6. *[Miguel et al. 2019] Uma função* $O_G\colon [0,1]^n \to [0,1]$ *que, além das propriedades* $\mathbf{A3}$*,* $\mathbf{A6}$ *e* $\mathbf{A7}$*, também satisfaz as condições* $\mathbf{A14}'$ *e* $\mathbf{A15}'$*, é denominada uma **função geral-overlap** (g-overlap).*

Tabela 1. Funções Overlap n-dimensionais

Expressões Analíticas	**Tipo**
$O_{EP}(x_1,\ldots,x_n) = \frac{\prod_{i=1}^{n} x_i}{1+\prod_{i=1}^{n} x_i(1-x_i)}$	overlap
$O_{mM}(x_1,\ldots,x_n) = \min(x_1,\ldots,x_n)\max\left(x_1^2,\ldots,x_n^2\right)$	overlap
$O_s(x_1,\ldots,x_n) = \sin\frac{\pi}{2}(\prod_{i=1}^{n} x_i)^p, p>0$	overlap
$O_m(x_1,\ldots,x_n) = \min_{1\le i\le n} x_i^p, p>0$	overlap
$O_{GM}(x_1,\ldots,x_n) = \prod_{i=1}^{n} x_i^p, p>0$	overlap
$O_{HM}(x_1,\ldots,x_n) = \begin{cases} \frac{n}{\frac{1}{x_i}+\ldots+\frac{1}{x_n}}, \text{ se } x_i>0, \\ 0, \text{caso contrário.} \end{cases}$	overlap
$O_L(x_1,\ldots,x_n) = \max((\Sigma_{i=1}^{n} x_i)-(n-1),0)$	0-overlap
$O_U(x_1,\ldots,x_n) = \begin{cases} n\prod_{i=1}^{n} x_i, \text{ se } \prod_{i=1}^{n} x_i \le \frac{1}{n}, \\ 1, \text{caso contrário.} \end{cases}$	1-overlap
$O_G(x_1,\ldots,x_n) = \begin{cases} nO_L(x_1,\ldots,x_n), \text{ se } O_L(x_1,\ldots,x_n) \le \frac{1}{n}, \\ 1, \text{caso contrário.} \end{cases}$	g-overlap

Vale ressaltar a diferença entre a classe de funções overlap n-dimensionais e a classe de funções geral-overlap. Sendo que a primeira tem condições de contorno suficientes e necessárias ($\mathbf{A14}$ e $\mathbf{A15}$), enquanto a última tem apenas condições suficientes ($\mathbf{A14'}$ e $\mathbf{A15'}$). Isso significa que a classe de funções g-overlap pode resultar em 0 para alguns vetores $(x_1, ..., x_n)$, tais que $x_i \neq 0, \forall i \in \mathbb{N}_n$. Da mesma forma, pode resultar em 1 para os vetores $(x_1, ..., x_n)$, tais que $x_i \neq 1$, para algum $i \in \mathbb{N}_n$. Matematicamente, isso significa que a classe de funções g-overlap pode ter divisores zero e divisores de um.

Como consequência imediata da Definição 6, segue a proposição:

Proposição 1. *Seja $O\colon [0,1]^n \to [0,1]$ uma função overlap n-dimensional, 0-overlap ou 1-overlap, então O também é uma função geral-overlap.*

Proposição 2. *A função $G\colon [0,1]^n \to [0,1]$ é uma função geral-overlap se e somente se, pode ser definida pela expressão: $G(x_1,\ldots,x_n) = \frac{f(x_1,\ldots,x_n)}{f(x_1,\ldots,x_n)+g(x_1,\ldots,x_n)}$ sempre que $f\colon [0,1]^n \to [0,1]$ é uma função crescente ($\mathbf{A3}$) e $g\colon [0,1] \to [0,1]$ é uma função decrescente ($\mathbf{A5}$) e, ambas além de verificar as propriedades $\mathbf{A4}$, $\mathbf{A8}$ também satisfazem as seguintes condições:*

A14" *Se $\prod_{i=1}^{n} x_i = 0$ então $f(x_1,\ldots,x_n) = 0$;*

A15" *Se $\prod_{i=1}^{n} x_i = 1$ então $g(x_1,\ldots,x_n) = 1$;*

A16 $f(x_1,\ldots,x_n) + g(x_1,\ldots,x_n) \neq 0$.

Remark 1. *[Paiva et al. 2021] Dadas duas funções Overlap $O_1, O_2\colon [0,1]^n \to [0,1]$, podemos construir exemplos interessantes de funções overlap, a partir de operados de máximo, mínimo e somas conexas, como:*

1. $(O_1 \wedge O_2)(x_1,\ldots,x_n) = \min(O_1(x_1,\ldots,x_n), O_2(x_1,\ldots,x_n))$;
2. $(O_1 \vee O_2)(x_1,\ldots,x_n) = \max(O_1(x_1,\ldots x_n), O_2(x_1,\ldots,x_n))$;
3. $O_{SC}(x,y) = wO_1(x_1,\ldots,x_n) + (1-w)O_2(x_1,\ldots,x_n)$, $\forall w \in [0,1]$.

3.3. Funções Grouping

Nesta sessão, focamos no conceito de funções grouping, como complemento natural de funções overlap, apresentado por [Bustince et al. 2012, Miguel et al. 2019].

Definição 7. *[Gómez et al. 2016] Uma função* $G\colon [0,1]^n \to [0,1]$ *é uma função grouping* n*-dimensional se verifica* **A3**, **A6** *e* **A8** *e as seguintes condições:*

A17 $G(x_1,\ldots,x_n) = 0$ *se, e somente se,* $x_i = 0$, $\forall i \in \mathbb{N}_n$*;*

A18 $G(x_1,\ldots,x_n) = 1$ *se, e somente se,* $\exists i \in \mathbb{N}_n$ *tal que* $x_i = 1$.

Definição 8. *[Dimuro et al. 2019, Def 9] Uma função* $G_g\colon [0,1]^n \to [0,1]$ *é uma função geral-grouping (indicada por* G_g*) se verifica* **A3**, **A6** *e* **A8** *e as seguintes condições:*

A17' $O(x_1,\ldots,x_n) = 0$ *se* $\sum_{i=1}^{n} x_i = 0$*;*

A18' $G(x_1,\ldots,x_n) = 1$ *se* $\exists i \in \mathbb{N}_n$ *tal que* $x_i = 1$.

Exemplo 2. *Na Tabela 2, veja as funções grouping, como construções* N_S*-duais referentes às funções overlaping* n*-dimensionais da Tabela 1.*

Tabela 2. Funções grouping n-dimensionais

Expressões Analíticas	Tipo
$G_{EP}(x_1,\ldots,x_n) = 1 - \frac{\prod_{i=1}^{n}(1-x_i)}{1+\prod_{i=1}^{n} x_i(1-x_i)}$	grouping
$G_{mM}(x_1,\ldots,x_n) = 1 - \min(1-x_1,\ldots,1-x_n)\max\left((1-x_1)^2,\ldots,(1-x_n)^2\right)$	grouping
$G_s(x_1,\ldots,x_n) = 1 - \sin\frac{\pi}{2}(\prod_{i=1}^{n}(1-x_i))^p, p>0$	grouping
$G_O(x_1,\ldots,x_n) = \min_{1\le i\le n}(1-x_i)^p, p>0$	grouping
$G_{GM}(x_1,\ldots,x_n) = \prod_{i=1}^{n}(1-x_i)^P, p>0$	grouping
$G_{HM}(x_1,\ldots,x_n) = \begin{cases} 1 - \frac{n}{\frac{1}{1-x_i}+\ldots+\frac{1}{1-x_n}}, \text{ se } x_i < 1, \\ 1, \text{caso contrário.} \end{cases}$	grouping
$G_L(x_1,\ldots,x_n) = \max((1-\Sigma_{i=1}^{n}x_i), 0)$	1-grouping
$G_U(x_1,\ldots,x_n) = \begin{cases} \prod_{i=1}^{n} x_i, \text{ se } \prod_{i=1}^{n} x_i \le \frac{n^2-1}{n}, \\ 0, \text{caso contrário.} \end{cases}$	0-grouping
$G_G(x_1,\ldots,x_n) = \begin{cases} nG_L(x_1,\ldots,x_n), \text{se } G_L(x_1,\ldots,x_n) \le \frac{n^2-1}{n}, \\ 0, \text{caso contrário.} \end{cases}$	g-grouping

Proposição 3. *[Dimuro et al. 2019, Prop 1] Seja* $G\colon [0,1]^n \to [0,1]$ *uma função grouping n-dimensional, então* F *também é uma função geral-grouping.*

4. Implicações Fuzzy

Na sequência, estudamos as propriedades de implicações fuzzy, considerando duas classes de implicações representáveis: QL-implicações e D-implicações.

Definição 9. *[Fodor 1995] Uma implicação fuzzy* $I\colon [0,1]^2 \to [0,1]$ *verifica as condições:*

I1 *Se* $x \le z$ *então* $I(x,y) \ge I(z,y)$ *(antitonicidade no primeiro argumento);*
I2 *Se* $y \le z$ *então* $I(x,y) \le I(x,z)]$ *(isotonicidade no segundo argumento);*
I3 $I(1,1) = 1$*;*
I4 $I(0,0) = 1$*;*

I5 $I(1,0) = 0$*;*

Pela Definição 9, uma implicação fuzzy $I\colon [0,1]^2 \to [0,1]$ satisfaz as condições:

I6 $I(x,1) = 1, \forall x \in [0,1]$; (dominância da verdade do consequente);
I7 $I(0,y) = 1, \forall y \in [0,1]$ (dominância da falsidade do antecedente).

E, neste contexto, também satisfaz a seguinte condição:

I8 $I(0,1) = 1$ (condição de normalidade).

Logo, uma implicação fuzzy estende a implicação clássica. E, tem-se condições extras:

I9 $I(1,y) = y, \forall y \in [0,1]$ (princípio da neutralidade);
I9' $I(1,y) \leq y, \forall y \in [0,1]$;
I10 $I(x,y) = I(N(y),N(x)), \forall x,y \in [0,1]$ (simetria contrapositiva).

4.1. QL-implicações e D-implicações

Esta seção reporta as definições e principais propriedades de QL-implicações e D-implicações [Baczyński and Jayaram 2010].

Definição 10. *Uma função $I_{S,N,T}\colon [0,1]^2 \to [0,1]$ é uma QL-implicação se existir uma t-conorma $S\colon [0,1]^2 \to [0,1]$, uma de negação fuzzy forte $N\colon [0,1] \to [0,1]$ e uma t-norma $T\colon [0,1]^2 \to [0,1]$, tais que*

$$I_{S,N,T}(x,y) = S(N(x), T(x,y)), \forall x,y,z \in [0,1]. \tag{3}$$

Definição 11. *Uma função $I_{S,T,N}\colon [0,1]^2 \to [0,1]$ é uma D-implicação, sempre que existir uma t-conorma $S\colon [0,1]^2 \to [0,1]$, uma t-norma $T\colon [0,1]^2 \to [0,1]$ e, uma negação fuzzy forte $N\colon [0,1] \to [0,1]$, tais que:*

$$I_{S,T,N}(x,y) = S(T(N(x),N(y)),y), \forall x,y,z \in [0,1]. \tag{4}$$

QL-implicadores $I_{S,N,T}$ e D-implicadores $I_{S,T,N}$ estão relacionados pela simetria contrapositiva ($\mathbf{A10}$) com respeito à negação forte N, ou seja:

$$I_{S,T,N}(N(y),N(x)) = I_{S,N,T}(x,y) \text{ e } I_{S,N,T}(N(y),N(x)) = I_{S,T,N}(x,y).$$

4.2. Implicações Explicitamente Definidas por Funções Overlap e Grouping

A seguir, estendemos resultados apresentados em [Shi et al. 2008, Prop. 3.1].

Proposição 4. *Seja $N\colon [0,1] \to [0,1]$ uma negação fuzzy, considere $O\colon [0,1]^2 \to [0,1]$ uma função overlap bivariante, verificando a condição*

A10(i) $O(1,y) \leq y, \forall y \in [0,1]$*;*

e, dada $G\colon [0,1]^2 \to [0,1]$ uma função grouping bivariante, verificando a condição

A10(ii) $G(0,y) \leq y, \forall y \in [0,1]$.

Então, o operador $I_{G,N,O}\colon [0,1]^2 \to [0,1]$ definido pela expressão

$$I_{G,N,O}(x,y) = G(N(x), O(x,y)), \forall x,y \in [0,1], \tag{5}$$

verifica as propriedades $\mathbf{I2} - \mathbf{I5}$ e ainda, $\mathbf{I7}$, $\mathbf{I8}$ e $\mathbf{I9'}$.

Prova. *Se N é uma negação fuzzy, O uma função overlap verificando* $\mathbf{A10(i)}$ *e G uma função grouping verificando* $\mathbf{A10(ii)}$*, tem-se então os seguintes resultados:*

I2 *Se $y \leq y'$, $I_{G,N,O}(x,y) = G(N(x), O(x,y)) \leq G(N(x), O(x,y')) = I_{G,N,O}(x,y')$.*
I3 $I_{G,N,O}(1,1) = G(N(1), O(1,1)) = G(0, O(1,1)) = G(0,1) = 1$*;*
I4 $I_{G,N,O}(0,0) = G(N(0), O(0,0)) = G(1, O(0,0)) = G(1,0) = 1$*;*
I5 $I_{G,N,O}(1,0) = G(N(1), O(1,0)) = G(0, O(1,0)) = G(0,0) = 0$*;*
I7 $I_{G,N,O}(0,y) = G(N(0), O(0,y)) = G(1,0) = 1$*;*
I8 $I_{G,N,O}(0,1) = G(N(0), O(0,1)) = G(1,1) = 1$*;*
I9' $I_{G,N,O}(1,y) = G(N(1), O(1,y)) \leq G(0,y) \leq y$*, por* $\mathbf{A10(i)}$ *e* $\mathbf{A10(ii)}$.

Proposição 5. *Sejam $N\colon [0,1] \to [0,1]$ uma negação fuzzy, $O\colon [0,1]^2 \to [0,1]$ uma função overlap e $G\colon [0,1]^2 \to [0,1]$ uma função grouping verificando a condição* $\mathbf{A10(ii)}$*. Então, o operador $I_{G,O,N} : [0,1]^2 \to [0,1]$ definido por*

$$I_{G,O,N}(x,y) = G(O(N(x), N(y)), y), \forall x, y \in [0,1], \tag{6}$$

é uma D-Implicação que verifica as propriedades $\mathbf{I1}$*,* $\mathbf{I3 - I6}$ *e ainda,* $\mathbf{I8}$ *e* $\mathbf{I9'}$.
Prova. *Se N é uma negação fuzzy, O uma função overlap verificando* $\mathbf{A10(i)}$ *e G uma função grouping verificando* $\mathbf{A10(ii)}$*. Tem-se então os seguintes resultados:*

I1 *Se $x \leq x'$, $I_{G,O,N}(x,y) = G(O(N(x), N(y)), y) \geq G(O(N(x'), N(y)), y) = I_{G,O,N}(x',y)$.*
I3 $I_{G,O,N}(1,1) = G(O(N(1), N(1)), 1) = G(O(0,0), 1) = G(0,1) = 1$*;*
I4 $I_{G,O,N}(0,0) = G(O(N(0), N(0)), 0) = G(O(1,1), 0) = G(1,0) = 1$*;*
I5 $I_{G,O,N}(1,0) = G(O(N(1), N(0)), 0) = G(O(0.1), 0) = G(0,0) = 0$*;*
I6 $I_{G,O,N}(x,1) = G(O(N(x), N(1)), 1) = G(O(N(x), 0), 1) = G(0,1) = 1$*;*
I8 $I_{G,O,N}(0,1) = G(O(N(0), N(1)), 1) = G(O(1,0), 1) = G(0,1) = 1$*;*
I9' $I_{G,O,N}(1,y) = G(O(N(1), N(y)), y) = G(O(0, N(y)), y) = G(0,y) \leq y$*, por* $\mathbf{A10(ii)}$.

As funções overlap e grouping bivariantes, na Tabela 3, ilustram os resultados das Proposições 4 e 5, consideram-se os operadores $I_{G,N,O}$ e $I_{G,O,N}$ explicitamente definidas considerando a função negação $N_\top$ dada na Eq.(1).

Tabela 3. Construções $N_\top$-duais de pares grouping e overlap bivariantes

Funções Grouping	Funções Overlaping
$G_2^V(x,y) = \begin{cases} \frac{1}{2}(1-(1-x)^2(1-y)^2), \text{se } x,y \in [0,\frac{1}{2}]; \\ \max\{x,y\}, \text{ caso contrário.} \end{cases}$	$O_2^V(x,y) = \begin{cases} \frac{1}{2}\left(1+(2x-1)^2(2y-1)^2, \text{se } x,y \in [0,0.5[\right. \\ \min\{x,y\}, \text{caso contrário.} \end{cases}$
$G_{DB}(x,y) = \begin{cases} \frac{x+y-2xy}{2-(x+y)} & \text{se } x+y \neq 2; \\ 0, & \text{se } x+y = 2. \end{cases}$	$O_{DB}(x,y) = \begin{cases} \frac{2xy}{x+y} & \text{se } x+y \neq 0; \\ 0 & \text{caso contrário.} \end{cases}$
$G_{m\frac{1}{2}}(x,y) = 1 - \min\{\sqrt{1-x}, \sqrt{1-y}\}$	$O_{m\frac{1}{2}}(x,y) = \min\{\sqrt{x}, \sqrt{y}\}$

5. Conclusão

Neste artigo estudamos a aplicação de funções de overlap e grouping para construção de QL- e D-implicações. A proposta recupera as características destas funções propondo um modelo construtivo para a geração de implicações fuzzy. Exemplos de classes são apresentados de forma a validar os métodos. O artigo está estruturado em seções, as quais apresentam inicialmente as definições teóricas do trabalho, seguido da proposta e, finalmente, apresenta exemplos de aplicação da metodologia estudada.

Tabela 4. Implicações geradas por funções grouping G e overlap O

Funções $I_{G,N,O}$	Funções $I_{G,O,N}$
$I_{G_2^V,N_T,O_2^V}(x,y)=\begin{cases}\frac{1}{2}\left(1+(2y-1)^2\right), \text{se } x=1 \text{ e } y\geq 0.5;\\ \frac{1}{2}\left(1-(1-y)^2\right), \text{se } x=1 \text{ e } y<0.5;\\ 1, \text{ caso contrário.}\end{cases}$	$I_{G_2^V,O_2^V,N_T}(x,y)=\begin{cases}y, \text{se } x=1 \text{e } y\geq 0.5;\\ \frac{1}{2}\left(1-(1-y)^2\right), \text{se } x=1 \text{e } y<0.5;\\ 1, \text{caso contrário.}\end{cases}$
$I_{G_{DB},N_T,O_{DB}}(x,y)=\begin{cases}y, \text{se } x=1;\\ 1, \text{caso contrário.}\end{cases}$	$I_{G_{DB},O_{DB},N_T}(x,y)=\begin{cases}\frac{y}{2-y}, \text{se } x=1;\\ 1, \text{caso contrário.}\end{cases}$
$I_{Gm_{\frac{1}{2}},N_T,Om_{\frac{1}{2}}}(x,y)=\begin{cases}1-\sqrt{1-\sqrt{y}}, \text{se } x=1;\\ 1, \text{ caso contrário.}\end{cases}$	$I_{Gm_{\frac{1}{2}},Om_{\frac{1}{2}},N_T}(x,y)=\begin{cases}1-\sqrt{1-y}, \text{se } x=1;\\ 1, \text{ caso contrário.}\end{cases}$

Referências

Baczyński, M. and Jayaram, B. (2010). QL-implications: Some properties and intersections. *Fuzzy Sets Syst.*, 161:158–188.

Bedregal, B., Dimuro, G. P., Bustince, H., and Barrenechea, E. (2013). New results on overlap and grouping functions. *Information Sciences*, 249:148–170.

Bustince, H., Fernandez, J., Mesiar, R., Montero, J., and Orduna, R. (2010). Overlap functions. *Nonlinear Analysis: Theory, Methods Applications*, 72(3):1488–1499.

Bustince, H., Pagola, M., Mesiar, R., Hullermeier, E., and Herrera, F. (2012). Grouping, overlap, and generalized bientropic functions for fuzzy modeling of pairwise comparisons. *IEEE Transactions on Fuzzy Systems*, 20(3):405–415.

Dimuro, G. P., Santos, H. S., Bedregal, B. R. C., Borges, E. N., Palmeira, E. S., Fernández, J., and Bustince, H. (2019). On D-implications derived by grouping functions. In *2019 IEEE Intl Conf. on Fuzzy Systems, LA, USA, June 23-26, 2019*, pages 1–6. IEEE.

Fodor, J. C. (1995). Contrapositive symmetry of fuzzy implications. *Fuzzy Sets and Systems*, 69(2):141–156.

Gómez, D., Rodríguez, J. T., Montero, J., Bustince, H., and Barrenechea, E. (2016). n-dimensional overlap functions. *Fuzzy Sets Syst.*, 287:57–75.

Grabisch, M., Marichal, J., Mesiar, R., and Pap, E. (2011). Aggregation functions: Construction methods, conjunctive, disjunctive and mixed classes. *Inf. Sci.*, 181(1):23–43.

Klement, E. P. and Navara, M. (1999). A survey on different triangular norm-based fuzzy logics. *Fuzzy Sets and Systems*, 101(2):241–251.

Mayor, G. and Trillas, E. (1986). On the representation of some aggregation functions. In *Proceeding of ISMVL*, pages 111–114.

Mesiar, R. and Komornikova, M. (1997). Aggregation operators. *Proceeding of the XI Conference on applied Mathematics PRIM 96*, pages 193–211.

Miguel, L. D., Gómez, D., Rodríguez, J. T., Montero, J., Bustince, H., Dimuro, G. P., and Sanz, J. A. (2019). General overlap functions. *Fuzzy Sets Syst.*, 372:81–96.

Paiva, R., Santiago, R. H. N., Bedregal, B. R. C., and Palmeira, E. S. (2021). Lattice-valued overlap and quasi-overlap functions. *Inf. Sci.*, 562:180–199.

Shi, Y., Van Gasse, B., Ruan, D., and Kerre, E. (2008). On the first place antitonicity in ql-implications. *Fuzzy Sets and Systems*, 159(22):2988–3013. Theme: Logic and Algebra.

On the problem of compensatory mating in animal breeding*

Ana Paula Lüdtke Ferreira[1]

[1]Programa de Pós-graduação em Computação Aplicada
Universidade Federal do Pampa

Abstract. *Animal breeding relies on two processes to achieve its objectives: the selection and the mating systems. Mating systems devise a particular plan to perform one or more breeding goals, which often encompass improving the herd's health and maximising financial gains in animal production systems. Compensatory mating is a strategy to produce animals with more homogeneous selection trait characteristics, discarding the production of exceptional animals in favour of a more balanced herd. This paper defines and investigates the complexity class of the optimal compensatory mating problem, proving that a polynomial-time algorithm can solve it.*

Resumo. *O melhoramento animal depende de dois processos para atingir seus objetivos: os sistemas de seleção e de acasalamento. Os sistemas de acasalamento estabelecem um plano específico para atingir um ou mais objetivos de produção, que muitas vezes incluem a melhora da saúde do rebanho e a maximização dos ganhos financeiros nos sistemas de produção animal. O acasalamento compensatório é uma estratégia para produzir animais com características de seleção mais homogêneas, descartando a produção de animais excepcionais em favor de um rebanho mais equilibrado. Este artigo define e investiga a classe de complexidade do problema de acasalamento compensatório ótimo, provando que um algoritmo de tempo polinomial pode resolvê-lo.*

1. Introduction

Animal breeding involves directing the next generation's genetics towards financial profits for the production system, using essentially two strategies: the selection and the mating systems. The selection process works by choosing the best animals that will reproduce and spread their genes within the population, increasing the frequency of desirable traits and decreasing the undesirable ones. Genetic or phenotypic similarities are the keys to the development of mating systems. Parents can pass their genetic superiority on to their progeny, and the obtained genetic gains are cumulative and long-lasting for later generations [Falconer and Mackay 1996]. Thus, in this case, the goal is to optimize the mating options, considering the animals of better genetic values. For example, some dam (a female parent) may have several sire (male) possibilities to mate, producing similar genetic results. Furthermore, mating systems may have different goals.

The mating system configured so that the best males reproduce with the best females and, consequently, the worst males reproduce with the worst females is called mating between peers. That strategy aims to obtain extreme products, that is, animals that are exceptional in specific characteristics. On the other hand, a compensatory or corrective

*Resultado finalizado.

mating strategy seeks to bring more homogeneity within the population. A compensatory mating system works by mating individuals differing in performance. For instance, an excellent male, regarding a particular trait, mates with a female deficient in that same trait [Eler 2017, Cardoso 2009].

The compensatory mating problem has not been dealt with in the literature from an algorithmic perspective, as far as we can tell. Both selection and mating processes are performed by people, analysing the particular animals involved. However, animal breeding related problems do have described algorithmic approaches [Nayeri et al. 2021]. The focus on those approaches is twofold: maximising expected offspring trait values and minimising co-ancestry and inbreeding. The literature presents several attempts to solve the first problem. Research on that matter focuses mainly on metaheuristics based on genetic algorithms [Carvalheiro et al. 2010, Kinghorn 2011, Barreto Neto et al. 2014, Carvalho et al. 2016, da Fontoura et al. 2020]. However, a recent development has shown an optimal polynomial-time greedy strategy to solve the problem. The second problem is essential for animal breeding programs since breeding exceptional animals will dramatically increase the herd's inbreeding levels. Inbreeding creates issues such as inbreeding depression, which is the reduced survival and fertility of offspring of related individuals in several production traits [Mi et al. 1965, Charlesworth and Willis 2009].

In this paper, we use the optimal algorithm presented in [Ferreira et al. 2021] to tackle the compensatory mating problem. We define optimal compensatory mating as the one that is both maximal in the sense of offspring values and has a minimum variance. We show that the problem belongs to the complexity class **P** [Arora and Barak 2009] by reducing it to the minimum assignment in weighted bipartite graphs (Sec. 2). We also present a discussion on the work presented herein and point to further developments (Sec. 3).

2. Problem characterization and analysis

A *selection process* intends to elect the animals to become the next generation parents [Yokoo et al. 2019, Simões et al. 2020, Weigel et al. 2017, Dunne et al. 2021]. The selection process analyses the herd's genetic/economic values and phenotypes of genetic/economic interest. The value of each animal is computed from a selection index, chosen accordingly to the elicited breeding objectives. A *selection index* expresses the relative reward/punishment of each trait to consider in the breeding ahead. Formally, a *selection index* is a pair $\mathcal{I} = (T, w)$ where T is a set of animal traits and $w : T \to \mathbb{R}$ is the index weighting function.

Given a selection index $\mathcal{I}$, the *individual value* of each animal $a \in A$ is the sum of its individual characteristics' values weighted by the relative importance of each characteristic accordingly to $\mathcal{I}$. I.e., the summation given in (1), where $\nu : T \times A \to \mathbb{R}$ outputs $a \in A$'s value for each trait $t \in T$. When the selection index is understood from the context, we write $\iota(a)$ instead of $\iota^{\mathcal{I}}(a)$.

$$\iota^{\mathcal{I}}(a) = \sum_{k \in T} \nu(k, a) \cdot w(k) \tag{1}$$

The mating contribution is the average of both parent's individual values, except when they have a co-ancestry above an acceptable level. The breed of related animals

shrinks genetic diversity and lead to undesirable health issues. Inbreeding strategies are helpful to fix a desirable phenotype within a population, but it is usually undesirable in the mating process. Animal breeding programs control the degree of kinship amongst animals, making them available to producers and researchers. As in the case of traits, databases or spreadsheets maintain animal kinship information. We assume that the producer can choose whether inbreeding is acceptable and, if so, to what degree. Definition 1 inserts that idea into the calculation of mating pairs expected offspring contribution value.

Definition 1 (Mating contribution) *Let $A = M \cup F$ be a set of animals, where M is the set of sires and F is the set of dams, with $M \cap F = \emptyset$, $\mathcal{I}$ be a selection index, $r : A \times A \to \mathbb{R}^+$ be the degree of which two animals are related, and l be the maximum inbreeding threshold. The* mating contribution *$\pi : M \times F \to \mathbb{R}$ of a sire $m \in M$ and a dam $f \in F$ is computed as follows:*

$$\pi(m, f) = \begin{cases} (\iota^{\mathcal{I}}(m) + \iota^{\mathcal{I}}(f))/2 & r(m, f) \leq l \\ -\infty & r(m, f) > l \end{cases} \tag{2}$$

where $\iota^{\mathcal{I}}$ is the individual contribution of each animal, given selection index $\mathcal{I}$.

The mating contribution is computed accordingly to the expected offspring trait values of each dam/sire pair, which is the average of both parents' contributions.

Problem 1 (Optimal mating assignment)

Input*: A set of sires M, a set of dams F, a selection index $\mathcal{I}$, a contribution function $\pi : M \times F \to \mathbb{R}$ computed as in Definition 1, a function $\max : M \to \mathbb{N}$ expressing the use limit of each sire in the mating process.*

Output*: A function $b^* : F \to M$ obeying the use limit for each sire, i.e. for each $m \in M$ we have that $|\{f \mid b^*(f) = m\}| \leq \max(m)$, and such that the sum $\sum_{f \in F} \pi(b^*(f), f)$ is maximal, i.e., for each other function $b : F \to M$ obeying the sire's use limit we have*

$$\sum_{f \in F} \pi(b(f), f) \leq \sum_{f \in F} \pi(b^*(f), f) \tag{3}$$

In [Ferreira et al. 2021] we present several results concerning strategies for finding optimal mating assignments, including an optimal greedy strategy to solve Problem 1. The proposed algorithm puts the problem within the complexity class **P**, since it runs in worst-case $O(|M|.|F|)$ time. We have also shown that given two mating pairs $(m_1, f_1), (m_2, f_2) \in M \times F$, we have that $\pi(m_1, f_1) + \pi(m_2, f_2) = \pi(m_2, f_1) + \pi(m_1, f_2)$, as long as the maximal accepted inbreeding is not exceeded between pairs (m_2, f_1) and (m_1, f_2). That result can be the basis for an algorithm to build a solution for the compensatory mating problem while retaining the solution's optimality since both total and average mating values are unchanged by the swapping of mated pairs. Theorem 1 proves that claim.

Theorem 1 *Let F be a set of dams, M be a set of sires, and $b : F \to M$ be a mating function. Let s_b be the total value of function b, or*

$$s_b = \sum_{f \in F} \pi(b(f), f) \tag{4}$$

Let $f_1, f_2 \in F$ such that neither $\pi(b(f_2), f_1)$ nor $\pi(b(f_1), f_2)$ exceed the inbreeding maximum limit (i.e., neither value equals $-\infty$), and let

$$s'_b = \pi(b(f_2), f_1) + \pi(b(f_1), f_2) + \sum_{f \in F \setminus \{f_1, f_2\}} \pi(b(f), f)$$

then $s_b = s'_b$.

Proof: The proof is straightforward given how each animal pair contribution is computed:

$$\begin{array}{rl} s'_b = & \pi(b(f_2), f_1) + \pi(b(f_1), f_2) + \sum_{f \in F \setminus \{f_1, f_2\}} \pi(b(f), f) \\ = & \iota(b(f_2))/2 + \iota(f_1)/2 + \iota(b(f_1))/2 + \iota(f_2)/2 + \sum_{f \in F \setminus \{f_1, f_2\}} \pi(b(f), f) \\ = & \pi(b(f_1), f_1) + \pi(b(f_2), f_2) + \sum_{f \in F \setminus \{f_1, f_2\}} \pi(b(f), f) \\ = & \sum_{f \in F} \pi(b(f), f) \\ = & s_b \end{array}$$

. □

Corollary 1 *Changing mated pairs within an optimal solution for Problem 1 dos not alter the solution average μ.*

Proof: It follows directed from Theorem 1: if $s_b = s'_b$ then $\mu_b = \mu'_b$, given that $\mu_b = s_b/|F|$ and $\mu'_b = s'_b/|F|$. □

Theorem 1 shows that changing mating pairs, as long as each the match is under the inbreeding limit, does not affect neither the total solution value nor its mean.

Problem 2 formalizes the problem we intend to tackle.

Problem 2 (Optimal Compensatory Mating problem)

Input*: A tuple $S = (M, F, \pi, b^*)$ where M is a set of sires, F is a set of dams, π is the contribution function described in (2), and $b^* : F \to M$ is an optimal mating function given as the solution of Problem 1.*

Output*: A function $b^c : F \to M$ with the same total value as b^*, i.e., $\sum_{k \in F} \pi(b^c(k), k) = \sum_{k \in F} \pi(b^*(k), k)$, and such that the solution variance is minimum. I.e, for any other function $b' : F \to M$ with $\sum_{k \in F} \pi(b'(k), k) = \sum_{k \in F} \pi(b^*(k), k)$ we have that*

$$\frac{1}{|F|} \sum_{f \in F} (\pi(b'(f), f) - \mu_{b'})^2 \leq \frac{1}{|F|} \sum_{f \in F} (\pi(b^c(f), f) - \mu_{b^c})^2 \tag{5}$$

The greedy strategy used in [Ferreira et al. 2021] does not work for Problem 2 because searching for minimum variance will not preserve the maximal function value. However, we can take advantage of the fact that changing already mated pairs belonging to the optimal solution will not change its value. We will use a reduction strategy [Garey and Johnson 1979] to show that Problem 2 have a polynomial-time solution. The reduction will also give an algorithm to solve the problem, albeit not necessarily the most efficient one. Specifically, we will prove the existence of a polynomial-time reduction

from the optimal compensatory mating problem to the assignment problem of weighted bipartite graphs.

A bipartite graph is a graph $G = (V, E)$ where the vertex set V has a partition $\{L, R\}$, such that $L \cap R = \emptyset$ and $L \cup R = V$, induced by the edge set E, as follows: for each $(v_1, v_2) \in E$ we have that $v_1 \in L$ and $v_2 \in R$. For that reason, bipartite graphs are commonly referred as a tuple $G = (L, R, E \subseteq L \times R)$. The definition of weighted graphs extends naturally to bipartite graphs, where a weighting function $w : E \to \mathbb{R}$ attributes a value to each edge $e \in E$. The *minimum assignment problem* for weighted bipartite graph can be defined as follows:

Problem 3 (Minimum assignment in weighted bipartite graphs)

Input*: A weighted bipartite graph* $G = (L, R, E, w : E \to \mathbb{R})$

Output*: An* assignment *from* L *to* R *which is minimum, i.e., a function* $a^* : L \to R$ *such that for each* $v \in L$ *we have that* $(v, a^*(v)) \in E$*, and for any other function* $a : L \to R$ *with* $(v, a(v)) \in E$*, the following holds:*

$$\sum_{v \in L} w(v, a^*(v)) \leq \sum_{v \in L} w(v, a(v)) \tag{6}$$

The assignment problem in weighted bipartite graphs can be trivially modelled as an integer linear programming problem, which is **NP**-complete in the general case [Garey and Johnson 1979]. However, the literature presents several polynomial-time algorithms to solve the problem faster. When the graph is *balanced*, i.e., when $|L| = |R|$, the Kuhn-Munkres algorithm (also known as the Hungarian method) [Kuhn 1955, Munkres 1957] can solve the problem in $O(mn + n^2 \log n)$ [Ramshaw and Tarjan 2012] where $n = |L| = |R|$ and $m = |E|$, proving it belongs to the complexity class **P**.

A *reduction* from a problem P_1 to a problem P_2, with (respectively) inputs I_1 and I_2, and outputs O_1 and O_2 is a pair of functions $f_I : I_1 \to I_2$ and $f_O : O_1 \to O_2$ such that for any problem instance $x \in I_1$ we have that $P_1(x) = f_O(P_2(f_I(x)))$. The function notation to represent problems P_1 and P_2 derives from the problem definition as a mapping between input and outputs instance values (i.e., a function). A reduction gives us an alternative algorithm for solving the problem P_1 if an algorithm for solving P_2 is known. Furthermore, the complexity of P_1 is bounded by the sum of f_I, P_2, and f_O complexities [Arora and Barak 2009].

Problem 1 does not make any assumptions regarding the relative number of sires or dams nor it requires an equal number of animals in each set. Actually, the number of dams usually exceeds the number of sires by a large amount. However, the problem solution delivered by the greedy algorithm described in [Ferreira et al. 2021] is a surjective function from F to the actual image of the mating function in M. We will use the solution of Problem 1, given as input to Problem 2 to build two sets of equal size. The reduction from the Compensatory Mating problem to the assignment problem for weighted bipartite graph appears in Definition 2.

Definition 2 (Problem Reduction) *Let* F *be a set of dams,* M *be a set of sires,* π *be the pair contribution function and* $b : F \to M$ *be an optimal solution to the Mating Selection*

Maximization problem. Let $n_b : M \to \mathbb{N}$ *be the function* $n_b(m) = |\{f \in F \mid b(f) = m\}|$*, i.e., the function that for each sire returns the number of dams it was mated with. Let* $G = (F, M', E, w : E \to \mathbb{R})$ *be a weighted bipartite graph built as follows:*

- $M' = \cup_{m \in M}\{[m, i] \mid 1 \leq i \leq n_b(m)\}$
- $E = \{(f, [m, i]) \mid 1 \leq i \leq n_b(m), \pi(m, f) \neq -\infty\}$
- $w(f, [m, i]) = (\pi(m, f) - \mu_b)^2$*, for each* $w([m, i], f) \in E$*, with* $1 \leq i \leq n_b(m)$

The assignment problem for weighted bipartite graph output when presented with G *as input is the function* $a^* : F \to M'$*, which serves as basis to build the Compensatory Mating problem solution* $b^c : F \to M$ *as follows:* $b_c(f) = m$*, for each* $a^*(f) = [m, i]$*, for some* $1 \leq i \leq n_b(m)$*.*

The bipartite graph built by the reduction process contains the set of dams F as its left side all the elements in M belonging to the actual domain of the mating function b, multiplied by the number of times they appear in the mating function. Therefore, both sides of G, namely F and M', have equal sizes. The graph edges connect each dam-sire pair below the inbreeding limit. Since it is a minimisation problem, the missing edges could exist with a very large weight between them (represented as ∞). We have omitted them for the sake of clarity, knowing that graph algorithms that require complete graphs represent missing edges in that way. The weighting function informs how far the pair's contribution is from the mean function value μ_b or, in other words, how the mating of those animals contributes to the solution's variance. Therefore, minimising the total weight of the graph assignment problem actually indicates the mating function that delivers the lowest variance value. We prove that claim in Theorem 2.

Theorem 2 *Definition 2 presents a correct reduction from Problem 2 to Problem 3.*

Proof: Let $G = (F, M', E, w : E \to \mathbb{R})$ be the weighted bipartite graph built as in Definition 2. Theorem 1 assures that changing already mated pairs within an optimal solution does not change its value. Therefore, the requirement of Problem 2 is met by construction: the only sires within the set M' are those belonging to the optimal solution b in Definition 2. Now, let the solution for the assignment problem for weighted bipartite graph for G be a function $a^* : F \to M'$ such that $\sum_{f \in F} w(f, a^*(f))$ is minimum. Since $w(f, a^*(f)) = (\pi(a^*(f), f) - \mu_b)^2$, we have that $\sum_{f \in F}(\pi(a^*(f), f) - \mu_b)^2$ is minimum, where μ_b is the average value of b. The output reduction function maps each $a^*(f) = [m, i]$, for some $1 \leq i \leq n_b(m)$, to the pair (m, f). Therefore, the output function $b_c : F \to M$ where $b_c(f) = m$ whenever $a^*(f) = [m, i]$ assures that the value $s_c = \sum_{f \in F}(\pi(b_c(f), f) - \mu_b)^2$ is minimum. The variance of b_c, $\sigma_c^2 = s_c/|F|$ is also minimum, since $|F|$ is a constant value. □

Corollary 2 *The Optimal Compensatory Mating problem (Problem 2) belongs to the complexity class* ***P****.*

Proof: Direct from the fact it can be solved by the reduction $f_O(P_2(f_I(x)))$, which has worst-case running time $O(n^2) + O(mn + n^2 \log n) + O(n) = O(mn + n^2 \log n)$ [Ramshaw and Tarjan 2012]. In the case of the assignment problem for weighted bipartite graph, that expression translates into $O((|M|.|F|)|F|+|F|^2 \log |F|)$, which is polynomial in the number of animals involved. □

3. Conclusion

Mating systems devise a particular strategy to perform one or more breeding goals. Breeding goals are usually related to health issues and economic gains. Producers use selection indexes to measure each animal value. Arranging the pairing of sires and dams can lead to different results, such as the production of exceptional animals or to a more homogeneous herd. In either case, one wants to maximise the total herd next generation's expected value.

This paper shows a reduction from the compensatory mating problem to the minimal assignment for weighted bipartite graphs. A compensatory mating intends to produce a more homogeneous offspring, given the breeding goals expressed as a selection index. The literature on breeding problems does not appear to discuss their complexity classes thoroughly. This paper gives a sound contribution to the matter by showing a polynomial-time algorithm to solve it.

We can devise further developments from the results presented here. First and foremost, the problem appears to admit a more efficient solution than the Hungarian algorithm. The same happened to the maximisation problem for the mating assignment: although we could have used the same reduction presented in this paper, we have developed an optimal greedy strategy with worst-case quadratic execution time [Ferreira et al. 2021].

Another critical problem, which is the minimisation of the herd inbreeding levels, will be carried on in the future. All algorithms developed so far rely on a parameter specifying the maximum number of times a sire can participate in the breeding process. That parameter is a rule of thumb, not assuring the co-ancestry relations within the generations produced over time. That information is critical for animal breeding programs. We intend to explore the results reached so far to create a model of long-term strategies to achieve breeding goals whilst preserving the co-ancestry levels.

References

Arora, S. and Barak, B. (2009). *Computational Complexity: a Modern Approach*. Cambridge University Press, Cambridge, UK.

Barreto Neto, A. D. et al. (2014). Estrutura populacional e otimização de esquemas de acasalamento em ovinos com uso de algoritmos evolucionários.

Cardoso, F. (2009). Ferramentas e estratégias para o melhoramento genético de bovinos de corte. *Embrapa Pecuária Sul-Documentos (INFOTECA-E)*.

Carvalheiro, R., Queiroz, S. A. d., and Kinghorn, B. (2010). Optimum contribution selection using differential evolution. *Revista Brasileira de Zootecnia*, 39(7):1429–1436.

Carvalho, T., Santos, N., Lira, W., Oliveira, P. A., Neto, P. S., Lindenberg, J., and Rabêlo, R. (2016). Um sistema de informação para melhoramento genético de caprinos e ovinos. In *Anais Principais do XII Simpósio Brasileiro de Sistemas de Informação*, pages 100–107. SBC.

Charlesworth, D. and Willis, J. H. (2009). The genetics of inbreeding depression. *Nature Reviews Genetics*, 10:783–796.

da Fontoura, D. C. N., Camargo, S. S., de Almeida Torres Junior, R. A., Carvalho, H. G., and Cardoso, F. F. (2020). Optimizing mate selection: a genetic algorithm approach. In *Proceedings of the ICAR Conference*, ICAR Technical Series no. 24 - NEW TRAITS AND ADDING VALUE TO THE RECORDING AND ID SERVICES IN THE ANIMAL PRODUCTION, Prague, CZ.

Dunne, F. L., Evans, R. D., Kelleher, M. M., Walsh, S. W., and Berry, D. P. (2021). Formulation of a decision support tool incorporating both genetic and non-genetic effects to rank young growing cattle on expected market value. *Animal*, 15:100077.

Eler, J. P. (2017). *Teorias e métodos em melhoramento genético animal: sistemas de acasalamento*. Faculdade de Zootecnia e Engenharia de Alimentos da USP.

Falconer, D. S. and Mackay, T. F. C. (1996). *Introduction to Quantitative Genetics*. Longman Group Ltd., Harlow, 4 edition.

Ferreira, A. P. L., Yokoo, M. J.-I., and Motta, B. E. T. (2021). On the problem of optimal mating in animal breeding. In *Simposio Latinoamericano de Teoría Computacional, Proceedings of the XLVII Conferencia Latinoamericana de Informática (CLEI 2021) (to be pulished in IEEE Xplore)*, San José, Costa Rica.

Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, 1st edition.

Kinghorn, B. P. (2011). An algorithm for efficient constrained mate selection. *Genetics Selection Evolution*, 43(1):4.

Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Mi, M. P., Chapman, A. R., and Tyler, W. J. (1965). Effect of mating system on production traits in dairy cattle. *J. Dairy Sci.*, 48:77–84.

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 5(1):32–38.

Nayeri, S., Sargolzaei, M., and Tulpan, D. (2021). A review of traditional and machine learning methods applied to animal breeding. *Animal Health Research Reviews*, 20:31–46.

Ramshaw, L. and Tarjan, R. E. (2012). On minimum-cost assignments in unbalanced bipartite graphs. Technical Report HPL-2012-40.

Simões, M. R. S., Leal, J. J. B., Minho, A. P., Gomes, C. C., MacNeil, M. D., Costa, R. F., Junqueira, V. S., Schmidt, P. I., Cardoso, F. F., Boligon, A. A., and Yokoo, M. J. (2020). Breeding objectives of brangus cattle in brazil. *Journal of Animal Breeding and Genetics*, 137(2):177–188.

Weigel, K. A., VanRaden, P. M., Norman, H. D., and Grosu, H. (2017). A 100-year review: Methods and impact of genetic selection in dairy cattle—from daughter–dam comparisons to deep learning algorithms. *J. Dairy Sci.*, 100(12):10234–10250.

Yokoo, M. J.-I., de Oliveira Seno, L., Oliveira, L. C., da Costa, P. U., da Silva, G. M., Suñé, R. W., and Cardoso, F. F. (2019). Economic selection index in small rural dairy farms. *The Journal of Dairy Research*, 86(1):25–33.

Proposta de Atividades Lúdicas no Minecraft Educacional para a Promoção do Pensamento Computacional

Jonnhy M. Marques[1], **Luciana Foss**[1], **Simone André Da Costa Cavalheiro**[1]

[1]Universidade Federal de Pelotas - UFPEL
Rua Gomes Carneiro, 1 - CEP 96.010-610 - Pelotas - RS - Brasil

{jmmarques,lfoss,simone.costa}@inf.ufpel.edu.br

Abstract. *Computational Thinking is a problem solving methodology based on Computer Science concepts that can be applied in different areas, not only in computing. This paper presents a set of activities for children that aim to intentionally address Computational Thinking skills and concepts, such as abstraction, decomposition, pattern recognition and algorithmic thinking, using Minecraft Educational.*

Resumo. *O Pensamento Computacional é uma metodologia de resolução de problemas baseada em conceitos da Ciência da Computação que pode ser aplicada em diferentes áreas, não somente na computação. Esse trabalho apresenta um conjunto de atividades para crianças que visam abordar de forma intencional as habilidades e conceitos do Pensamento Computacional, tais como abstração, decomposição, reconhecimento de padrões e pensamento algorítmico, fazendo uso do Minecraft Educacional.*

1. Introdução

Inovações tecnológicas estão em constante progresso e causam muitas mudanças em nosso cotidiano, produzindo um grande impacto econômico e social. Entre elas o modo como o Pensamento Computacional (PC) passou a ser estudado. Segundo Jeannette Wing (2006), o PC é o processo de pensamento envolvido na formulação de um problema e na expressão de suas soluções de forma que um computador ou ser humano possa executar com eficácia.

No âmbito da educação, várias pesquisas estão sendo feitas com base nos conceitos e nas habilidades do PC, inclusive voltadas para agregar novos saberes aos professores no contexto profissional [Urzêda et al. 2020, Csizmadia et al. 2015]. Também pode-se destacar trabalhos que propõem o uso de jogos educacionais para auxiliar no ensino e qualificar a aprendizagem [COSTA 2015], tendo como base as habilidades do PC.

O Minecraft Educacional [Microsoft 2021] é uma plataforma de aprendizagem baseada em jogos que promove a criatividade, a colaboração e a solução de problemas em um ambiente virtual. O Minecraft está presente na vida de muitas crianças e adolescentes [KNITTEL et al. 2017], sendo um ambiente familiar a esses indivíduos, o que pode tornar o processo de ensino-aprendizagem mais atrativo, além de permitir o desenvolvimento de atividades divertidas, lúdicas e criativas. A própria ferramenta disponibiliza diversas atividades que abordam diferentes temas, tais como: matemática, ciências, geografia, artes, entre outros [Microsoft 2006]. Já as atividades que se propõem desenvolver habilidades relacionadas com o PC estão geralmente associadas ao ensino de programação. Outras

habilidades e conceitos (como abstração, reconhecimento de padrões e decomposição) são abordados de forma periférica, não fazendo parte dos objetivos das atividades. Desta forma, este trabalho propõe um conjunto de atividades para trabalhar de forma explícita tais habilidades.

O restante deste artigo está organizado como segue. A Seção 2 apresenta brevemente os conceitos e habilidades do PC abordados neste trabalho. Na Seção 3 são detalhadas as atividades propostas. A Seção 4 descreve quais e como as habilidades e conceitos do PC são trabalhadas em cada atividade. Finalmente, na Seção 5 são feitas algumas considerações finais e indicados os trabalhos futuros.

2. Pensamento Computacional

Jeannette Wing declarou que "O pensamento computacional é uma habilidade fundamental para todos, não apenas para cientistas da computação" [Wing 2006]. Desde então, o termo Pensamento Computacional (PC) vem sendo usado e referenciado de diferentes maneiras [Kalelioglu et al. 2016]. Além disso, existem diversas propostas relacionado o PC com diferentes habidades e conceitos [ISTE and CSTA 2011, Selby and Woollard 2013, Royal Society 2017, Denning and Tedre 2019]. Neste trabalho, o PC é visto como uma metodologia para resolução de problemas baseada em conceitos da Ciência da Computação. Dentre estes conceitos estão a abstração, decomposição, reconhecimento de padrões e pensamento algorítmico.

A abstração consiste em simplificar a realidade para focar nos aspectos mais relevantes do problema e ignorar os demais, permitindo assim lidar com problemas complexos [Wing 2008]. Já a decomposição é uma técnica para resolução de problemas que consiste em dividir um problema em partes menores, solucionar cada parte e combinar essas soluções para resolver o problema original. Essa técnica permite que se consiga resolver problemas mais complexos de forma mais simples [Ribeiro et al. 2017]. O reconhecimento de padrões é um processo de identificação de semelhanças compartilhadas por diferentes problemas, tanto nas informações quanto nas estratégias de resolução [Csizmadia et al. 2015]. Essa identificação permite reformular soluções para que possam resolver uma classe de problemas semelhantes. Por sua vez, o pensamento algorítmico refere-se ao processo de se chegar a uma solução através de uma definição clara de etapas [Futschek 2006]. Está diretamente relacionado a um conjunto de habilidades conectadas à construção e compreensão de algoritmos [Curzon et al. 2014] como, por exemplo, a capacidade de simular e escrever instruções em sequência, com decisões ou repetições.

3. Atividades Educacionais no Minecraft

Minecraft é um jogo eletrônico de sobrevivência, onde o jogador tem como objetivo explorar mundos tridimensionais formados por blocos, sendo capaz de realizar construções, interagir com outros jogadores, extrair matérias-primas, combater inimigos, entre outros. O jogo é composto por três modos: sobrevivência, criativo e aventura. Neste trabalho usa-se o modo criativo, o qual permite aos jogadores facilmente criar e destruir estruturas permitindo o uso infinito de blocos. Também neste modo, está disponível ao jogador o inventário, no qual pode-se remover ou adicionar qualquer item.

O Minecraft Educacional foi lançado em 2006, sendo praticamente a mesma ferramenta conhecida mundialmente, porém o ambiente foi adaptado para ser utilizado em

salas de aula. Nesta nova versão, cada servidor do jogo tem capacidade para 30 usuários além do admistrador. Dentre as principais diferenças para o Minecraft tradicional estão: a inclusão de lousas, onde o aluno pode encontrar orientações das atividades a serem realizadas ou registrar soluções para tais atividades; a criação de personagens não programáveis que podem servir de guias pelas atividades; a possibilidade de armazenar fotos das construções em portfólios dos alunos; as coordenadas nos mundos que facilitam combinar pontos de encontro; a plataforma multijogador que permite os alunos colaborarem em suas construções; e a adição de um novo personagem (além do Avatar do jogador), denominado Agente, que é um robô que pode ser programado para realizar ações e interagir com o Avatar.

Fazendo uso desta plataforma, foi proposto um conjunto de atividades para desenvolver diferentes habilidades relacionadas ao PC para serem realizadas em encontros de aproximadamente uma hora. Algumas dessas atividades foram criadas e outras adaptadas para serem implementadas no Minecraft. São disponibilizados planos de aula para as atividades, bem como materiais de apoio, nos quais o professor pode se guiar para a realização dos encontros. Esse material é acompanhado dos "mundos" que serão utilizados nas atividades: o Mundo das Frações, o dos Objetos 2D, o dos Objetos 3D, o dos Autômatos e o de Treinamento. Além das cinco atividades apresentadas nas subseções a seguir, foi proposta uma atividade de introdução, com o objetivo de apresentar os comandos e funcionalidades do Minecraft, bem como de introduzir os mundos. Mais especificamente, no Mundo de Treinamento foram apresentados comandos básicos como: movimentar para trás e para frente, correr, pular, entre outros.

3.1. Atividade I: Frações

Objetivo: introduzir o conceito e a representação de frações.

Pré-Requisito: saber ler, escrever e contar.

Descrição: a atividade é realizada no Mundo das Frações, tendo o Avatar como personagem principal. Esta atividade deve ser interativa, onde o professor deve introduzir o conceito e a representação de frações na plataforma. Para tal, são usados conjuntos de cubos de mesmo tamanho (já disponíveis no Mundo das Frações) dispostos de modo a formar um único bloco, representando assim uma figura dividida em diversas partes iguais. Dentre estes cubos, existem alguns brancos e outros coloridos. O conjunto de cubos coloridos representa uma fração de um bloco maior. Um exemplo de bloco representando a fração $1/4$ é ilustrado na imagem à esquerda da Figura 1, onde tem-se 4 cubos agrupados, sendo 3 brancos e 1 azul. Após a identificação da fração no bloco, os alunos devem representá-la na mini lousa, escrevendo-a sob a forma n/d. O professor deve guiar os alunos de modo que eles consigam concluir que, para qualquer fração apresentada, o número total de cubos determina o denominador (d) da fração e o número de cubos coloridos determina o numerador (n).

3.2. Atividade II: Frações Equivalentes

Objetivo: introduzir o conceito de fração equivalente.

Pré-Requisito: saber ler, escrever e conhecer a representação de frações.

Descrição: a atividade é realizada no Mundo das Frações, tendo o Avatar como personagem principal. Esta também é uma tarefa interativa, em que o professor começa expli-

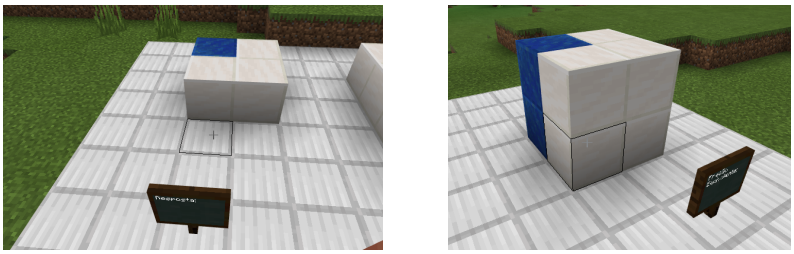


Figura 1. Representação de frações no Mundo das Frações: fração $1/4$ (à esquerda) e fração $2/8$ (à direita).

cando o conceito de fração equivalente (obtida por meio de múltiplos da fração original). Nesta tarefa, o aluno, tendo acesso às soluções das tarefas da Atividade I, deve construir frações equivalentes e indicar suas soluções nas mini lousas. A construção de uma fração equivalente deve ser realizada adicionando um bloco de mesmo tipo e cor sobre cada um dos blocos originais. A quantidade de blocos adicionados pode variar, gerando diferentes frações equivalentes. Na imagem à direita da Figura 1, tem-se a representação da fração $2/8$, equivalente àquela apresentada na imagem à esquerda da mesma figura.

3.3. Atividade III: Objetos 2D

Objetivo: introduzir a noção de instruções e algoritmos, bem como o conceito de coordenadas dadas por linhas e colunas.

Pré-Requisito: saber ler, escrever e contar.

Descrição: a atividade é realizada no Mundo dos Objetos 2D, tendo o Avatar como personagem principal. Inicialmente, o conceito de coordenada (com linhas e colunas) é apresentado, a partir de um tabuleiro presente no mundo (imagem à esquerda da Figura 2). Este tabuleiro representa uma matriz 21×21, onde os índices das linhas são dados por números (de 1 à 21) e os das colunas por letras (de A a U). Os índices estão indicados em mini lousas distribuídas pelas bordas do tabuleiro. Para exercitar esse conceito de coordenada, o professor pode solicitar que os alunos posicionem seu Avatar sobre determinadas coordenadas, façam uma foto da cena e a salvem em seu portifólio, nomeando-a com as coordenadas do Avatar no tabuleiro. Depois disso, os alunos devem seguir a sequência de instruções localizadas na lousa deste mundo. Essas instruções levam à representação de uma imagem no tabuleiro. Cada instrução corresponde a colocação de blocos de uma determinada cor em determinadas coordenadas. Por exemplo, dentre as instruções dadas para representar a imagem ilustrada à direita da Figura 2, a instrução "Coloque blocos pretos nas posições: $(15, J)$ e $(15, L)$", faz com que os olhos do caranguejo sejam desenhados. No mundo correspondente à avaliação é solicitado que os estudantes criem seus próprios desenhos no tabuleiro e escrevam as instruções para a sua construção.

3.4. Atividade IV: Objetos 3D

Objetivo: trabalhar os conceitos de instruções e a algoritmos para construir objetos 3D, introduzindo noções de profundidade.

Pré-Requisito: saber ler, escrever e contar.

Descrição: a atividade é realizada no Mundo dos Objetos 3D, tendo o Avatar como personagem principal. Inicialmente, nesta atividade, os alunos devem seguir instruções da

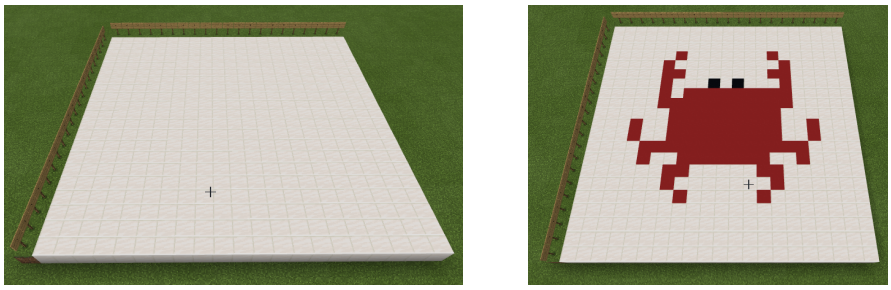


Figura 2. Tabuleiro (à esquerda) e imagem de um caranguejo (à direita) obtida após a execução das instruções do Mundo dos Objetos 2D.

lousa do Mundo dos Objetos 3D, as quais levarão à construção de um monitor de computador flat, conforme ilustrado à esquerda na Figura 3. Após a construção do objeto no Minecraft, o professor apresentará a imagem do monitor, ilustrada à direita na Figura 3 e iniciará uma discussão sobre quais as diferenças e semelhanças entre o objeto construído e a imagem mostrada. Os alunos devem conseguir perceber que as bordas e partes arredondadas, bem como a "marca" do monitor não foram representados. O professor deve destacar que, "esquecendo-se" de algumas características do monitor da imagem apresentada, pode-se dizer que se trata do mesmo objeto criado no mundo do Minecraft. Com base nestas discussões, os alunos devem construir, em duplas, outros objetos no Mundo dos Objetos 3D, mas agora sem instruções. Alguns exemplos de objetos que podem ser construídos estão ilustrados na Figura 4. Inicialmente, os alunos devem decidir quais as características que devem e podem ser representadas no mundo e, após, construir o objeto, relatando e justificando, ao final, as decisões tomadas.

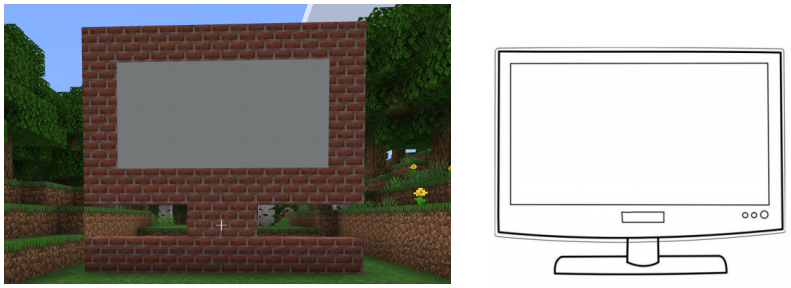


Figura 3. Objeto construído no Mundo do Objetos 3D (à esquerda) e imagem do monitor que deu origem ao objeto construído (à direita).

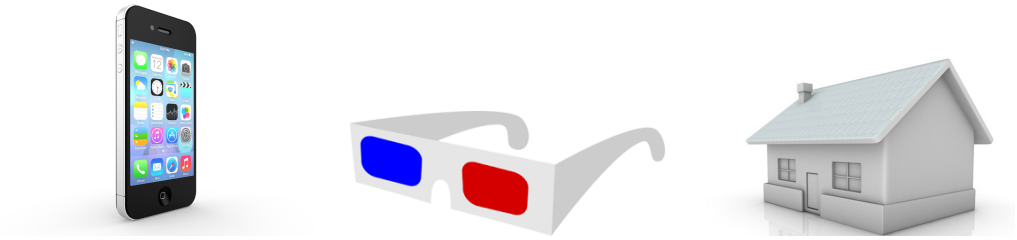


Figura 4. Objetos para serem construídos no Mundo dos Objetos 3D.

3.5. Atividade V: Autômatos

Objetivo: introduzir a noção de autômatos finitos.

Pré-Requisito: saber ler, escrever e ter noções de localização (direita, esquerda, frente e atrás).

Descrição: a atividade é realizada no Mundo dos Autômatos, tendo o Avatar como personagem principal. Essa atividade foi adaptada de [BRACKMANN 2017] e, aqui, o aluno deve percorrer circuitos de autômatos, seguindo as restrições indicadas em uma lousa. Um exemplo de circuito a ser percorrido está ilustrado na Figura 5. Blocos brancos representam os estados, blocos coloridos as transições e mini lousas indicam os estados iniciais e finais de cada autômato. A restrição indicada para este circuito é a de sair da posição inicial e chegar na final percorrendo apenas as cores vermelho e azul. O Avatar deve percorrer o autômato jogando sementes pelo caminho, de forma a deixar registrado o percurso realizado. Essas sementes vão estar disponíveis no inventário. Após realização do percurso o aluno deve descrever a sequência cores das arestas percorridas. Em conjunto, os alunos devem identificar todos os possíveis percursos para cada circuito, levando em conta as restrições estabelecidas e identificando as cores presentes nestes caminhos.

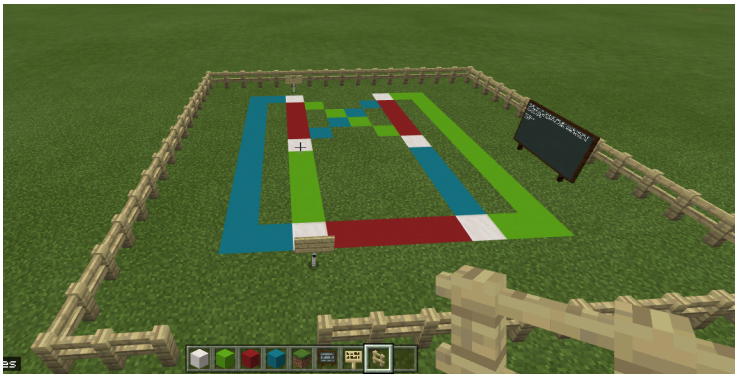


Figura 5. Exemplo de circuito do Mundo dos Autômatos.

4. Pensamento Computacional nas Atividades Propostas

As atividades propostas neste trabalho abordam diferentes conceitos do PC, sendo que, de forma mais contundente, podemos destacar a Abstração, a Decomposição, o Reconhecimento de Padrões e o Pensamento Algorítimico. A Tabela 1 sumariza a relação dos conceitos abordados por atividade.

Na Atividade I, a decomposição é usada ao identificar as partes (cubos) que compõem o todo (bloco), bem como ao representar a fração algebricamente, visto que o problema de identificar o denominador e o numerador são tratados de forma separada. A abstração nesta atividade está refletida na identificação do denominador, onde todos os cubos, independentes de cor, devem ser considerados para este cálculo. Por fim, o reconhecimento de padrões é abordado quando os alunos identificam nos vários exemplos a similaridade na forma de identificar e representar as frações, isto é, o denominador é obtido contando-se a quantidade total de cubos (independente de cor) e o numerador é obtido contando-se a quantidade de cubos coloridos. A Atividade II é desenvolvida seguindo as mesmas técnicas e conceitos da primeira atividade, além de incluir também a

Tabela 1. Habilidades e conceitos do PC abordados em cada atividade proposta.

	Ativ. I	Ativ. II	Ativ. III	Ativ. IV	Ativ. V
Abstração	X	X	X	X	X
Decomposição	X	X			
Reconhecimento de Padrões	X	X			X
Pensamento Algorítimico			X	X	X

identificação do padrão do procedimento para se obter as frações equivalentes, isto é, multiplicando de forma proporcional a quantidade de cubos coloridos e brancos dos blocos, a qual representa a multiplicação do numerador e denominador na representação algébrica.

Ambas as Atividades III e IV trabalham fortemente o conceito de abstração ao identificar quais as características podem ser descritas nas representações 2D e 3D dos objetos, respectivamente. Além disso, a simulação e construção de algoritmos são realizadas, considerando diferentes conjuntos de instruções: na primeira atividade envolviam coordenadas de uma matriz e na segunda incluem instruções que consideram as 3 dimensões dos objetos.

Finalmente, na Atividade V, a abstração está presente na descrição dos caminhos como sequência de cores percorridas. O pensamento algorítmico também é desenvolvido nesta atividade na medida em que o Avatar deve largar sementes ao se deslocar pelos circuitos, simulando assim um algoritmo definido para atingir seu objetivo. Esse algoritmo deve incluir testes da existência do caminho que satisfaça as restrições impostas. O reconhecimento de padrão é trabalhado no momento em que os alunos identificam as cores dos caminhos que são válidos em um determinado circuito.

5. Conclusão

Este trabalho teve como objetivo apresentar uma proposta de atividades que visam ensinar e desenvoler alguns conceitos e habilidades do PC de forma atrativa e divertida, fazendo uso da plataforma de jogos do Minecraft Educacional. As atividades aqui apresentadas buscam trabalhar os conceitos do PC de forma intencional, permitindo que o professor observe o quanto os alunos conseguem compreendê-los e utilizá-los nas tarefas desenvolvidas. Como trabalhos futuros, pretende-se estender estas atividades, considerando outros conceitos e habilidades do PC, bem como validar esta proposta por meio de experimentos com estudantes.

Referências

BRACKMANN, C. P. (2017). *Desenvolvimento do pensamento computacional através de atividades desplugadas na educação básica*. PhD thesis, UFRGS, Porto Alegre.

COSTA, S. S. e. a. (2015). Um estudo exploratório dos games para introdução ao pensamento computacional. In *Anais do 7o CONAHPA - Congresso Nacional de Ambientes Hipermídia para Aprendizagem*.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., and Woollard, J. (2015). *Computational thinking–A guide for teachers*. Computing At School. Disponível em https://community.computingatschool.org.uk/resources/2324/single. Acesso em: julho de 2021.

Curzon, P., Dorling, M., Ng, T., Selby, C., and Woollard, J. (2014). Developing computational thinking in the classroom: a framework. Project report, Computing at School.

Denning, P. J. and Tedre, M. (2019). *Computational Thinking*. MIT Press.

Futschek, G. (2006). Algorithmic thinking: The key for understanding computer science. In Mittermeir, R. T., editor, *Informatics Education – The Bridge between Using and Understanding Computers*, pages 159–168, Berlin, Heidelberg. Springer Berlin Heidelberg.

ISTE and CSTA (2011). Computational thinking in K-12 education: leadership toolkit. Disponivel em https://cdn.iste.org/www-root/2020-10/ISTE_CT_Leadership_Toolkit_booklet.pdf. Acesso em: julho de 2021.

Kalelioglu, F., Gulbahar, Y., and Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic J. Modern Computing*, 4(3):583–596.

KNITTEL, T., SANTANA, L., PEREIRA, M., and MENUZZI, M. (2017). Minecraft: experiências de uso dentro e fora da sala de aula. In *In: SIMPÓSIO BRASILEIRO DE JOGOS E ENTRETENIMENTO DIGITAL*, page 789–795.

Microsoft (2006). Teaching and learning activities. Disponível em https://education.minecraft.net/pt-pt/resources. Acesso em: julho de 2021.

Microsoft (2021). Minecraft education edition. Disponível em https://education.minecraft.net/. Acesso em: julho de 2021.

Ribeiro, L., Foss, L., and da Costa Cavalheiro, S. A. (2017). Entendendo o pensamento computacional. Disponível em https://arxiv.org/pdf/1707.00338.pdf. Acesso em: julho de 2021.

Royal Society (2017). After the reboot: computing education in uk schools. Disponível em https://royalsociety.org/~/media/policy/projects/computing-education/computing-education-report.pdf. Acesso em: julho de 2021.

Selby, C. and Woollard, J. (2013). Computational thinking: the developing definition. Project report, University of Southampton.

Urzêda, R., Severiano, E., and Amorim, L. (2020). O uso do scratch no curso de pedagogia: relato de uma experiência interdisciplinar. In *Anais do XXVIII Workshop sobre Educação em Computação*, pages 21–25, Porto Alegre, RS, Brasil. SBC.

Wing, J. (2006). Computational thinking. *Communications of the ACM*, 49:33–35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A*, 366(1881):3717–3725.

Representabilidade de Operadores via Ordens Admissíveis

Lidiane da Silva [1], Guilherme Schneider [1], Adenauer Yamin [1], Helida Santos [2], Benjamin Bedregal [3], Renata Reiser [1]

[1]PPGC/CDTEC/UFPEL, Pelotas – RS – Brasil
[2]C3/FURG, Rio Grande – RS – Brasil
[3]DIMAP/UFRN, Natal – RN – Brasil

{lcdsilva,gbschneider,adenauer,reiser}@inf.ufpel.edu.br
helida@furg.br
bedregal@dimap.ufrn.br

Abstract. *Este trabalho apresenta um estudo sobre a representabilidade de conectivos fuzzy valorados intervalarmente considerando ordens parciais e admissíveis. Nossa abordagem considera aquelas ordens baseadas em funções de agregação injetivas que permitem a construção de operadores fuzzy valorados intervalarmente. Um resultado imediato é a construção da implicação usada na Lógica Quântica, conhecida como QL-implicação, que em nossa proposta é generalizada para abordagem intervalar, via classe de ordem admissíveis.*
Keywords: *Operadores fuzzy valorados intervalarmente, Lógica fuzzy valorada intervalarmente, QL-implicações, ordens admissíveis.*

1. Introdução

A lógica fuzzy valorada intervalarmente (IvFL) [Zadeh 1975], vista no sentido estrito, tem a capacidade de lidar com a incerteza e também agrega precisão na representação dos graus de pertinência que definem os conjuntos fuzzy.

A análise de conectivos representáveis na IvFL se fundamenta no princípio de extensão e representabilidade dos conjuntos fuzzy [Zadeh 1965]. Sendo assim, uma ordem parcial, considera as representações canônica, intervalar e ainda a pseudo-representação, na interpretação dos graus de pertinência de uma implicação fuzzy relacionada à regra condicional em sistemas de inferência baseados em lógica fuzzy, modelando o (a falta de) conhecimento sobre o valor de uma variável. Nesse sentido, a melhor representação de um conectivo fuzzy, resulta na correta interpretação de uma variável que agrega tanto a incerteza como a imprecisão nos possíveis conjuntos considerados no sistema de inferência fuzzy [Hickey et al. 2001].

Para atingir nossos objetivos, focamos nosso estudo na representabilidade de vários operadores fuzzy, considerando ordens parciais e introduzindo uma classes de ordens admissíveis. Nesse contexto, estudamos algumas funções intervalares como t-(co)normas, negações fuzzy e implicações, incluindo a classe de QL-implicações, que foi estendida para abordagem intervalar considerando as classes de ordens admissíveis.

Este artigo está organizado da seguinte forma: As Seções 2 e 3 estudamos as possíveis formas de ordenação (parcial e total) dos conjuntos fuzzy valorados intervalarmente e a representabilidade dos conectivos pertencentes a estes conjuntos. A proposta de uma nova forma de representação com respeito a ordens admissíveis, juntamente com

a nova família de ordens admissíveis é apresentada na Seção 4. Nossas conclusões estão na Seção 5.

2. Ordens Parciais e Admissíveis em $L([0,1])$

Este estudo considera as principais classes de conectivos fuzzy valorados intervalarmente, analisando as propriedades relacionadas à representabilidade.

Consideramos $L([0,1]) = \{[x,y] \mid 0 \leq x \leq y \leq 1\}$ como a família de todos os valores fuzzy intervalares. As projeções $l, r : L([0,1]) \to [0,1]$ são definidas respectivamente, por $l([x_1,x_2]) = x_1$ e $r([x_1,x_2]) = x_2$. Para $X \in L([0,1])$, $l(X)$ e $r(X)$ podem também ser denotadas respectivamente, por $\underline{X}$ e $\overline{X}$. E, seja $D(L([0,1])) = \{[x,y] \in L([0,1]) | x = y\}$ o conjunto de todos os intervalos degenerados, denotando um elemento degenerado como $[x,x] = \mathbf{x}$, para $x \in [0,1]$. Sejam as ordens parciais em $L([0,1])$:
O1 *Ordem de Inclusão*: $X \subseteq Y$ sss $\underline{X} \geqslant \underline{Y}$ e $\overline{X} \leqslant \overline{Y}$;
O2 *Ordem Produto (Usual)*: $X \leq Y$ sss $\underline{X} \leqslant \underline{Y}$ e $\overline{X} \leqslant \overline{Y}$.
Os intervalos degenerados $[0,0] = \mathbf{0}, [1,1] = \mathbf{1} \in \mathbb{D}(L([0,1]))$ são, respectivamente, o menor e o maior elementos em $(L([0,1], \leq)$.

A seguir, tem-se as ordens admissíveis [Bustince et al. 2013, Santana et al. 2020].

Definição 1. *[Bustince et al. 2013] Uma ordem $\preceq_{L([0,1])}$ é admissível em $L([0,1])$ se:*
(i) $\preceq_{L([0,1])}$ é uma ordem linear em $L([0,1])$, e
(ii) $\preceq_{L([0,1])}$ refina uma ordem parcial $\leq$, i.e., $\forall X_1, X_2 \in L([0,1])$, $X_1 \leq X_2 \Rightarrow X_1 \preceq_{L([0,1])} X_2$.

Os intervalos degenerados $\mathbf{0}$ e $\mathbf{1}$ também são respectivamente, o menor e o maior elementos para qualquer intervalo $\langle L([0,1]), \preceq_{L([0,1])}\rangle$.
Exemplo 1. *Principais ordens admissíveis consideradas neste estudo:*

(i) $X \preceq_{Lex1} Y \Leftrightarrow \underline{X} < \underline{Y} \vee (\underline{X} = \underline{Y} \wedge \overline{X} \leq \overline{Y})$;
(ii) $X \preceq_{Lex2} Y \Leftrightarrow \overline{X} < \overline{Y} \vee (\overline{X} = \overline{Y} \wedge \underline{X} \leq \underline{Y})$;
(iii) $X \preceq_{XY} Y \Leftrightarrow \underline{X} + \overline{X} < \underline{Y} + \overline{Y} \vee (\underline{X} + \overline{X} = \underline{Y} + \overline{Y} \wedge \overline{X} - \underline{X} \leq \overline{Y} - \underline{Y})$

E, a classe de ordens admissíveis em $\langle L([0,1]), \preceq_A\rangle$ via funções injetivas A.
Teorema 2. *Seja $A\colon L([0,1]) \to [0,1]$ uma função crescente com respeito a ordem parcial $\leq$, $A(\mathbf{0}) = 0$ e $A(\mathbf{1}) = 1$. A relação $\preceq_A$ em $L([0,1])$ definida por*

$$X \preceq_A Y \Leftrightarrow \begin{cases} X = Y, \text{ ou} \\ A(X) < A(Y), \end{cases} \tag{1}$$

é uma ordem admissível em $L([0,1])$, quando A for injetiva.

Prova. A relação $\preceq_A$ é reflexiva e antissimétrica. Seja $X, Y, Z \in L([0,1])$ tal que $X \preceq_A Y$ e $Y \preceq_A Z$. Então, consideramos os seguintes casos: (i) Se $X = Y$ ou $Y = Z$, temos que $X \preceq_A Z$; (ii) Se $X \neq Y$ e $Y \neq Z$ então $A(X) < A(Y)$ e $A(Y) < A(Z)$ e portanto $A(X) < A(Z)$. Logo, $X \preceq_A Z$. (iii) Se $X \neq Y$ e $Y = Z$ então $A(X) < A(Y)$ e $A(Y) = A(Z)$, significa que $A(X) < A(Z)$. Assim, $X \preceq_A Z$. (iv) Se $X = Y$ e $Y \neq Z$, $A(X) = A(Y)$ e $A(Y) < A(Z)$, então $A(X) < A(Z)$. Assim, $X \preceq_A Z$. Com base nos casos vistos, tem-se uma relação transitiva $\langle L([0,1]), \preceq_A\rangle$. Para cada $X, Y \in L([0;1])$, vemos que: (i) No caso $A(X) < A(Y)$, isso implica que $X \preceq_A Y$; (ii) No caso $A(Y) < A(X)$, implica que $Y \preceq_A X$; e ainda, (iii) Se $A(X) = A(Y)$, desde que

A seja uma função injetiva, resulta em $X = Y$. Então, também temos uma relação total $\langle L([0,1]), \preceq_A \rangle$. Portanto, temos uma ordem linear $\langle L([0,1]), \preceq_A \rangle$. Demonstrando que $\preceq_A$ é uma ordem linear. Quando temos $X, Y \in L([0,1])$, tal que $X \leq Y$. Se $X = Y$ então $X \preceq_A Y$. Para a $\leq$-ordem parcial, $X < Y$ implica $A(X) < A(Y)$, desde que A seja injetiva e crescente. Portanto, $X \preceq_A Y$ e a ordem $\preceq_A$ refina a ordem usual $\leq$. Logo, a $\preceq_A$-ordem é admissível em $L([0,1])$. □

A seguir, definimos a ordem $\preceq_{\mathbf{A}}$, baseada na função $\mathbf{A}$ obtida pela expansão decimal infinita dos extremos de um intervalo $X = \left[\underline{X}, \overline{X}\right] \subseteq [0,1]$, indicado como: $\left[\underline{X}, \overline{X}\right] = \left[0.\underline{X}^{[1]}\underline{X}^{[2]}\ldots,\ 0.\overline{X}^{[1]}\overline{X}^{[2]}\ldots\right]$. Neste contexto, $1{=}0.99\ldots$ e $0 = 0.00\ldots$.

Definição 3. *A função* $\mathbf{A}\colon L([0,1]) \to [0,1]$ *dada por*

$$\mathbf{A}(X) = \begin{cases} 0.\underline{X}^{[1]}\overline{X}^{[1]}\underline{X}^{[2]}\overline{X}^{[2]}\ldots, \text{ if } 0 \leq \underline{X} \leq \overline{X} < 1; \\ 0.\underline{X}^{[1]}9\underline{X}^{[2]}9\ldots, \text{ se } 0 \leq \underline{X} \leq \overline{X} = 1; \\ 1, \text{ se } X = \mathbf{1}; \end{cases} \tag{2}$$

é uma função crescente com respeito a ordem parcial usual $\leq$ *em* $L([0,1])$ *que satisfaz as condições de contorno* $\mathbf{A}(\mathbf{0}) = 0$ *e* $\mathbf{A}(\mathbf{1}) = 1$.

Proposição 4. *A função* $\mathbf{A}\colon L([0,1]) \to [0,1]$ *dada pela Eq. (2) é uma função injetiva tal que* $X \leq Y \Rightarrow \mathbf{A}(X) \leq \mathbf{A}(Y)$*, pela ordem usual* $\leq$.

Prova. Prova direta a partir dos resultados de [Santana et al. 2020]. □

Corolario 1. *A ordem* $\preceq_{\mathbf{A}}$ *dada por:*

$$X \preceq_{\mathbf{A}} Y \Leftrightarrow \begin{cases} X = Y, \text{ or} \\ \mathbf{A}(X) < \mathbf{A}(Y), \end{cases} \tag{3}$$

é uma ordem admissível $\preceq_{\mathbf{A}}$ *com respeito a ordem parcial usual* $\leq$*-ordem em* $L([0,1])$.

Prova. Diretamente a partir do Teorema 2 e Prop. 4. □

Exemplo 2. *Seja* $X = [0.15, 0.88], Y = [0.16, 0.86] \in L([0,1])$. *Temos que:*
(i) $X \preceq_{Lex1} Y$, $Y \preceq_{Lex2} X$ *e* $Y \preceq_{XY} X$*;*
(ii) $\mathbf{A}(X) = 0.1858 < 0.1866 = \mathbf{A}(Y)$*, então* $X \preceq_{\mathbf{A}} Y$.

Definição 5. *Seja* $\mathbf{A}\colon L([0,1]) \to [0,1]$ *dada na Def. 3. Para* $x = 0.x^{[1]}x^{[2]}\cdots = (0.x^{[i]})_{i\in\mathbb{N}_+} \in [0,1]$, $\mathbb{N}_+ = \{1, 2, \ldots\}$*, a função* $\mathbf{A}^{(-1)}\colon [0,1] \to L([0,1])$ *é dada por:*

$$\mathbf{A}^{(-1)}(x) = \begin{cases} \mathbf{1} \text{ se } x = 1; \\ [(0.x^{[2i-1]})_{i\in\mathbb{N}_+}, (0.x^{[2i]})_{i\in\mathbb{N}_+}] \text{ se} \\ \quad (0.x^{[2i-1]})_{i\in\mathbb{N}_+} \leq (0.x^{[2i]})_{i\in\mathbb{N}_+}; \text{ e} \\ \inf\{X \in L([0,1]) : A(X) \geq x\} = [x, x], \text{ c.c.} \end{cases} \tag{4}$$

Lema 1. *A função* $\mathbf{A}^{(-1)}\colon [0,1] \to L([0,1])$ *definida pela Eq. (4) verifica as condições:*

1. *para cada* $X \in L([0,1])$*, tal que* $\mathbf{A}(X) \neq 1$, $\mathbf{A}^{(-1)}(\mathbf{A}(X)) = X$*;*
2. *para cada* $x \in [0,1]$, $\mathbf{A}(\mathbf{A}^{(-1)}(x)) \leq x$*;*
3. *se* $x, y \in [0,1]$ *e* $x \leq y$, $\mathbf{A}^{(-1)}(x) \preceq_{\mathbf{A}} \mathbf{A}^{(-1)}(y)$*;*
4. *para cada* $x \in [0,1]$, $\mathbf{A}^{(-1)}(x) = \mathbf{0} \Leftrightarrow x = 0$ *e* $\mathbf{A}^{(-1)}(x) = \mathbf{1} \Leftrightarrow x = 1$.

Prova. Prova direta. □

3. Representabilidade em $\langle L([0,1]), \leq \rangle$

Seja $\langle L([0,1])), \leq \rangle$ o conjunto de todos os valores fuzzy intervalares. Em [Deschrijver 2008], um intervalo $X \in L([0,1])$ é a representação intervalar de um número real α sempre que $\alpha \in X$. Considerando dois intervalos representáveis X e Y de um número real α, X é dito uma melhor representação de α que Y se $X \subseteq Y$.

Definição 6. *[Santiago et al. 2006] Uma função* $F\colon L([0,1])^n \to L([0,1])$ *é a representação intervalar de uma função* $f\colon [0,1]^n \to [0,1]$ *se, para cada* $\vec{X} = (X_1, \ldots, X_n) \in L([0,1])^n$ *e* $\vec{x} \in \vec{X}$, $f(\vec{x}) \in F(\vec{X})$.

Além disso, uma função intervalar $F\colon L([0,1])^n \to L([0,1])$ é uma representação intervalar melhor de $f\colon L([0,1])^n \to L([0,1])$ que $G\colon L([0,1])^n \to L([0,1])$, denotada por $G \sqsubseteq F$, se para cada $\vec{X} \in L([0,1])^n$, a inclusão $F(\vec{X}) \subseteq G(\vec{X})$. Neste contexto, a melhor representação intervalar (representação canônica) de uma função real $f\colon [0,1]^n \to [0,1]$, é a função $\widehat{f}\colon L([0,1])^n \to L([0,1])$ definida por

$$\widehat{f}(\vec{X}) = [\inf\{f(\vec{x}) | \vec{x} \in \vec{X}\}, \sup\{f(\vec{x}) | \vec{x} \in \vec{X}\}]. \tag{5}$$

E, quando f é contínua no sentido usual, $\widehat{f}(\vec{X}) = \{f(\vec{x}) | \vec{x} \in \vec{X}\} = f(\vec{X}), \forall \vec{X} \in L([0,1])^n$. Além disso, $\mathbb{F}\colon L([0,1])^n \to L([0,1])$ é chamada de função valorada intervalarmente decomposta, se existem $F_1, F_2 : [0,1]^n \to [0,1]$, tal que

$$\mathbb{F}(X_1, \ldots, X_n) = [F_1(\underline{X}_1, \ldots, \underline{X}_n), F_2(\overline{X}_1, \ldots, \overline{X}_n)], \forall X_1, \ldots, X_n \in L([0,1]).$$

3.1. Conectivos Fuzzy Representáveis $\langle L([0,1]), \leq \rangle$

Um conectivo fuzzy valorado intervalarmente pode ser considerado como um conectivo fuzzy valorado intervalarmente representável [Bedregal and Takahashi 2006]. Essa generalização se ajusta ao princípio fuzzy e, o grau de pertinência com valor de intervalo pode ser considerado como uma aproximação do grau exato. As agregações em $L([0,1])$, importantes operadores nas aplicações de tomada de decisão, são revisadas logo a seguir.

Definição 7. *Uma função* $\mathbb{M}\colon L([0,1])^n \to L([0,1])$ *é uma função de agregação valorada intervalarmente (IvA), relativa a ordem parcial* $\leq$*, se satisfaz as condições:*
$\mathbb{M}1$*:* $X_i \leq Y_i \Rightarrow \mathbb{M}(X_1, ..., X_n) \leq \mathbb{M}(Y_1, ..., Y_n)$, $\forall i \in \mathbb{N}_n$ *e* $\forall X_i, Y_i \in L([0,1])$*;*
$\mathbb{M}2$*:* $\mathbb{M}(\mathbf{0}, ..., \mathbf{0}) = 0$ *e* $\mathbb{M}(\mathbf{1}, ..., \mathbf{1}) = 1$.

Seja $\mathbb{M}\colon L([0,1])^n \to L([0,1])$ uma IvA. $\mathbb{M}$ é $\langle L([0,1]), \leq \rangle$-representável se, existir agregações $M_1, M_2\colon [0,1]^n \to [0,1]$, com $M_1 \leq M_2$ e tal que:

$$\mathbb{M}(X_1, \ldots, X_n) = [M_1(\underline{X}_1, \ldots, \underline{X}_n), M_2(\overline{X}_1, \ldots, \overline{X}_n)], \forall X_1, \ldots, X_n \in L([0,1]). \tag{6}$$

Segue a extensão de t-(co)normas em $L([0,1])$ [Bedregal and Takahashi 2006].

Definição 8. *Uma função* $\mathbb{T}(\mathbb{S})\colon L([0,1])^2 \to L([0,1])$ *é uma* ***t-norma valorada intervalarmente (t-conorma)*** *IvT (IvS), se for uma função comutativa, associativa, monotônica com respeito a ordem produto, e tem* $\mathbf{1}$ *(*$\mathbf{0}$*) como elemento neutro.*

Pela Eq. (6), as funções $\mathbb{T}_{T,T'}(\mathbb{S}_{S,S'})\colon L([0,1])^2 \to L([0,1])$, são chamadas $\langle L([0,1]), \leq\rangle$-representáveis t-(co)norm dadas por

$$\mathbb{T}_{T,T'}(X,Y) = [T(\underline{X},\underline{Y}), T'(\overline{X},\overline{Y})], \mathbb{S}_{S,S'}(X,Y) = [S(\underline{X},\underline{Y}), S'(\overline{X},\overline{Y})], \quad (7)$$

se, e somente se, $T(x,y) \leq T'(x,y)$ ($S(x,y) \leq S'(x,y)$), $\forall x,y \in [0,1]$. Neste caso, temos que $T_{T,T'}([0,1],[0,1]) = [0,1]$ ($S_{S,S'}([0,1],[0,1]) = [0,1]$). E ainda, se $T = T'(S = S')$, aplica-se a notação $\mathbb{T}_T(\mathbb{S}_S)$. Observa-se que, $\mathbb{T}_T = \widehat{T}$ ($\mathbb{S}_S = \widehat{S}$).

Exemplo 3. *Veja ilustrações de t-(co)normas representáveis em* $\langle L([0,1]), \leq\rangle$.

$\mathbb{T}_M(X,Y) = [\min(\underline{X},\underline{Y}), \min(\overline{X},\overline{Y})]$; $\mathbb{S}_M(X,Y) = [\max(\underline{X},\underline{Y}), \max(\overline{X},\overline{Y})]$;
$\mathbb{T}_P(X,Y) = [\underline{XY}, \overline{XY}]$; $\mathbb{S}_P(X,Y) = [\underline{X}+\underline{Y}-\underline{XY}, \overline{X}+\overline{Y}-\overline{XY}]$;
$\mathbb{T}_{LK}(X,Y) = [\max(0, \underline{X}+\underline{Y}-1), \max(0, \overline{X}+\overline{Y}-1)]$; $\mathbb{S}_{LK}(X,Y) = [\min(\underline{X}+\underline{Y}, 1), \min(\overline{X}+\overline{Y}, 1)]$.

Logo, uma IvT(IvS) é representável, sss for monotônica referente a ordem parcial $\subseteq$ [Deschrijver 2008]. Mas, nem todas as t-(co)norms em $L([0,1])$ são t-representáveis.

Sejam $T(S)$ uma t-(co)norm. As funções $\mathbb{T}^\star_S(\mathbb{S}^\star_T) : L([0,1])^2 \to L([0,1])$

$$\begin{aligned} \mathbb{T}^\star_T(X,Y) &= [T(\underline{X},\underline{Y}), \max(T(\underline{X},\overline{Y}), T(\overline{X},\underline{Y}))]; \\ \mathbb{S}^\star_S(X,Y) &= [\min(S(\underline{X},\overline{Y}), S(\overline{X},\underline{Y})), S(\overline{X},\overline{Y})] \end{aligned} \quad (8)$$

são denominadas como pseudo $\langle L([0,1]), \leq\rangle$-representáveis t-(co)norm. E, nesta classe, tem-se satisfeista a condição: $\mathbb{T}^\star_T([0,1],[0,1]) = \mathbf{0}$ ($\mathbb{S}^\star_S([0,1],[0,1]) = \mathbf{1}$).

Exemplo 4. *Ilustram-se t-(co)normas pseudo* $\langle L([0,1]), \leq\rangle$*-representáveis.*

$\mathbb{T}^\star_M(X,Y) = [\min(\underline{X},\underline{Y}), \max(\min(\underline{X},\overline{Y}), \min(\overline{X},\underline{Y}))]$ $\mathbb{S}^\star_M(X,Y) = [\min(\max(\overline{X},\underline{Y}), \max(\underline{X},\overline{Y})), \max(\overline{X},\overline{Y})]$
$\mathbb{T}^\star_P(X,Y) = [\underline{XY}, \max(\underline{X}\overline{Y}, \overline{X}\underline{Y})]$ $\mathbb{S}^\star_P(X,Y) = [\min(\overline{X}+\underline{Y}-\overline{X}\underline{Y}, \underline{X}+\overline{Y}-\underline{X}\overline{Y}), \overline{X}+\overline{Y}-\overline{XY}]$
$\mathbb{T}^\star_{LK}(X,Y) = [\max(0, \underline{X}+\underline{Y}-1), \max(0, \underline{X}+\overline{Y}-1, \overline{X}+\underline{Y}-1)]$ $\mathbb{S}^\star_{LK}(X,Y) = [\min(\overline{X}+\underline{Y}, \underline{X}+\overline{Y}, 1), \min(1, \overline{X}+\overline{Y})]$

Sejam T (S) uma t-(co)norm em $L([0,1])$. Em [Deschrijver 2008], a função $\mathbb{T}'_T(\mathbb{S}'_S)(X,Y)\colon L([0,1])^2 \to L([0,1])$, dada por:

$$\mathbb{T}'_T(X,Y) = [\min(T(\underline{X},\overline{Y}), T(\overline{X},\underline{Y})), T(\overline{X},\overline{Y})]; \quad (9)$$

$$\mathbb{S}'_S(X,Y) = [S(\underline{X},\underline{Y}), \max(S(\underline{X},\overline{Y}), S(\overline{X},\underline{Y}))], \quad (10)$$

não é uma t-representável t-(co)norm, sempre que tem-se $T \neq \min$ ($S \neq \max$).

Exemplo 5. *Apresentamos alguns exemplos na classe de* $\mathbb{T}'_T$ *(*$\mathbb{S}'_S$*).*

$\mathbb{T}'_M(X,Y) = [\min(\min(\underline{X},\overline{Y}), min(\overline{X},\underline{Y})), \min(\overline{X},\overline{Y})]$
$\mathbb{S}'_M(X,Y) = [\max(\underline{X},\underline{Y}), \max(\max(\underline{X},\overline{Y}), \max(\overline{X},\underline{Y}))]$

$\mathbb{T}'_P(X,Y) = [\min(\underline{X}\overline{Y}, \overline{X}\underline{Y}), \overline{XY}]$;
$\mathbb{S}'_P(X,Y) = [\underline{X}+\underline{Y}-\underline{XY}, \max(\underline{X}+\overline{Y}-\underline{X}\overline{Y}, \overline{X}+\underline{Y}-\overline{X}\underline{Y})]$

$\mathbb{T}'_{LK}(X,Y) = [\min(\max(0, \underline{X}+\overline{Y}-1), \max(0, \overline{X}+\underline{Y}-1)), \max(0, \overline{X}+\overline{Y}-1)]$
$\mathbb{S}'_{LK}(X,Y) = [\min(\underline{X}+\underline{Y}, 1), \max(\min(\underline{X}+\overline{Y}, 1), \min(\overline{X}+\underline{Y}, 1))]$

Definição 9. *[Bedregal and Takahashi 2006] Uma função* $\mathbb{N}\colon L([0,1]) \to L([0,1])$ *é uma negação fuzzy valorada intervalarmente (IvN) se,* $\forall X,Y \in L([0,1])$*, temos que:*
$\mathbb{N}1$*:* $\mathbb{N}(\mathbf{0}) = \mathbf{1}$ *e* $\mathbb{N}(\mathbf{1}) = \mathbf{0}$*;*
$\mathbb{N}2$*: Se* $X \geq Y$*, então* $\mathbb{N}(X) \leq \mathbb{N}(Y)$*;*

E, uma IvN $\mathbb{N}$ forte satisfaz a involutividade: $\mathbb{N}3$**:** $\mathbb{N}(\mathbb{N}(X)) = X$, $\forall X \in L([0,1])$.

De acordo com [Bedregal and Takahashi 2006, Deschrijver and Cornelis 2007], podemos construir uma IvN forte $\mathbb{N}$ a partir de uma negação fuzzy forte **N**, que é dada por $\mathbb{N}(X) = [N(\overline{X}), N(\underline{X})]$. Exemplo: $N_C(x) = 1 - x$, $\mathbb{N}_C(X) = [1 - \overline{X}, 1 - \underline{X}]$.

As propriedades mínimas de implicações fuzzy podem ser naturalmente estendidas para a abordagem de intervalar. Conforme [Reiser et al. 2007], uma função $\mathbb{I}\colon L([0,1])^2 \to L([0,1])$ é uma implicação fuzzy valorada intervalarmente (IvI), se satisfaz $\mathbb{I}1$, condições de contorno (lógica clássica), $\mathbb{I}2$ antitonicidade no primeiro argumento e $\mathbb{I}3$ isotonicidade no segundo argumento. É sempre possível obter, a representação canônica de uma IvI a partir de qualquer implicação fuzzy, que também preserve as mesmas propriedades satisfeitas pela implicação fuzzy.

Proposição 10. *[Bedregal et al. 2010] Uma implicação $I\colon [0,1]^2 \to [0,1]$ satisfaz as propriedades* **I1** *e* **I2** *se e somente se $\widehat{I}$ for expressa como: $\widehat{I}(X,Y) = [I(\overline{X},\underline{Y}), I(\underline{X},\overline{Y})]$.*

Uma IvF é uma **QL-implicação valorada intervalarmente** se houver uma IvS $\mathbb{S}$, uma IvN $\mathbb{N}$ e uma IvT $\mathbb{T}$, tal que:

$$\mathbb{I}_{\mathbb{S},\mathbb{N},\mathbb{T}}(X,Y) = \mathbb{S}(\mathbb{N}(X), \mathbb{T}(X,Y)). \tag{11}$$

Teorema 11. *[Reiser et al. 2007, Theo. 4] Seja S uma t-conorma, N uma negação fuzzy e T uma t-norma. Então, temos que: $\mathbb{I}_{\widehat{S},\widehat{N},\widehat{T}} = \widehat{I_{S,N,T}}$.*

Exemplo 6. *Veja a QL-implicação de Łukasiewski $\mathbb{I}_{\mathbb{S}_{LK},\mathbb{N}_C,\mathbb{T}_M}(X,Y) = \mathbb{I}_{LK}(X,Y)$.*

$$\begin{aligned}
\mathbb{I}_{LK}(X,Y) &= \big[\min(1-\overline{X}+min(\underline{X},\underline{Y}),1), min(1-\underline{X}+min(\overline{X},\overline{Y}),1)\big]\\
\mathbb{I}^{\star}_{LK}(X,Y) &= \big[\min\big(\min(1-\overline{X}+\max(\min(\underline{X},\overline{Y}),\min(\overline{X},\underline{Y})),1),\\
&\quad \max\big(\min(1-\overline{X}+min(\underline{X},\underline{Y}),1)\big), \min\big(1-\underline{X}+\max(\min(\underline{X},\overline{Y}),\min(\overline{X},\underline{Y}))\big),1\big)\big]\\
\mathbb{I}'_{LK}(X,Y) &= \big[min\big(1-\overline{X}+\min\big(\min(\underline{X},\overline{Y}),\min(\overline{X},\underline{Y})\big)\big),1\big),\\
&\quad \max\big(\min(1-\overline{X}+\min(\overline{X},\overline{Y}),1), \min(1-\underline{X}+\min(\min(\underline{X},\overline{Y}),\min(\overline{X},\underline{Y})),1)\big)\big]
\end{aligned}$$

Além disso, algumas implicações são exclusivamente QL-implicações (o que significa que não pertencem a outras classes, como implicações S ou implicações R). Como exemplo, consideramos $\mathbb{I}_{\mathbb{S}_{LK},\mathbb{N}_K,\mathbb{T}_P}(X,Y) = \mathbb{I}_{KP}(X,Y)$. Veja o próximo exemplo.

Exemplo 7. *Veja uma ilustração de QL-implicações:*

$$\begin{aligned}
\mathbb{I}_{KP}(X,Y) &= \big[\min(1-\overline{X}^2+\underline{XY},1), \min(1-\underline{X}^2+\overline{XY},1)\big]\\
\mathbb{I}^{\star}_{KP}(X,Y) &= [\min\big(\min(1-\overline{X}^2+\max(\underline{X}\overline{Y},\overline{X}\underline{Y}),1), \min(1-\underline{X}^2+\underline{XY},1)\big),\\
&\quad \min(1-\underline{X}^2+\max(\underline{X}\overline{Y},\overline{X}\underline{Y}),1)]\\
\mathbb{I}'_{KP}(X,Y) &= [\min\big(1-\underline{X}^2+\max(\underline{X}\overline{Y},\overline{X}\underline{Y}),1\big),\\
&\quad \max\big(\min(1-\overline{X}^2+\overline{XY},1), \min(1-\underline{X}^2+\min(\underline{X}\overline{Y},\overline{X},\underline{Y}),1)\big)]
\end{aligned}$$

4. Representabilidade em $\langle L([0,1], \preceq_A\rangle$

O conceito de $\langle L([0,1]), \preceq_A\rangle$-conectivos é apresentado, para uma $\preceq_A$-ordem admissível.

Proposição 12. *Sejam $M\colon [0,1]^n \to [0,1]$ uma agregação estritamente crescente, e A uma função nas condições do Teorema 2. A função $\mathbb{M}^A\colon L([0,1])^n \to L([0,1])$, dada por*

$$\mathbb{M}^A(X_1,\ldots,X_n) = A^{(-1)}\left(M(A(X_1),\ldots,A(X_n))\right) \tag{12}$$

é uma IvA (estritamente crescente) relativa à $\preceq_A$-ordem.

Prova. Diretamente, as condições de contorno são alcançadas: $\mathbb{M}^A(\mathbf{0},\ldots,\mathbf{0}) = A^{(-1)}(M(A(\mathbf{0}),\ldots,A(\mathbf{0}))) = A^{(-1)}(M(0,\ldots,0)) = \mathbf{0}$. E, $\mathbb{M}^A(\mathbf{1},\ldots,\mathbf{1}) = A_1^{(-1)}(M(A(\mathbf{1}),\ldots,A(\mathbf{1}))) = A_1^{(-1)}(M(1,\ldots,1)) = \mathbf{1}$. Seja $X_1,\ldots,X_n,Y \in L([0,1])$, tal que $X_i \preceq_A Y$, para $i \in \mathbb{N}_n$. Logo, $A(X_i) \leq A(Y)$ e porque M é estritamente crescente, $M(A(X_1),\ldots,A(X_n)) \leq M(A(X_1),\ldots,A(X_{i-1}),A(Y),A(X_{i+1}),\ldots,A(X_n))$, e pelo Lema 1, $\mathbb{M}^A(X_1,\ldots,X_n) \preceq_A \mathbb{M}^A(X_1,\ldots,X_{i-1},Y,X_{i+1},\ldots,X_n)$. Além disso, se A é injetiva e M é estritamente crescente, então $\mathbb{M}^A$ é também estritamente crescente. Portanto, a Proposição 12 é verificada. □

Teorema 13. *Seja* $N\colon [0,1] \to [0,1]$ *uma negação fuzzy estritamente decrescente,* $A\colon L([0,1]) \to [0,1]$*, A e $A^{(-1)}$ sejam funções que verificam as condições do Teorema 2, conforme Lema 1. A* $\langle L([0,1]), \preceq_A\rangle$*-negação,* $\mathbb{N}^A\colon L([0,1]) \to L([0,1])$*, é definida por*

$$\mathbb{N}^A(X) = A^{(-1)}(N(A(X))). \tag{13}$$

Prova. Trivialmente, $\mathbb{N}^{\mathbf{A}}(\mathbf{0}) = \mathbf{1}$ e $\mathbb{N}^{\mathbf{A}}(\mathbf{1}) = \mathbf{0}$. Quando $X \preceq_{\mathbf{A}} Y$, desde que N seja estritamente decrescente, pelo Lema 1, $A(X) < A(Y) \Leftrightarrow N(A(Y)) < N(A(X))) \Leftrightarrow A^{(-1)}(N(A(Y))) < A^{(-1)}(N(A(X))) \Leftrightarrow \mathbb{N}^A(Y) \preceq_A \mathbb{N}^A(X)$. □

Exemplo 8. *Ilustramos a* $\langle L([0,1]), \preceq_{\mathbf{A}}\rangle$*-representabilidade de t-(co)normas:*

$\mathbb{T}^{\mathbf{A}}_M(X,Y)=\mathbf{A}^{(-1)}(\min(\mathbf{A}(X),\mathbf{A}(Y)))$ $\quad$ $\mathbb{S}^{\mathbf{A}}_M(X,Y)=\mathbf{A}^{(-1)}(\max(\mathbf{A}(X),\mathbf{A}(Y)))$
$\mathbb{T}^{\mathbf{A}}_P(X,Y)=\mathbf{A}^{(-1)}(\mathbf{A}(X)\cdot\mathbf{A}(Y))$ $\quad$ $\mathbb{S}^{\mathbf{A}}_P(X,Y)=\mathbf{A}^{(-1)}(\mathbf{A}(X)+\mathbf{A}(Y)-\mathbf{A}(X)\cdot\mathbf{A}(Y))$
$\mathbb{T}^{\mathbf{A}}_{LK}(X,Y)=\mathbf{A}^{(-1)}(\max(0,\mathbf{A}(X)+\mathbf{A}(Y)-1))$ $\quad$ $\mathbb{S}^{\mathbf{A}}_{LK}(X,Y)=Aa^{(-1)}(\min(\mathbf{A}(X)+\mathbf{A}(Y),1))$

Uma $\langle L([0,1])), \preceq_{\mathbf{A}}\rangle$*-negação* $\mathbb{N}\colon L([0,1]) \to L([0,1])$ *é uma função que verifica* $(\mathbb{N}1)$ *e a propriedade:* $(\mathbb{N}_{\mathbf{A}}b)$ *: Se* $X \preceq_{\mathbf{A}} Y$*,* $\mathbb{N}(Y) \preceq_{\mathbf{A}} \mathbb{N}(X)$*,* $\forall X, Y \in L([0,1])$*.*

Corolario 2. *Sempre que* $N\colon [0,1] \to [0,1]$ *for uma negação fuzzy forte,* $\mathbb{N}^A$ *verifica a propriedade* $\mathbb{N}^A(\mathbb{N}^A(X)) \succeq_A X$*.*

Prova. Através do Lema 1, nós temos que $\mathbb{N}^A(\mathbb{N}^A(X)) \succeq_A A^{(-1)}(N^2(A(X)))$. Portanto, $\mathbb{N}^A(\mathbb{N}^A(X)) \succeq_A X$. □

Exemplo 9. *A* $\langle L([0,1]), \preceq_{\mathbf{A}}\rangle$*- negação, gerada por uma negação padrão* N_C *é dada por* ${\mathbb{N}_C}^{\mathbf{A}}(X) = \mathbf{A}^{(-1)}(N_C(\mathbf{A}(X))), \forall X \in L([0,1])$*. Através do Exemplo 2,* ${\mathbb{N}_C}^{\mathbf{A}}(X) = \mathbf{A}^{(-1)}N_C(0.1858) = \mathbf{A}^{(-1)}(0.8142) = [0.8, 0.8]$ *e* ${\mathbb{N}_C}^{\mathbf{A}}(Y) = [0.8, 0.8]$*. Logo,* $X \preceq_{\mathbf{A}} Y$*,* $\mathbb{N}^{\mathbf{A}}_C(Y) \preceq_{\mathbf{A}} \mathbb{N}^{\mathbf{A}}_C(X)$*. E, portanto,* $\mathbb{N}^{\mathbf{A}}_S$ *não é uma negação forte.*

Uma $\langle L([0,1]), \preceq_{\mathbf{A}}\rangle$*-implicação* $\mathbb{I}\colon L([0,1])^2 \to L([0,1])$ *é uma função que verifica as condições de contorno (tabela clássica), incluindo antitonicidade no primeiro argumento e isotonicidade no segundo argumento relativa a* $\preceq_{\mathbf{A}}$*-ordem admissível.*

Teorema 14. *Seja* $I\colon [0,1]^2 \to [0,1]$ *uma implicação fuzzy,* $\mathbf{A}\colon L([0,1]) \to [0,1]$ *e* $\mathbf{A}^{(-1)}\colon [0,1] \to L([0,1])$ *as funções definidas em Def. 3 e 5, como requerido no Lema 1. A* $\langle L([0,1]), \preceq_{\mathbf{A}}\rangle$*-implicação* $\mathbb{I}^{\mathbf{A}}\colon L([0,1])^2 \to L([0,1])$ *é definida por*

$$\mathbb{I}^{\mathbf{A}}(X,Y) = \mathbf{A}^{(-1)}(I(\mathbf{A}(X),\mathbf{A}(Y))), \tag{14}$$

Prova. Seja $I\colon [0,1]^2 \to [0,1]$ uma implicação fuzzy. Então, tem-se verificadas:
$\mathbb{I}^{\mathbf{A}}(\mathbf{0},\mathbf{0}) = \mathbf{A}^{(-1)}(I(\mathbf{A}(\mathbf{0}),\mathbf{A}(\mathbf{0}))) = \mathbf{A}^{(-1)}(I(0,0)) = \mathbf{1}$;
$\mathbb{I}^{\mathbf{A}}(\mathbf{0},\mathbf{1}) = \mathbf{A}^{(-1)}(I(\mathbf{A}(\mathbf{0}),\mathbf{A}(\mathbf{1}))) = \mathbf{A}^{(-1)}(I(0,1)) = \mathbf{1}$;
$\mathbb{I}^{\mathbf{A}}(\mathbf{1},\mathbf{0}) = \mathbf{A}^{(-1)}(I(\mathbf{A}(\mathbf{1}),\mathbf{A}(\mathbf{0}))) = \mathbf{A}^{(-1)}(I(1,0)) = \mathbf{0}$;
$\mathbb{I}^{\mathbf{A}}(\mathbf{1},\mathbf{1}) = \mathbf{A}^{(-1)}(I(A(\mathbf{1}),A(\mathbf{1}))) = \mathbf{A}^{(-1)}(I(1,1)) = \mathbf{1}$.

($\mathbb{I}1$) Se $X_1 \preceq_{\mathbf{A}} X_2$, qualquer $\mathbf{A}(X_1) < \mathbf{A}(X_2)$ ou $\mathbf{A}(X_1) = \mathbf{A}(X_2)$. Por $I1$, nós temos que $I(\mathbf{A}(X_1),\mathbf{A}(Y)) \geq I(\mathbf{A}(X_2),\mathbf{A}(Y))$. Baseado no Lema 1 e na Equação (1), $\mathbb{I}^{\mathbf{A}}(X_1,Y) = \mathbf{A}^{(-1)}(I(\mathbf{A}(X_1),\mathbf{A}(Y))) \succeq_{\mathbf{A}} \mathbf{A}^{(-1)}(I(\mathbf{A}(X_2),\mathbf{A}(Y))) = \mathbb{I}^{\mathbf{A}}(X_2,Y)$. Caso contrário, se $\mathbf{A}(X_1) = \mathbf{A}(X_2)$, então $\mathbf{A}^{(-1)}(\mathbb{I}^{\mathbf{A}}(X_1,Y)) = \mathbf{A}^{(-1)}(\mathbb{I}^{\mathbf{A}}(X_2,Y))$.
($\mathbb{I}2$) De forma análoga. □

Exemplo 10. *Apresentamos duas QL-implicações representáveis em* $\langle L([0,1]), \preceq_{\mathbf{A}}\rangle$*:*
$\mathbb{I}^{\mathbf{A}}_{LK}(X,Y) = \mathbf{A}^{(-1)}(\min(1 - \mathbf{A}(X) + \min(\mathbf{A}(X),\mathbf{A}(Y)),1))$
$\mathbb{I}^{\mathbf{A}}_{KP}(X,Y) = \mathbf{A}^{(-1)}(\min(1 - (\mathbf{A}(X))^2) + \mathbf{A}(X)\cdot\mathbf{A}(Y),1))$

5. Conclusão

Este trabalho apresentou a representação intervalar de QL-implicações em $L([0,1])$ dotado de $\langle L([0,1]), \leq\rangle$-ordem parcial e $\langle L([0,1]), \preceq_A\rangle$-ordens admissíveis, enfatizando as propriedades das principais classes de operadores fuzzy valorados intervalarmente, como IvT, IvS, IvN e IvI. O trabalho promove uma abordagem alternativa para gerar conectivos da IvFL, permitindo o estudo da representabilidade desses operadores em $\langle L([0,1]), \preceq_A\rangle$.

Os resultados apresentados colaboram para a atual pesquisa, para definição da entropia fuzzy intuicionista valorada intervalarmente considerando $\langle L([0,1]), \preceq_A\rangle$-ordens. Outros estudos consideram extensões da $\preceq_A$-ordem para bicondicionais e diferenças fuzzy, incluindo agregadores fuzzy, como funções overlap e grouping.

Referências

Bedregal, B., Dimuro, G. P., Santiago, R., and Reiser, R. (2010). On interval fuzzy S-implications. Information Sciences*, 180(8):1373–1389.*

Bedregal, B. and Takahashi, A. (2006). Interval valued versions of t-conorms, fuzzy negations and fuzzy implications. In IEEE Intl Conf on Fuzzy Systems*, pages 1981–1987.*

Bustince, H., Fernandez, J., Kolesárová, A., and Mesiar, R. (2013). Generation of linear orders for intervals by means of aggregation functions. Fuzzy Sets Syst*, 220:69 – 77.*

Deschrijver, G. (2008). A representation of t-norms in interval-valued L-fuzzy set theory. Fuzzy Sets Syst*, 159(13):1597–1618.*

Deschrijver, G. and Cornelis, C. (2007). Representability in interval-valued fuzzy set theory. Int Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 15(03):345–361.*

Hickey, T., Ju, Q., and Van Emden, M. H. (2001). Interval arithmetic: From principles to implementation. J. ACM*, 48(5):1038–1068.*

Reiser, R. H. S., Dimuro, G. P., Bedregal, B. R. C., and Santiago, R. H. N. (2007). Interval valued ql-implications. In Logic, Language, Information and Computation, 14th Int. Workshop, WoLLIC 2007, Proc.*, volume 4576, pages 307–321. Springer.*

Santana, F., Bedregal, B., Viana, P., and Bustince, H. (2020). On admissible orders over closed subintervals of [0,1]. Fuzzy Sets Syst*, 399:44–54.*

Santiago, R., Bedregal, B., and Acióly, B. (2006). Formal aspects of correctness and optimality of interval computations. Formal Aspects of Computing*, 18(2):231–243.*

Zadeh, L. A. (1965). Fuzzy sets. Information and control*, 8(3):338–353.*

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning–I. Information Sciences*, 8(3):199–249.*

Sintaxe baseada em gramáticas de grafos para uma linguagem funcional visual voltada ao aprendizado de programação*

Marina Silva[1], **Ana Paula Lüdtke Ferreira**[1]

[1]Universidade Federal do Pampa
Campus Bagé

{marinasds2.aluno,anaferreira}@unipampa.edu.br

Abstract. *Imperative-based programming languages tend to emphasize syntactic aspects to the detriment of the problem-solving process. This work presents the syntax of the functional visual language Pandora, built from a graph grammar equipped with an algebra, whose rules emphasize the construction of a program with an emphasis on specification composition and reuse.*

Resumo. *Linguagens baseadas no paradigma imperativo tendem a enfatizar aspectos sintáticos em detrimento do processo de resolução de problemas. Este trabalho apresenta a sintaxe da linguagem visual funcional Pandora, construída a partir de uma gramática de grafos equipada com uma álgebra, cujas regras enfatizam a construção de um programa com ênfase em composição e reúso de especificações.*

1. Introdução

As primeiras experiências com programação no ensino superior costumam ser com linguagens de propósito geral fundadas no fluxo de execução do paradigma imperativo [Gomes e Mendes 2007, Pimenta 2019]. A complexidade das linguagens escolhidas e a forma como se representam os algoritmos em linguagens imperativas faz com que os métodos de ensino direcionem o foco (tanto do professor quanto do aluno) para a sintaxe da linguagem, em vez de destacar o processo de resolução de problemas por meio da programação, fazendo com que a atividade de desenvolvimento de algoritmos acabe com o foco na retirada de erros de compilação e não no pensar a solução de problemas.

A Engenharia de Software moderna preconiza o desenvolvimento modular e o reúso. O aproveitamento de especificações, modelos e códigos já construídos aumenta a produtividade das equipes de desenvolvimento e facilita os procedimentos de verificação, pelo uso de artefatos já testados, inclusive em ambiente de produção [Pressman 2016]. O reúso efetivo de artefatos de software exige do desenvolvedor um pensamento voltado à composição dos artefatos já conhecidos para a solução de um problema novo. A habilidade de compor especificações ou código não pode ser desenvolvida quando os estudantes, a cada nova tarefa, são instados a construir sempre uma solução a partir do zero.

Linguagens funcionais apresentam duas vantagens em relação ao ensino de algoritmos, nesse contexto. A primeira é que são focadas na solução de um problema, com definição das entradas e saídas, sem preocupação com detalhes de implementação e sem

*Trabalho de conclusão de curso de graduação, concluído.

desviar (muito) o foco do problema para a sintaxe. A segunda é que favorecem a ideia de que cada função deve ter um único propósito, devido à impossibilidade de efeitos colaterais gerados pelo programa, favorecendo modularidade e reúso de especificações. Como a noção de memória não está presente no paradigma, dificuldades com atribuições, alto acoplamento de módulos causados por variáveis globais e programação monolítica não ocorrem com facilidade.

Uma das formas de desenvolver habilidades de composição de artefatos de software é ensinar os alunos a programar por meio de linguagens que facilitem o uso de componentes já desenvolvidos para a construção de uma nova solução. O paradigma funcional de programação possui essa característica composicional [Sebesta 2018], ainda que as linguagens funcionais textuais possuam sintaxes que não favorecem o ensino. Cita-se, como exemplos, LISP [Touretzky 1990], Haskell [Thompson 1999], CLOS [Keene 1989], Elixir [Almeida 2018] ou Scala [Odersky et al. 2008].

Linguagens visuais já se provaram efetivas para aumentar o interesse de jovens pela área de programação [Resnick et al. 2009, Souza e Castro 2016, Feijó e De la Rosa 2016, Watanabe et al. 2020, Silva 2021]. A proposta a ser desenvolvida neste trabalho é reunir as vantagens da composicionalidade associada à programação funcional em uma linguagem visual que possa ser usada por iniciantes, com foco no processo de resolução de problemas, movendo a atenção do processo de compilação para o processo de desenvolvimento e execução de código. Espera-se, dessa forma, que os estudantes habituem-se a pensar em termos da solução de um problema como uma combinação da solução de problemas menores, favorecendo o pensamento lógico, o raciocínio abstrato, a modularidade e o reúso de especificações. Espera-se que o desenvolvimento dessas habilidades e postura frente à programação possa ser mantida quando os programadores migrarem para linguagens textuais com fluxo de execução imperativo, usadas em cursos de programação e na indústria de software.

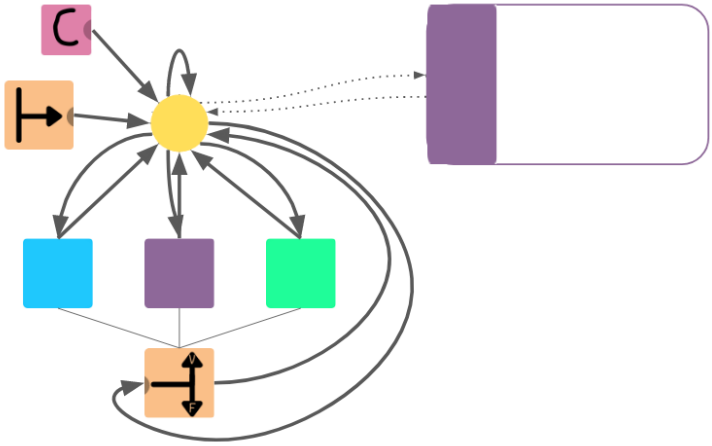
Este trabalho objetiva apresentar a sintaxe formal e a semântica informal da linguagem funcional visual Pandora, projetada para ser usada como primeira linguagem de programação em processos de ensino e aprendizagem em qualquer nível, incluindo o apoio ao ensino de Matemática no ensino fundamental. Essa é uma primeira aproximação do projeto da linguagem, que ainda não tem tipos de dados ou mesmo um interpretador construído. Contudo, tanto a sintaxe quanto a semântica de uma linguagem devem estar formalmente definidas para que se possa construir interpretadores corretos.

O restante do texto está organizado como se segue: a Seção 2 apresenta os princípios de construção e a sintaxe linguagem, a Seção 3 apresenta exemplos de programas em Pandora e a Seção 4 faz uma discussão sobre o trabalho e apresenta os desenvolvimentos futuros.

2. A linguagem Pandora

Um programa funcional é uma coleção de funções que podem ser definidas e invocadas em qualquer parte do código. Cada função recebe uma sequência de $n \geq 0$ parâmetros e devolve um único resultado. Não há operação de atribuição, o que elimina a possibilidade de efeitos colaterais e exige que o programador pense em termos de entradas e saídas. Programas em Pandora são expressos por um grafo. Definições e chamadas de funções, parâmetros e valores de retorno são representados por vértices. Constantes são considera-

Figura 1. Grafo tipo da linguagem Pandora



das funções de aridade zero e representadas com um símbolo especial. Arcos representam fluxo de dados ao longo do programa.

A sintaxe da linguagem Pandora é dada por uma gramática de grafos [Ehrig et al. 1996b] equipada com uma álgebra, para que a semântica de dados e operações possam ser definidas. Uma gramática de grafos é uma generalização das gramáticas de Chomsky [Lewis e Papadimitriou 1998] para reescrita de grafos. Formalmente, um grafo é uma tupla $G = (V, E, src, tar)$ em que V é um conjunto de nodos ou vértices, E é um conjunto de arcos ou arestas e $src, tar : E \to V$ são as funções que mapeiam as arestas em seus respectivos nodos de origem e destino. As regras da gramática são morfismos entre grafos. Um *morfismo de grafos* entre os grafos $G_1 = (V_1, E_1, src_1, tar_1)$ e $G_2 = (V_2, E_2, src_2, tar_2)$ é um par de funções $(f_V : V_1 \to V_2, f_E : E_1 \to E_2)$ tal que, para cada $e \in E_1$ tem-se que $f_V(src_1(e)) = src_2(f_E(e))$ e $f_V(tar_1(e)) = tar_2(f_E(e))$. Grafos podem ser rotulados ou tipados para que sua representação visual seja mais informativa. Um *grafo tipado* é uma tupla $G_T = (G, t, T)$ onde G e T são grafos e t é um morfismo total (i.e., ambas as funções do morfismo são totais) entre G e T.

O grafo tipo da linguagem restringe o tipo dos elementos e das relações que podem aparecer no programa e é apresentado na Figura 1. Os vértices do grafo em formato quadrado representam funções: verde e azul para as funções aritméticas e lógicas construídas na linguagem, roxo para funções definidas pelo usuário, rosa para funções constantes e salmão para a operação de teste condicional (*if-then-else*); o círculo amarelo representa um valor numérico ou lógico. Cada função é rotulada com um identificador, que não aparece no grafo por questões de simplicidade. O retângulo com um elemento roxo representa o escopo da definição de uma função, ao que os demais elementos estão ligados. Para visualização mais clara, os elementos do corpo da função aparecerão sempre em seu interior, estrutura garantida pelas regras. Note-se que o grafo tipo não restringe a quantidade de arestas entre os nodos. As restrições devem ser garantidas pelas regras da gramática.

O conjunto de regras que descreve a sintaxe da linguagem é apresentado a se-

guir. A abordagem algébrica *single-pushout* para gramática de grafos [Ehrig et al. 1996a] é usada neste trabalho, por razão de simplicidade da descrição das regras. Como não há deleção de elementos na construção de um programa representado por grafos, a abordagem *double-pushout* é igualmente adequada. A gramática da linguagem construída é composta por 20 regras para a construção dos programas, apresentadas na Figura 2. Além dos símbolos apresentados anteriormente, também é utilizado um quadrado cinza menor como símbolo não terminal, para garantir a definição de pelo menos uma função.

As regras (i) e (ii) permitem a definição de novas funções, no número que se desejar. Note-se que a estrutura para a definição de uma nova função contém somente o nome e um valor de saída. Os parâmetros e o corpo da função são construídos com outras regras. O símbolo f é uma variável da álgebra de strings usada para construção de identificadores. Os nodos referentes às álgebras não estão sendo mostrados por questão de simplicidade de notação. As regras (iii)-(v) inserem novos elementos na definição de uma função, respectivamente: um novo parâmetro, uma nova chamada de função e um novo parâmetro em uma chamada de função. Todos os elementos criados estão vinculados ao escopo da função f. As regras seguinte implementam a composição dos dados, seja estabelecendo ligações entre valores de parâmetros e chamadas de função (vi), indicando o valor de retorno da função sendo definida (vii), compondo duas funções (viii) ou compondo o resultado do ramo executado como entrada de outra função (ix). Note-se que todas essas regras possuem condições de aplicação negativas. Ou seja, uma valor somente pode ser passado a um nodo se ele já não estiver recebendo um valor de outra origem. A regra (x) estabelece que uma função pode ser substituída por uma função constante. No caso, o nodo referente ao conjunto suporte da álgebra também é omitido da regra. As regras (xi)-(xiv) determinam que a função usadas no grafo deve ser substituída por um operador aritmético e as regras (xv)-(xviii) fazem o mesmo para operadores lógicos. A regra (xix) cria uma estrutura do tipo *if-then-else* no corpo de uma função, em que o valor de saída da função (existente) a determina e execução de b ou se c consoante for verdadeiro ou falso, respectivamente. A junção dos valores de saída estabelece para onde vai o resultado, que pode ser proveniente de qualquer um dos ramos da função. A regra (xx) permite a composição de funções em um ramo de uma estrutura condicional.

3. Programas em Pandora

As regras da gramática permitem que um programa possa ser construído interativamente, com adição de novas funções, parâmetros em funções já existentes e novas chamadas no corpo de uma função, em um processo bastante similar ao usado em atividades de programação com linguagens textuais. A passagem de uma linguagem visual para uma linguagem textual exige um nível mínimo de similaridade no processo. Contudo, da mesma forma que em linguagens textuais, um problema somente pode ser executado se sua sintaxe está correta. Particularmente, a execução de um programa em Pandora exige que algumas condições sejam verdadeiras, especialmente as que se referem a passagens de parâmetros: (i) todas as chamadas de funções devem receber argumentos de fluxos de dados válidos e (ii) toda definição de função deve ter algum valor ligado ao seu valor de retorno.

O processo de derivação de um programa Pandora por meio da aplicação de regras reforça as noções de composição de funções, passagem de parâmetros e obtenção de resultados. Os programas resultantes são tais que o fluxo dos dados e das chamadas é

Figura 2. Gramática da linguagem Pandora

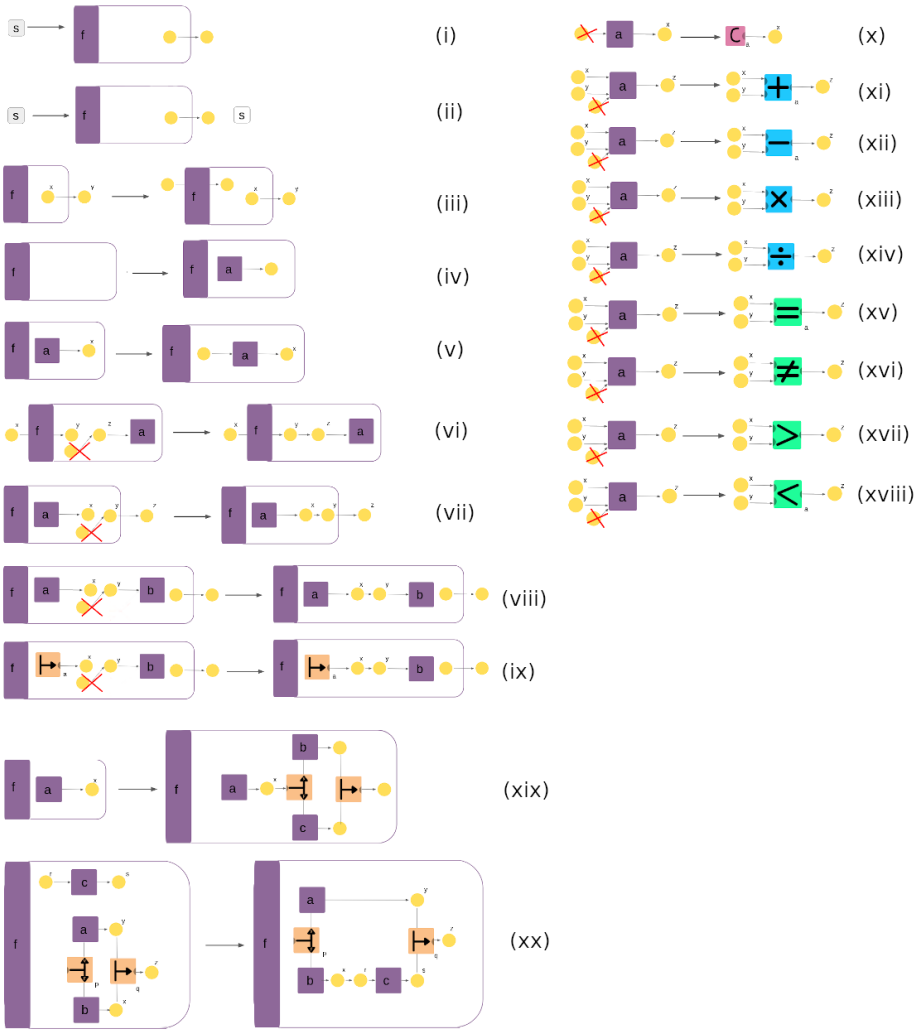


Figura 3. Função Pandora para adição e média de três valores

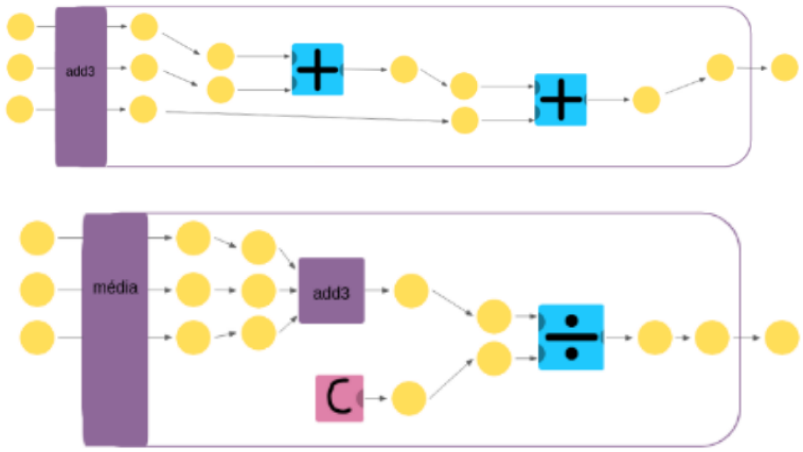
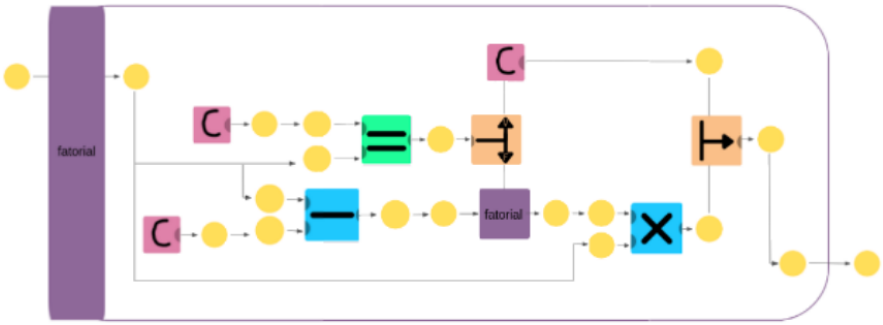


Figura 4. Função Pandora para o cálculo recursivo da função fatorial



tornado explícito. A Figura 3 apresenta a definição de uma função para o cálculo da adição de três valores (*add3*) e como essa função pode ser invocada por outra função que calcula a média de três valores (*média*), logo abaixo, na mesma figura.

A Figura 4 apresenta definição da função fatorial recursiva em Pandora, formalmente definida como

$$\text{fatorial}(n) = \begin{cases} 1 & n = 0 \\ n \cdot \text{fatorial}(n-1) & n > 0 \end{cases}$$

Note-se que não existem nomes de parâmetros em Pandora, ainda que eles possam ser incluídos sem esforço, fazendo uso da mesma álgebra usada para os nomes de funções. A função constante, neste ponto do desenvolvimento é somente indicada, mas espera-se implementar e mostrar o valor da constante, também por meio do uso de álgebras, que servirão para tipar os valores e encontrar erros em tempo de execução. A regra da composição permite que um mesmo valor seja argumento de diferentes funções, pontadas no grafo que representa o programa como setas com a mesma origem.

4. Conclusão

As construções da linguagem Pandora obedecem estritamente ao paradigma funcional, sem noção de memória ou de variáveis locais ou globais para armazenamento de resultados intermediários. Diferentemente das linguagens funcionais tradicionais, a notação da composição é dada por uma organização sequencial das funções, enfatizando a composição como um encadeamento de chamadas. O paradigma reforça a ideia de que cada função deve fazer uma única coisa e que funções podem ser combinadas para construir outras, acostumando o novo programador a pensar na questão do reúso de especificações. O foco da linguagem está em fazer o aprendiz pensar em termos de algoritmos como mapeamentos entre entradas e saídas, sem necessitar entender conceitos que têm a ver com implementações mas não com a resolução do problema proposto. Questões relacionadas à eficiência e ao armazenamento de resultados intermediários da computação podem ser codificados no interpretador de forma transparente ao usuário. A construção visual torna Pandora uma linguagem acessível para iniciantes em programação de diversas faixas etárias. A organização horizontal visa facilitar a transição para linguagens textuais.

Gramáticas de grafos são uma forma natural de representar estruturas visuais e sua sintaxe, formalmente. A semântica formal da linguagem Pandora está em construção. A partir da definição forma da semântica da linguagem, deverá ser construído um interpretador que possa ser executado sobre o grafo que representa o programa, enfatizando o fluxo de dados e o funcionamento das chamadas de funções. Já existe um proposta de IDE para a linguagem. A interface está sendo planejada de forma a facilitar as operações de encapsulamento de códigos em novas funções na edição de programas. A execução de um programa Pandora ma IDE, construída deverá mostrar os valores de parâmetros e resultados aparecendo nos círculos que correspondem ao fluxo de dados do programa.

Nesta definição inicial da sintaxe, somente funções numéricas e lógicas são suportadas. Pretende-se também, no futuro, expandir essa definição para construções que permitam tipos abstratos de dados e orientação a objetos. Para isso, gramáticas de grafos orientados a objetos e especificações algébricas deverão ser usados como suporte teórico.

Naturalmente que somente a definição sintática da linguagem não torna possível afirmar vantagens dessa abordagem sobre quaisquer outras, especialmente na questão do pensar computacional a longo prazo. Quando a IDE estiver completa, contudo, experimentos podem ser traçados de forma a mensurar – tanto em curto quando em médio prazo – o impacto do uso da linguagem no processo de solução de problemas por parte dos estudantes.

Referências

Almeida, U. (2018). *Learn Functional Programming with Elixir – New Foundations for a New World*. The Pragmatic Bookshelf.

Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., e Corradini, A. (1996a). Algebraic approaches to graph transformation. Part II: single-pushout approach and comparison with double pushout approach. In Ehrig, H., Kreowski, H.-J., Montanari, U., e Rozemberg, G., editors, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, chapter 4, pages 247–312. World Scientific, Singapore.

Ehrig, H., Kreowski, H.-J., Montanari, U., e Rozemberg, G. (1996b). *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1. World Scientific, Singapore.

Feijó, P. G. e De la Rosa, F. (2016). Roblock: Programming learning with mobile robotics. In *Proceedings of the ITiCSE '16*, page 361, New York, NY, USA. Association for Computing Machinery.

Ferreira, A. P. L. e Ribeiro, L. (2003). Towards object-oriented graphs and grammars. In Najm, E., Nestmann, U., e Stevens, P., editors, *International Conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS*, volume 2884 of *Lecture Notes in Computer Science*, pages 16–31, Paris. Berlin: Springer-Verlag.

Gomes, A. e Mendes, A. J. (2007). Learning to program - difficulties and solutions. In *International Conference on Engineering Education (ICEE 2007)*.

Keene, S. E. (1989). *Object-Oriented Programming in Common Lisp*. Addison-Wesley.

Lewis, H. R. e Papadimitriou, C. H. (1998). *Elements of the Theory of Computation*. Prentice Hall, Upper Saddle River, 2nd edition.

Odersky, M., Spoon, L., e Venners, B. (2008). *Programming in Scala*. Artima, 4th edition.

Pimenta, J. M. M. (2019). Temple - uma linguagem de programação para o ensino de programação. Mestrado, Universidade de Évora, Évora.

Pressman, R. S. (2016). *Engenharia de Software: Uma abordagem profissional*. Bookman, Porto Alegre, 8 edition.

Resnick, M., Maloney, J. H., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. A., Millner, A., Rosenbaum, E., Silver, J. S., Silverman, B. S., e Kafai, Y. B. (2009). Scratch: programming for all. In *Communications of the ACM*.

Sebesta, R. W. (2018). *Conceitos de linguagens de programação*. Bookman, Porto Alegre, 11 edition.

Silva, M. (2021). Proposta de uma linguagem composicional visual para ensino de programação. Trabalho de Conclusão de Curso (Graduação), Universidade Federal do Pampa.

Souza, S. S. e Castro, T. H. C. (2016). Investigação em programação com scratch para crianças: uma revisão sistemática da literatura. In *Anais dos Workshops do V Congresso Brasileiro de Informática na Educação (CBIE 2016)*.

Thompson, S. (1999). *Haskell: The Craft of Functional Programming*. Addison-Wesley, 2 edition.

Touretzky, D. S. (1990). *COMMON LISP: A Gentle Introduction to Symbolic Computation*. The Benjamin/Cummings Publishing Company, Inc., Redwood City.

Watanabe, T., Nakayama, Y., Harada, Y., e Kuno, Y. (2020). Analyzing viscuit programs crafted by kindergarten children. In *Proceedings of the ICER '20*, page 238–247, New York, NY, USA. Association for Computing Machinery.

Testando Mecanismos de Refatoração Utilizando Programas Gerados Aleatoriamente*

Luiz Felipe Kraus[1], **Bruno Coelho**[1], **Samuel da Silva Feitosa**[1]

[1] Instituto Federal de Santa Catarina (IFSC)– Caçador – SC – Brasil

{luiz.k1999, bruno.sc11}@aluno.ifsc.edu.br, samuel.feitosa@ifsc.edu.br

Abstract. *With the advance in need, the use of random program generators for tool testing has been presented, since the specific test cases required from the programmer's imagination, where hardly all possibilities are tested. In this sense, this work aims to use the random code generator to test as the main refactoring tools in the Java language, to find possible refactoring bugs in IDEs such as Netbeans, Eclipse and IntelliJ.*

Resumo. *Com o avanço na computação, o uso de geradores de programas aleatórios para testes de ferramentas tem se mostrado promissor, uma vez que os casos de testes definidos manualmente dependem da imaginação do programador, onde dificilmente são testadas todas as possibilidades. Neste sentido, este trabalho tem como objetivo utilizar o gerador de códigos aleatórios para testar as principais ferramentas de refatoração da linguagem Java, para encontrar possíveis bugs de refatoração em IDEs como Netbeans, Eclipse e IntelliJ.*

1. Introdução

Atualmente, o Java é uma das linguagens mais utilizadas no mundo [Cass 2019]. Por se tratar de uma linguagem de propósito geral, fortemente tipada e orientada a objetos, tem se tornado cada vez mais utilizada em projetos de médio à grande porte. Devido aos avanços nas ferramentas de desenvolvimento, a introdução de mecanismos de refatoração automática tem auxiliado na produção de código mais legível, simples e com melhor desempenho, visto que a proposta é alterar o código-fonte sem modificar o seu comportamento.

Entretanto, a partir desta refatoração automática, o projeto pode introduzir erros, sejam de sintaxe ou semântica, os quais não existiam antes da aplicação da refatoração, com isso o objetivo do projeto é utilizar programas gerados de forma aleatória para testar os principais mecanismos de refatoração da linguagem Java, testando assim a confiabilidade das ferramentas. Sendo assim, é necessário garantir que as alterações de código realizadas de forma automática mantenham as propriedades definidas inicialmente. Esta necessidade vem ao encontro do que se propõe neste projeto, sendo a realização de testes desses mecanismos de refatoração a partir da execução de milhares de programas ou trechos de código gerados de forma automática e aleatória.

Para possibilitar o desenvolvimento deste projeto, se faz necessário cumprir diversas etapas:

*Este trabalho encontra-se em fase de desenvolvimento.

- Identificação das principais refatorações aplicadas e os mecanismos mais utilizados.
- Criação de um mecanismo de geração de código Java que permita testar as refatorações.
- Execução de testes dos mecanismos de refatoração utilizando os códigos gerados de forma aleatória.

Para viabilizar o avanço deste projeto, foi desenvolvido um gerador de códigos Java, que permite a geração de acesso a atributos, invocação de métodos, criação de objetos, conversor de tipos, além de algumas diretivas de controle de fluxo (condicionais e repetição). Para tal, foram utilizadas as ferramentas *Java Reflection*[1], que obtém informações sobre código Java; a biblioteca *JavaParser*[2], que manipula os dados de construtores sintáticos e exportar código real; e a biblioteca *JQWik*[3], que dispõe da geração para construções básicas e permite a execução dos testes com as propriedades definidas.

O restante deste texto está organizado da seguinte forma: a seção 2 apresenta os trabalhos relacionados. As seções 3 e 4 apresentam os mecanismos de refatoração e a geração de código aleatório, respectivamente. Na seção 5 são apresentados os resultados parciais. Finalmente, a seção 6 encerra este documento com os resultados esperados e contribuições.

2. Trabalhos Relacionados

A ferramenta de teste de refatoração JDolly [Soares et al. 2013] com o mecanismo do SafeRefactor [Soares et al. 2009] são instrumentos que propõem uma técnica para testar os motores de refatoração da linguagem Java. A diferença entre a técnica utilizada no projeto citado e o presente projeto é a possibilidade de trabalhar com construções inseridas em versões mais recentes da linguagem Java, como, por exemplo, expressões lambda, referências para métodos, etc.

O trabalho de Kraus, Coelho e Feitosa [Kraus et al. 2021] apresenta uma arquitetura para a geração de código aleatório em Java, com o intuito de aplicar testes baseados em propriedades para verificar inconsistências em APIs. No presente artigo, é proposto a utilização do mesmo gerador de códigos aleatórios para a realização de testes dos mecanismos de refatoração presentes nas principais IDEs de desenvolvimento Java. Além destes, existem também diversos outros métodos e mecanismos para geração de código, como, por exemplo, a ferramenta de teste Csmith [Yang et al. 2011], sendo um gerador de programas para a linguagem C, onde suporta praticamente toda a geração dos recursos da linguagem.

3. Mecanismos de Refatoração Automática

Mecanismos de refatoração automática permitem a mudança de código sem alteração de comportamento, ou seja, ao aplicar uma refatoração, o programa deve executar apresentando o mesmo resultado. Essas ferramentas são muito utilizadas em Java, pois permite aos programadores uma melhor utilização dos conceitos implementados pela linguagem.

[1] `https://docs.oracle.com/javase/8/docs/api/java/lang/reflect/package-summary.html`

[2] `www.javaparser.org`

[3] `www.jqwik.com`

As principais IDEs (Ambiente de Desenvolvimento Integrado) da linguagem Java, como o IntelliJ, Eclipse e Netbeans possuem mecanismos de refatoração integrados. Estas ferramentas ajudam em problemas como extrair métodos, mover métodos, extrair classes, encapsular atributos, etc., além de sugerir modificações para que se utilizem os novos conceitos introduzidos na linguagem, para manter um código base atualizada.

4. Geração de Código Aleatório

O uso de testes com códigos gerados aleatoriamente na área de linguagens de programação é realizado desde meados da década de 60 [Sauder 1962]. Estes testes utilizam uma série de trechos de código como entrada para a checagem de diferentes condições que o sistema deve respeitar.

Como parte dos objetivos deste projeto, encontra-se em fase avançada de desenvolvimento um gerador de código aleatório Java. O trecho de código apresentado a seguir apresenta parte do método *genExpression*, o qual é responsável por gerar diferentes expressões de Java de forma recursiva.

```
@Provide
public Arbitrary<Expression> genExpression(Type t) {
    List<Arbitrary<Expression>> cand = new ArrayList<>();
    // Verifica se existem candidatos de tipos Primitivos
    if (t.isPrimitiveType()) {
        cand.add(Arbitraries.oneOf(mBase.genPrimitiveType(
        t.asPrimitiveType())));
    }
    ...
    //Verifica se existem candidatos de Methods
    if (!mCT.getCandidateMethods(t.asString()).isEmpty()) {
        cand.add(Arbitraries.oneOf(genMethodInvokation(t)));
    }
     return Arbitraries.oneOf(cand);
}
```

Figura 1. Código em Java: Geração de Código Aleatório

Como pode ser notado, o método *genExpression* recebe como parâmetro um tipo válido, que pode ser tanto de tipos primitivos como classes ou interfaces pré-definidas, resultando em possíveis expressões que devem ser do tipo solicitado. No trecho de código apresentado há duas condições: na primeira é verificado se o tipo a ser gerado é primitivo, o qual chama um método específico para proceder com a geração, adicionando o resultado em uma lista de expressões candidatas. De forma similar, na segunda condição é verificado se existe algum método na base de classes / interfaces existentes que possui tipo de retorno igual ao recebido por parâmetro, também adicionando as possíveis expressões candidatas na lista resultante.

5. Resultados Parciais

Para o desenvolvimento deste projeto, a partir das tecnologias escolhidas, foi definida a arquitetura apresentada na Figura 1.

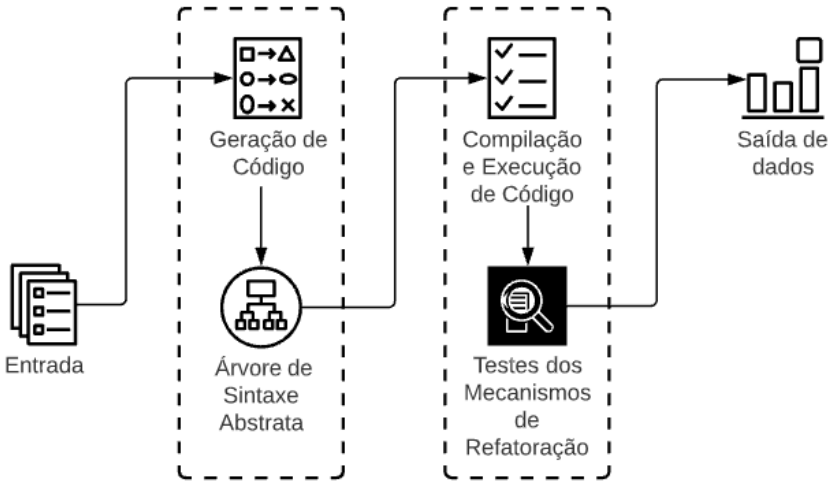


Figura 2. Arquitetura para implementação do projeto

A arquitetura proposta prevê a utilização de dois componentes principais: (1) o mecanismo gerador de código Java; e (2) a ferramenta que executa os códigos gerados antes e depois da refatoração. A execução é iniciada a partir da informação das entradas, que descreve as restrições que o gerador precisa respeitar para gerar testes válidos para a refatoração escolhida. Estas informações são repassadas para o módulo gerador (*JR-Generator*), que ao final, produz uma árvore de sintaxe abstrata contendo o caso de teste gerado, a qual será compilada para código Java, e inserida como entrada para a ferramenta de teste. Esta ferramenta, compila e executa o código gerado, coletando e armazenando seus possíveis resultados. Na sequência, é aplicada a refatoração escolhida no código original, repetindo o processo de compilação, execução e coleta de resultados. Ao final, os dados coletados são comparados para verificar possíveis inconsistências (erro de compilação ou execução) a partir dos códigos refatorados. Como saída, o sistema produz uma listagem contendo as estatísticas da execução.

A metodologia proposta para este projeto é dividida em cinco fases, conforme descrito a seguir.

- Fase 1: pesquisa a respeito do aparato ferramental para o desenvolvimento do projeto.
- Fase 2: definição de bibliotecas e *frameworks* que possibilitem a geração de código Java.
- Fase 3: criação do mecanismo de geração de código aleatório em Java.
- Fase 4: criação do instrumento que permita os testes de refatoração a partir dos códigos gerados.
- Fase 5: aplicar o instrumento de testes nas principais ferramentas de refatoração da linguagem Java.

Atualmente, o projeto encontra-se na Fase 3, possuindo o mecanismo de geração de código praticamente finalizado, dispondo da geração de tipos primitivos, acesso a atributos e métodos, construtores, *casts*, diretivas, etc. Na sequência serão realizadas

adaptações no mecanismo gerador, para permitir a entrada para o instrumento de teste, que deve ser implementado na sequência, possibilitando assim a execução dos testes dos mecanismos de refatoração.

6. Resultados Esperados e Contribuições

O escopo do projeto consiste no estudo dos mecanismos de refatoração automática da linguagem Java, e a realização de testes destas ferramentas a partir da execução de casos de teste gerados de forma aleatória. É sabido que através da utilização de casos de teste bem estruturados é possível detectar *bugs* em projetos de software de forma antecipada, evitando a liberação de ferramentas de desenvolvimento que possam introduzir problemas em projetos futuros.

Esta proposta pretende contribuir para o avanço desta área de pesquisa, fornecendo uma especificação formal (através de semântica operacional) e a implementação do mecanismo de geração de código utilizando a linguagem Java, além do instrumento de testes descrito neste artigo. Após o encerramento deste projeto, pretende-se ter uma ferramenta de software livre que permita agilizar o processo de testes e aumentar a confiabilidade das principais ferramentas de refatoração da linguagem Java.

Referências

Cass, S. (2019). As principais linguagens de programação de 2019. IEEE Spec.

Kraus, L. F., Schafaschek, B., and Feitosa, S. d. S. (2021). Desenvolvimento de um gerador de programas aleatórios em java. *Anais do Computer on the Beach*, 12(0):485–487.

Sauder, R. L. (1962). A general test data generator for cobol. In *Proceedings of the May 1-3, 1962, Spring Joint Computer Conference*, AIEE-IRE '62 (Spring), page 317–323, New York, NY, USA. Association for Computing Machinery.

Soares, G., Cavalcanti, D., Gheyi, R., Massoni, T., Serey, D., and Cornélio, M. (2009). Safe refactor: tool for checking refactoring safety.

Soares, G., Gheyi, R., and Massoni, T. (2013). Automated behavioral testing of refactoring engines. *IEEE Transactions on Software Engineering*, 39(2):147–162.

Yang, X., Chen, Y., Eide, E., and Regehr, J. (2011). Finding and understanding bugs in C compilers. *SIGPLAN Not.*, 46(6):283–294.

Theoretical Computer Science in Basic Education: A Systematic Review[‡]

Braz A. S. Junior[1], **Simone A. C. Cavalheiro**[1], **Luciana Foss**[1]

[1]Progama de Pós Graduação em Computação – Universidade Federal de Pelotas (UFPel)
CEP 96.010-610 – Pelotas – RS – Brazil

{badsjunior,simone.costa,lfoss}@inf.ufpel.edu.br

Abstract. *This paper presents a systematic literature review on theoretical computer science in basic education. Computing education has been reignited with the new computational thinking wave, calling out for the importance of problem-solving skills. Yet, it has been focusing its efforts in programming with visual languages, games and robotics. Meanwhile, theoretical computer science has rarely been touched in education. Therefore this work investigates 17 journal and conference paper that present solutions including this topic on basic education in the last 20 years. To conclude, a discussion about what topics of theoretical computer science were used takes place and a need for further research is identified.*

Resumo. *Este artigo apresenta uma revisão sisemática da literatura sobre informática teórica na educação básica. A educação em computação foi reacendida com a nova onda do pensamento computacional, clamando pela importância de habilidades para a resolução de problemas. No entanto, seus esforços têm se concentrado em programação com linguagens visuais, jogos e robótica. Enquanto isso, informática teórica tem sido raramente abordada na educação. Por isso, este trabalho investiga 17 artigos de periódico e de conferências que apresentam soluções incluindo este tópico na educação básica nos últimos 20 anos. Para concluir, é realizada uma discussão sobre os tópicos da informática teórica utilizados e indentificada uma demanda por mais pesquisas na área.*

1. Introduction

Computing Education (CEd) has gotten an increasing attention from the scientific community in the last years, making good use of promising and appealing allies, such as block-programming languages [Weintrop 2019], gamification [Gari et al. 2018], robotics [Yesharim and Ben-Ari 2018] and computational thinking [Angeli and Giannakos 2020]. Although, the roots of computation seems to have been left behind, **Systematic Literature Reviews (SLR)** [Garneli et al. 2015, Crick 2017] covering CEd as a whole have spotted that few content related to **Theoretical Computer Science (TCS)** has been approached.

[*]Concluded postgraduate paper.

[†]This work was supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, MCTIC/CNPq (Rede Sacci), SMED/Pelotas, PREC and PRPPG/UFPel.

A broad overview [Crick 2017] shows CEd has been done specially through the trend of using programming and the new central theme to the field: Computational Thinking. What "refers to a collection of computational ideas and habits of mind that people in computing disciplines acquire through their work in designing programs, software, simulations, and computations performed by machinery." [Crick 2017]. It is mentioned there that some pedagogic approaches are centred in topics of TCS: such as understanding automata [Isayama et al. 2016]; finite state and Turing machines [Korte et al. 2007]; and abstract/"notional" machines [Sorva 2007]. While others stand at least raise some discussion around them: as in algorithm design[Levitin 2016]; and concurrency/synchronization [Kolikant 2004].

Narrowing the scope, a SLR [Garneli et al. 2015] explores the educational contexts used for CEd in k-12. It makes clear how popular programming tools are in this field, specially visual programming languages. Studies making use of game design, tangible kits and robotics are also shown to be popular. The approaches closest to TCS in this SLR are about computer modeling and computation [Benacka and Reichel 2013, Sengupta and Farris 2012, Sengupta et al. 2013]; being 3 out of the 47 they analyzed.

These studies investigate CEd as a whole and focused on k-12, but they have no compromise with locating TCS within CEd. They actually expose how rare is TCS in CEd, far overshadowed by visual programming languages approaches. From this perspective, they leave us with the following Research Questions (RQ):

- **RQ1** - What are the studies explicitly treating TCS in k-12? Where are they from? Where are they published?
- **RQ2** - What are the topics or concepts of TCS being approached in k-12? What are the teaching strategies used?

Therefore, the purpose of this work is to find how is TCS going in k-12 education and discuss what could be its role there. That is why we present a SLR on TCS-centered approaches in k-12. The rest of this paper is organized as follows. Section 2 describes the method used in the research process. Section 3 presents the results and a discussion around them. Section 4 concludes the paper calling for further research.

2. Methods

We used the following four search engines due to their popularity and repository size within the fields of CS and/or Education: ACM Digital Library[1]; Springer Link[2]; IEEE Xplore[3]; and ERIC[4]. The search string was defined considering the synonyms of TCS and basic education as shown in Table 1. Producing the following search string:

```
("theoretical computer science" OR "theoretical
computing" OR "theoretical informatics") AND
((school OR primary OR secondary OR k-12 OR
basic OR kid*) AND education)
```

[1] https://dl.acm.org/
[2] https://link.springer.com/
[3] https://ieeexplore.ieee.org/Xplore/home.jsp
[4] https://eric.ed.gov/?

Table 1. Word table of synonyms for the search string

Theoretical	Computer Science Computing Informatics	basic k-12 primary secondary school kids	education

Using the search string in the aforementioned search engines we screened the title of the 1004 results (606 in ACM DL; 375 in Springer Link; 17 in IEEE Xplored; and 6 in ERIC) for studies meeting the inclusion criteria for this study were: written in English; being explicitly related to CEd; published in scientific journals or conferences; published within the last 20 years. Leaving 101 reports (74 in ACM DL; 16 in Springer Link; 8 in IEEE Xplored; and 3 in ERIC) to have their abstracts screened while consulting the following exclusion criteria: (1) does not explicit any content related to TCS; (2) does not target/include k-12 education; (3) unable to access; and (4) being a duplicate. At the final of the screenings, a total of 17 papers were considered, as summarized in Figure 1.

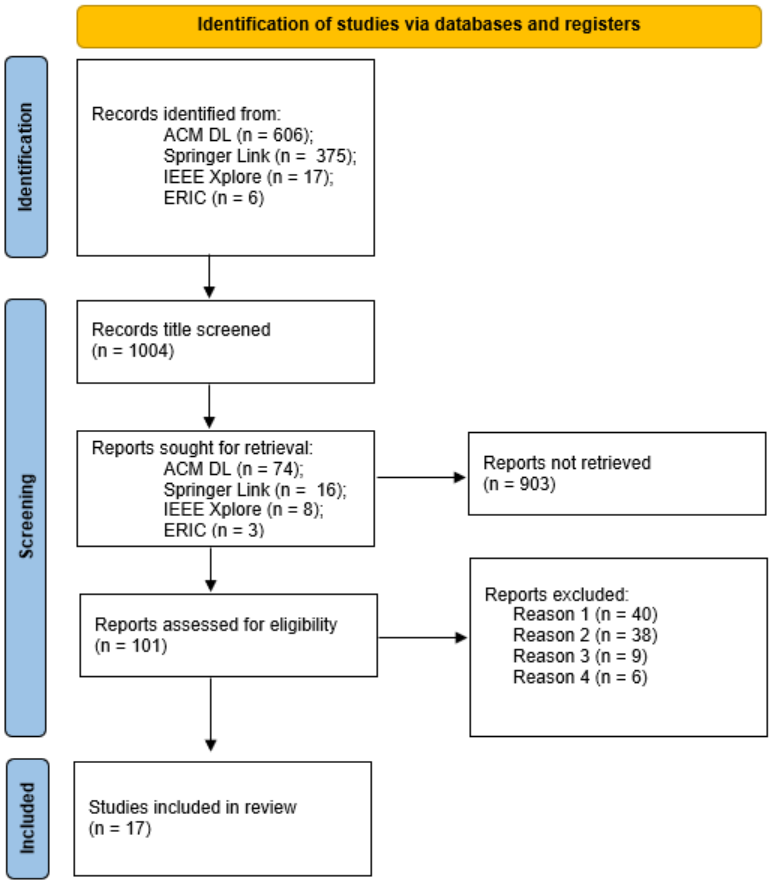


Figure 1. Work flowchart for this SLR.

3. Results and Discussion

Our SLR found 17 papers treating TCS in basic education, listed in the table 5, featuring the following. Traditional introductory courses on formal grammars/their corresponding automatons [Knobelsdorf et al. 2014] and graph theory [Carruthers et al. 2010], as well as virtual ones about complexity/computability [del Vado Vírseda 2019b, del Vado Vírseda 2020]. A moodle-supported online course on finite automata, push down automata and turing machines [Chuda 2007]. An automatic problem-generator tool to create exercises for students learning automatons [D'Antoni et al. 2020]. A game design activity using formal languages and regular expressions to build finite state and turing machines modeling games [Korte et al. 2007]. A puzzle game featuring fundamental concepts of automata theory [Isayama et al. 2016]. A card game using regular expressions to control finite state machines [Valente 2004]. A multi-agent system of ladybug robots being modeled by formal grammars [Kelemen and Kubík 2002]. A software (KARA) that allows students to control a ladybug using a FSM [Kiesmüller 2009]. A microworld environment of a zombie outbreak putting students to model the infection spread and containment measures [Maxville and Sandford 2020]. Flipped classroom experiments approaching mostly automata theory [Wolf et al. 2018, Morisse 2015]. And two papers discussing and suggesting curriculum, instead of actual experiments [Schlueter and Brinda 2008, Pantsar 2021].

In Tables 2-4, the column "N" reports the amount of papers found in the SLR that fits the row, this is represented between parenthesis in this paragraph. The studies were published in a variety of conferences and two journals, TOCE and Erkenntnis, ranging their themes from computer aided verification and robot motion to computing education and learning technologies, as listed in table 2. The authors, as seen in the Table 5, come from Germany (3), Spain (3), Slovakia (2), USA (2), Denmark (1), Finland (1), Australia (1), Scotland (1), Canada (1) and Japan (1). The disposition of TCS topics is shown in Table 3 as follows. The most approached topic was general Automata Theory (6), followed by specifically Finite State Machines/Automata (5) and Turing Machines (4). Which evens with Complexity Theory (4) and Computability (4). Then other Models of Computation and Formal Languages draw (3), followed by Regular Expressions (2). At last, Graph Theory was approached by a single work (1). Meanwhile, the teaching strategies used are shown in Table 4 as follows. The top teaching strategy is the use of Virtual/Digital Tools (6), followed by traditional lectures/tests (3) and unplugged activities, such as card and puzzle games (3). Then, there are Flipped Clasroom (2), Microworlds (1) and Robotics (1) initiatives.

Most papers give a special attention to the problem solving aspect of their proposed activities. Noteworthy, even an automated analysis of problem solving strategies is presented using the data gathered by an educational software (KARA) [Kiesmüller 2009]. Another common factor in the papers is the discussion of lack of related work (experiments on TCS in basic education), saying they are outdated (1980s and 1990s) [del Vado Vírseda 2019a] or having to relate to other CEd works like LOGO programming [Valente 2004]. It is clear that TCS is taught in basic education primarily through models of computation, generically or specifically one kind (FSM/TM). Additionally, formal Languages are used mostly to introduce automata. In second place comes computability and complexity theory, always approached together.

Table 2. Conferences and Journals where the selected studies were published.

Acronym	Full Name	N
E-Learn	World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education	1
ICALT	International Conference on Advanced Learning Technologies	1
RoMoCo	International Workshop on Robot Motion and Control	1
ICCAV	International Conference on Computer Aided Verification	2
CompSysTech	International Conference on Computer Systems and Technologies	2
SIGCSE	ACM Technical Symposium on Computer Science Education	2
Koli Calling	Koli Calling International Conference on Computing Education Research	1
ITiCSE	SIGCSE Conference on Innovation and technology in computer science education	1
WiPSCE	Workshop in Primary and Secondary Computing Education	1
CompEd	ACM Conference on Global Computing Education	1
WCCCE	Western Canadian Conference on Computing Education	1
TOCE	ACM Transactions on Computing Education	2
Erkenntnis	Erkenntnis	1

Table 3. Topics of Theoretical Computer Science approached in the selected studies.

Acronym	Full Name	N
AT	Automata Theory	6
FSM	Finite State Machines	5
TM	Turing Machines	4
CT	Complexity Theory	4
Comp	Computability	4
FL	Formal Languages	3
MoC	Models of Computation	3
RegEx	Regular Expressions	2
GT	Graph Theory	1

Table 4. Teaching Strategies used in the selected studies.

Acronym	Full Name	N
Virt	Virutal/Digital Tool	6
Trad	Traditional Lectures and Tests	3
unP	Unplugged Card/Puzzle Game, Paper and Pencil	3
Flip	Flipped Classroom	2
mW	Microworlds	1
RT	Robotics/Tinkering	1

Table 5. Studies on Theoretical Computer Science in Basic Education

Reference	Publisher	Country	Topics	Strategy
[Kelemen and Kubík 2002]	RoMoCo	Slovakia	FL, MoC	RT
[Valente 2004]	ICALT	Denmark	MoC, FSM, RegEx	unP
[Chuda 2007]	CompSysTech	Slovakia	AT, FSM, TM	Virt
[Korte et al. 2007]	ITiCSE	Scotland	FL, Regex, FSM, TM	Virt
[Schlueter and Brinda 2008]	CompSysTech	Germany	AT	–
[Kiesmüller 2009]	TOCE	Germany	FSM	Virt
[Carruthers et al. 2010]	WCCCE	Canada	GT	Trad
[Knobelsdorf et al. 2014]	SIGCSE	USA	FL, AT, FSM, TM	Trad
[Morisse 2015]	ELC	Slovakia	FL, AT, Comp	Flip
[Isayama et al. 2016]	TOCE	Japan	AT	unP
[Wolf et al. 2018]	WiPSCE	Germany	AT	Flip
[del Vado Vírseda 2019a]	CompEd	Spain	Comp, CT	Trad
[del Vado Vírseda 2019b]	SIGCSE	Spain	Comp, MoC, CT	Trad
[del Vado Vírseda 2020]	Koli Calling	Spain	Comp, CT	Virt
[Maxville and Sandford 2020]	ICCAV	Australia	Mod	mW, Virt, unP
[D'Antoni et al. 2020]	ICCAV	USA	AT, FL	Virt
[Pantsar 2021]	Erkenntnis	Finland	Comp, CT, TM	–

4. Conclusion

This paper offers a SLR that was able to capture only 17 studies involving TCS in basic education using large and popular scientific search engines of the CS and Education fields. This shows a gap in the basic CEd, that stands large volumes of studies reporting the use of programming, gaming and robotics to teach CS. Given the current trend towards computational thinking and problem-solving, TCS should be considered more often in CEd due to its analytic and problem-centered nature.

References

Angeli, C. and Giannakos, M. (2020). Computational thinking education: Issues and challenges. *Computers in Human Behavior*, 105:106185.

Benacka, J. and Reichel, J. (2013). Computer modeling with delphi. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives*, pages 138–146. Springer.

Carruthers, S., Milford, T., Pelton, T., and Stege, U. (2010). Moving k-7 education into the information age. In *Proceedings of the 15th Western Canadian Conference on Computing Education*, pages 1–5.

Chuda, D. (2007). Visualization in education of theoretical computer science. In *Proceedings of the 2007 international conference on Computer systems and technologies*, pages 1–6.

Crick, T. (2017). Computing education: An overview of research in the field. *London: Royal Society*.

del Vado Vírseda, R. (2019a). Computability and algorithmic complexity questions in secondary education. In *Proceedings of the ACM Conference on Global Computing Education*, pages 51–57.

del Vado Vírseda, R. (2019b). Introducing theoretical computer concepts in secondary education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, pages 1272–1272.

del Vado Vírseda, R. (2020). From the mathematical impossibility results of the high school curriculum to theoretical computer science. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, pages 1–5.

D'Antoni, L., Helfrich, M., Kretinsky, J., Ramneantu, E., and Weininger, M. (2020). Automata tutor v3. In *International Conference on Computer Aided Verification*, pages 3–14. Springer.

Gari, M. R. N., Walia, G. S., and Radermacher, A. D. (2018). Gamification in computer science education: A systematic literature review. In *2018 ASEE Annual Conference & Exposition*.

Garneli, V., Giannakos, M. N., and Chorianopoulos, K. (2015). Computing education in k-12 schools: A review of the literature. In *2015 IEEE Global Engineering Education Conference (EDUCON)*, pages 543–551.

Isayama, D., Ishiyama, M., Relator, R., and Yamazaki, K. (2016). Computer science education for primary and lower secondary school students: Teaching the concept of automata. *ACM Transactions on Computing Education (TOCE)*, 17(1):1–28.

Kelemen, J. and Kubík, A. (2002). Robots and agents in basic computer science curricula. In *Proceedings of the Third International Workshop on Robot Motion and Control, 2002. RoMoCo'02.*, pages 309–317. IEEE.

Kiesmüller, U. (2009). Diagnosing learners' problem-solving strategies using learning environments with algorithmic problems in secondary education. *ACM Transactions on Computing Education (TOCE)*, 9(3):1–26.

Knobelsdorf, M., Kreitz, C., and B¨ohne, S. (2014). Teaching theoretical computer science using a cognitive apprenticeship approach. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 67–72.

Kolikant, Y. B.-D. (2004). Learning concurrency: evolution of students' understanding of synchronization. *International Journal of Human-Computer Studies*, 60(2):243–268.

Korte, L., Anderson, S., Pain, H., and Good, J. (2007). Learning by game-building: a novel approach to theoretical computer science education. In *Proceedings of the 12th annual SIGCSE conference on Innovation and technology in computer science education*, pages 53–57.

Levitin, A. (2016). Algorithm design strategies in cs curricula 2013: hits and misses. *Journal of Computing Sciences in Colleges*, 31(3):78–84.

Maxville, V. and Sandford, B. (2020). Computational science vs. zombies. In *International Conference on Computational Science*, pages 622–633. Springer.

Morisse, K. (2015). Implementation of the inverted classroom model for theoretical computer science. In *E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education*, pages 426–435. Association for the Advancement of Computing in Education (AACE).

Pantsar, M. (2021). Cognitive and computational complexity: Considerations from mathematical problem solving. *Erkenntnis*, 86(4):961–997.

Schlueter, K. and Brinda, T. (2008). Characteristics and dimensions of a competence model of theoretical computer science in secondary education. *ACM SIGCSE Bulletin*, 40(3):367–367.

Sengupta, P. and Farris, A. V. (2012). Learning kinematics in elementary grades using agent-based computational modeling: a visual programming-based approach. In *Proceedings of the 11th international conference on interaction design and children*, pages 78–87.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., and Clark, D. (2013). Integrating computational thinking with k-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, 18(2):351–380.

Sorva, J. (2007). Notional machines and introductory programming education. *Trans. Comput. Educ*, 13(2):1–31.

Valente, A. (2004). Exploring theoretical computer science using paper toys (for kids). In *IEEE International Conference on Advanced Learning Technologies, 2004. Proceedings.*, pages 301–305. IEEE.

Weintrop, D. (2019). Block-based programming in computer science education. *Communications of the ACM*, 62(8):22–25.

Wolf, A., Wilhelm-Weidner, A., and Nestmann, U. (2018). A case study of flipped classroom for automata theory in secondary education. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*, pages 1–6.

Yesharim, M. F. and Ben-Ari, M. (2018). Teaching computer science concepts through robotics to elementary school children. *International Journal of Computer Science Education in Schools*, 2(3).

Uma análise de técnicas de classificação para predição de valores atribuídos aos jogadores no jogo FIFA

Maurício D.C. Balboni[1], **Helida Santos**[1], **Giancarlo Lucca**[2]

[1]Programa de Pós-graduação em Engenharia da Computação
Centro de Ciências Computacionais
Universidade Federal de Rio Grande (FURG)

{balboni,helida}@furg.br

[2]Programa de Pós-Graduação em Modelagem Computacional
Universidade Federal de Rio Grande (FURG)

giancarlo.lucca@furg.br

Abstract. *This work seeks to compare an application of two different machine learning algorithms. Precisely we take in consideration Public-C and a rule-based algorithm fuzzy, FURIA. For this, it uses two tools that help in the elaboration of predictive models, the KEEL software and an R programming language. The work uses accuracy as a measure of the quality of predictive models, in addition to presenting their confusion matrices. Showing that the model using FURIA presents a performance considered satisfactory.*

Resumo. *Esse trabalho busca realizar um comparativo de uma aplicação de dois diferentes algoritmos de aprendizado de máquina. Precisamente levamos em consideração o Public-C e um algoritmo baseado em regras fuzzy, o FURIA. Para isso, utilizamos duas ferramentas que auxiliam na elaboração de modelos preditivos, o software KEEL e a linguagem de programação R. O trabalho utiliza a acurácia como medida de qualidade dos modelos preditivos além de apresentar suas matrizes de confusão. Mostrando que o modelo utilizando o FURIA apresenta um desempenho considerado satisfatório.*

1. Introdução

O aprendizado de máquina [Zhang 2020] é um subcampo da ciência da computação dedicado a desenvolver técnicas e algoritmos na qual possibilitam o computador a desenvolver modelos que aprendem automaticamente sobre determinada aplicação. Os aprendizados podem ser divididos em supervisionado e não supervisionado [Monard and Baranauskas 2003a]. No presente trabalho será abordado apenas o aprendizado supervisionado, que consiste em gerar modelos de aprendizado a partir de resultados pré-definidos, utilizando valores passados pela variável destino do modelo para a geração dos resultados de saídas. Esses valores são utilizados para realizar a supervisão das previsões do modelo, permitindo o ajuste do mesmo com base nos erros preditivos [Ayodele 2010].

Os problemas de classificação buscam predizer a qual classe uma determinada instância pertence dentro de um conjunto finito de classes. Esses problemas podem ter apenas duas classes, por exemplo: se uma pessoa possui determinada doença ou não

$\{$tem a doença, não tem a doença$\}$. Além disto, podem existir n classes possíveis de predizer, por exemplo: afim de predizer como está a temperatura, podemos dividir em três classes, como $\{$quente, morno, frio$\}$, ou em cinco classes como $\{$muito quente, quente, morno, frio, muito frio$\}$. Existem diversas maneiras de lidar com esse problema, e um deles são as árvores de decisões [Monard and Baranauskas 2003b]. Estas árvores são um mapeamento dos possíveis resultados de uma determinada aplicação, subdividindo os dados em conjuntos cada vez menores e mais específicos conforme a árvore vai se expandindo, até gerar um modelo que tenha a capacidade de predizer as classes com base nas características da aplicação.

Na lógica clássica também conhecida como lógica booleana [Chenci et al. 2011], os conjuntos são denominados puros e um dado elemento desse domínio pode pertencer, representado por 1, ou não pertencer, representado por 0, ao referido conjunto. Essa lógica mostra-se limitada ao definir alguns tipos de sistemas, visto que não existe uma fronteira bem definida para decidir quando um elemento pertence ou não a alguns respectivos conjuntos. Visando contornar essa situação, surge a lógica *fuzzy* [Zadeh 1965], na qual considera infinitos valores entre o intervalo $[0, 1]$. Assim, um valor é atribuído ao elemento do conjunto e chamado de grau de pertinência, que indica o quanto este elemento (ou informação) pertence ao conjunto no determinado intervalo.

Nos dias atuais, uma das aplicações da teoria dos conjuntos *fuzzy* desenvolvida por Zadeh [Zadeh 1965] são os sistemas de classificação baseados em regras *fuzzy* (FRBSs[1]). Estes tipos de sistemas constituem uma extensão dos sistemas baseados na lógica clássica.

Um FRBS apresenta dois componentes principais, sendo eles:

1. Sistema de inferência: coloca em prática o processo de inferência *fuzzy* necessário para obter uma determinada saída do FRBS quando uma entrada é especificada.
2. Base de regras *fuzzy*: representa o conhecimento sobre determinado problema a ser resolvido, na qual é constituída pelo conjunto de regras *fuzzy*.

O presente trabalho tem como objetivo realizar um comparativo de uma aplicação de aprendizado de máquina utilizando diferentes algoritmos e utiliza-se de diferentes ferramentas que auxiliam na elaboração de modelos preditivos. Foi escolhido o ambiente de programação R-Studio[2] para compilar a linguagem de programação R, e nele foi treinado um modelo de classificação com as configurações predefinidas pelo ambiente e com validação cruzada igual a dez. A ferramenta KEEL[3] foi utilizada para fins comparativos, com o algoritmo PUBLIC-C [Rastogi and Shim 2000] para modelos de classificação e o algoritmo FURIA [Hühn and Hüllermeier 2009] para realizar um modelo de regras *fuzzy*.

O presente trabalho está dividido da seguinte forma, na Seção 2 é possível encontrar a fundamentação teórica necessária para entendimento do trabalho. A Seção 3 apresenta a metodologia e na Seção 4 encontram-se os resultados das aplicações dos modelos de classificação, além das métricas de qualidade do mesmo. A Seção 5 apresenta a conclusão e trabalhos futuros.

[1]Neste trabalho, usaremos a sigla em inglês para *Fuzzy Rule-based Systems*

[2]www.rstudio.com

[3]Disponível em: www.keel.es

2. Fundamentação Teórica

Nesta seção, os principais conceitos relacionados ao trabalho são apresentados brevemente. Começamos apresetando os tópicos sobre a lógica *fuzzy*. Na sequência, os modelos utilizados são discutidos.

2.1. Teoria dos Conjuntos *Fuzzy*

Na teoria clássica, um elemento pode pertencer ou não a um conjunto. Dado um universo de discurso denotado por U, e um elemento x pertencente a U, define-se o grau de pertinência pela equação 1, essa função é chamada de função característica na teoria clássica de conjuntos.

$$\mu_U(x) = \begin{cases} 1, & \text{se } x \in U \\ 0, & \text{se } x \notin U \end{cases} \tag{1}$$

A fim de se obter uma caracterização mais ampla, uma vez que alguns elementos podem pertencer a um conjunto mais do que os outros, surge a logica *fuzzy* [Zadeh 1965], onde o valor de pertinência, pode assumir valores entre 0 e 1, sendo 0 indicando uma exclusão absoluta no conjunto e 1 assumindo uma inclusão total na pertinência. Essa generalização aumenta a capacidade de expressão da função característica. Formalmente seja U o conjunto universo na qual pode ser continuo ou discreto, e um conjunto *fuzzy* A pertencente ao universo U, define-se a função de pertinencia $\mu_A(x) :\rightarrow [0; 1]$

Para expressar classes é muito comum a utilização de valores qualitativos para expressar valores quantitativos. Uma variável categórica tem como característica assumir valores dentro de um conjunto de termos linguísticos, ou seja, palavras ou definições. Assim, ao invés de assumir instâncias numéricas, as mesmas assumem instâncias linguísticas, como por exemplo, a altura de uma pessoa, podemos representar essa variável quantitativa com 170 centímetros, ou discretizar esse valor para que o mesmo pertença a uma palavra ou definição, como o conjunto alto, baixo, médio.

2.2. O método FURIA

FURIA (*Fuzzy Unordered Rule Induction Algorithm*) [Hühn and Hüllermeier 2009] é uma extensão do algoritmo evolutivo de regras *fuzzy* RIPPER [Cohen 1995], preservando as vantagens oferecidas pelo RIPPER, o FURIA recebe algumas modificações visando um melhor desempenho e aprende regras *fuzzy* no lugar de regras convencionais e conjuntos de regras não ordenados ao em vez de listas de regras. Além disto, o FURIA utiliza-se de funções de pertinência trapezoidais [Amendola et al. 2005] para melhor se adaptar aos problemas.

2.3. O método PUBLIC-C

Uma árvore de decisão pode ser gerada com todos as instâncias do conjunto de treinamento. No entanto para gerar uma árvore com maior precisão e menos sensíveis a novas instâncias de teste pode ser utilizado uma árvore menor que também classifica todos os registros conhecidos [Quinlan and Rivest 1989]. Para isso, a maioria dos algoritmos realiza uma fase de poda após a sua fase de construção. A partir disto, o surge o Public-C, um algoritmo de classificação de árvore de decisão aprimorado, que integra a fase de poda com

a construção inicial da árvore. É previamente determinado se o nó deverá ser podado ou não durante a fase de poda, para isto, o PUBLIC-C calcula um limite inferior da subárvore de custo mínimo enraizada no nó. Essa estimativa é utilizada para a identificação dos nós que certamente serão podados, e com isso evitando o custo computacional em suas divisões [Rastogi and Shim 2000].

3. Metodologia

Para a realização desse trabalho, é importante ter conhecimento da base de dados utilizado para que se possa entender as decisões de pré-processamento utilizadas visando a obtenção de melhores resultados. Com isto, essa seção detalha todo o processo realizado.

3.1. Descrição da Base de Dados

A base de dados Fifa, foi retirado da plataforma "sofifa"[4], essa base de dados se refere as informações referentes aos jogadores de futebol do jogo "FIFA 2018", o mesmo compreende 88 atributos referentes a cada uma das suas 18207 instâncias de teste, esses atributos constituem as características dos jogadores, como seus atributos físicos, os valores de desempenho do atleta e também características relacionadas ao valor de mercado que o mesmo tem. O objetivo proposto é que a partir da base de dados em questão, se possa estimar o atributo "Value", na qual se refere ao valor de mercado de cada jogador presente no jogo.

3.2. Pré-processamento

Inicialmente é preciso destacar que a base de dados é muito grande. Precisamente, ele tem dimensão inicial de 18208 instâncias x 84 atributos, além de diversos dados fora de padrões reconhecidos pelas linguagens de programação e com muito valores faltantes em alguns instâncias de teste. Sendo assim, indispensável a realização de redução de atributos.

Para isto, foi excluído todas as instâncias de teste em que tinham valores faltantes, além de exclusão de atributos em que não eram relevantes computacionalmente para o modelo, como por exemplo os atributos "ID, PHOTO, Nationality, Flag, logo"e outros. Além disto, foi padronizados valores da base de dados, como no lugar de K em valores numéricos, foi substituído pelo valor 000, e no lugar de M foi substituído por 000000, além da retirada de caracteres não benéficos para o modelo, como por exemplo o caractere $. Com isto foi reduzido um total de 40 atributos, resultando numa base de dados com dimensões de 17605 instâncias x 44 atributos. Contudo ainda não é o ideal para realização dos treinamentos. Para tratar o atributo alvo, primeiramente foi verificado que ele possui valores numéricos, como a proposta do trabalho é realizar um aprendizado de classificação, foi realizado a discretização dos valores. As classes geradas foram definidas como "Entre 5700 e 8500, Entre 8500 e 11500, Maior que 11500 e Menor que 5700".

Ainda assim, o *dataset* continua com uma dimensionalidade muito grande, então foi realizada uma análise mais detalhada, e foi verificado como se comportavam os valores baixos, foi visualizado que todos as instâncias que tinha o valor menor que 3000 eram muito parecidas, praticamente iguais, então foi realizado a exclusão dessas instâncias, restando apenas 1583 instâncias. Ao final, a base de dados foi reduzida em 91% em

[4]Base de dados disponível em: https://www.sofifa.com.br

relação a minha base inicial. Resultando em uma dimensionalidade de 1582 instâncias x 44 atributos.

3.3. Configuração da proposta

Os dois algoritmos selecionados neste trabalho foram executados com validação cruzada igual a dez e com as configurações pré-definidas pelo KEEL, sendo elas:

3.3.1. FURIA

Número de otimizações igual a dois e número de divisões igual a três

3.3.2. PUBLIC-C

O parâmetro *NodeBetweenPrune* igual a vinte e cinco.

4. Resultados

Foi realizado o treinamento do modelo em R utilizando uma validação cruzada [Schaffer 1993] igual a dez e dividido em 80% dos dados para treinamento e 20% dos dados para teste. Na Tabela 1 se encontra a matriz de confusão [Li et al. 2004] dos dados para teste do modelo, e podemos visualizar que a grande maioria das predições foram errôneas nas classes laterais dela, apenas na classe ”Maior que 11500”que se encontra um erro consistente para todas as classes. O modelo obteve uma acurácia de 0.851. Na Figura 1 se observa as medidas de qualidade entre as classes, nela podemos visualizar que a classe que obteve a pior precisão nas predições foi a classe ”Entre 8500 e 11500”, enquanto a que apresentou a melhor precisão foi a classe ”Menor que 5700”.

Tabela 1. Matriz de Confusão do modelo em R

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	133	6	0	0
Entre 5700 e 8500	1	100	9	0
Entre 8500 e 11500	1	10	22	3
Maior que 11500	8	8	13	82

```
Statistics by Class:

                     Class: Entre 5700 e 8500 Class: Entre 8500 e 11500
Precision                              0.9091                    0.61111
Recall                                 0.8065                    0.50000
F1                                     0.8547                    0.55000
Prevalence                             0.3131                    0.11111
Detection Rate                         0.2525                    0.05556
Detection Prevalence                   0.2778                    0.09091
Balanced Accuracy                      0.8848                    0.73011
                     Class: Maior que 11500 Class: Menor que 5700
Precision                            0.7387                0.9568
Recall                               0.9647                0.9301
F1                                   0.8367                0.9433
Prevalence                           0.2146                0.3611
Detection Rate                       0.2071                0.3359
Detection Prevalence                 0.2803                0.3510
Balanced Accuracy                    0.9357                0.9532
```

Figura 1. Medidas de Qualidade das Classes

Na tabela 2 encontra-se a matriz de confusão das predições realizadas utilizando o software ”KEEL”com o algoritmo Public-C, nela foi utilizada uma validação cruzada igual a dez e o modelo foi dividido em treino e teste pela definição padrão do KEEL. A partir disto, temos a acurácia do modelo de teste definida em 0.857.

Tabela 2. Matriz de Confusão do modelo Public-C

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	575	35	0	24
Entre 5700 e 8500	40	356	19	20
Entre 8500 e 11500	1	18	146	32
Maior que 11500	0	6	31	279

Por fim, com base na tabela 3 encontra-se a matriz de confusão das predições realizadas pelo KEEL com o algoritmo de classificação *fuzzy* FURIA. Com base na tabela, podemos inferir que a acurácia do modelo de teste foi 0.865.

Tabela 3. Matriz de Confusão do modelo FURIA

Predição	Menor que 5700	Entre 5700 e 8500	Entre 8500 e 11500	Maior que 11500
Menor que 5700	596	15	1	22
Entre 5700 e 8500	35	357	28	15
Entre 8500 e 11500	3	37	124	33
Maior que 11500	6	3	14	293

5. Conclusão

Neste trabalho pode-se conhecer o funcionamento de duas ferramentas que auxiliam na execução de modelos preditivos de aprendizado de máquina, além de comparar qual retornou o melhor resultado. Além disto, foi aplicado técnicas de pré-processamento para tratar os dados, modelando as informações para que seja maximizado o desempenho computacional e a precisão na predição dos modelos treinados.

Com base nos resultados apresentados, podemos visualizar o ganho de precisão nos modelos ao adotar um modelo de classificação baseado em regras *fuzzy*, onde o mesmo teve um ganho de 0.9% na sua precisão em relação ao modelo utilizando o KEEL com o algoritmo Public-C. Além disto, o modelo obteve um aumento de 1.6% na acurácia em relação ao modelo de classificação obtido na linguagem de programação R.

Agradecimentos

Os autores gostariam de agradecer às agências de fomento CAPES (PNPD 464880/2019-00) e FAPERGS (ARD 19/2551-0001279-9) pelo auxílio financeiro.

Referências

Amendola, M., Souza, A. d., and BARROS, L. C. (2005). Manual do uso da teoria dos conjuntos fuzzy no matlab 6.5. *FEAGRI & IMECC/UNICAMP*, pages 1–44.

Ayodele, T. O. (2010). Types of machine learning algorithms. *New advances in machine learning*, 3:19–48.

Chenci, G. P., Rignel, D. G., and Lucas, C. A. (2011). Uma introdução á lógica fuzzy. *Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica*, 1(1).

Cohen, W. W. (1995). Fast effective rule induction. In *Machine learning proceedings 1995*, pages 115–123. Elsevier.

Hühn, J. and Hüllermeier, E. (2009). Furia: an algorithm for unordered fuzzy rule induction. *Data Mining and Knowledge Discovery*, 19(3):293–319.

Li, Y.-X., Tan, C. L., Ding, X., and Liu, C. (2004). Contextual post-processing based on the confusion matrix in offline handwritten chinese script recognition. *Pattern Recognition*, 37(9):1901–1912.

Monard, M. C. and Baranauskas, J. A. (2003a). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32.

Monard, M. C. and Baranauskas, J. A. (2003b). Indução de regras e árvores de decisão. *Sistemas Inteligentes-Fundamentos e Aplicações*, 1:115–139.

Quinlan, J. R. and Rivest, R. L. (1989). Inferring decision trees using the minimum description lenght principle. *Information and computation*, 80(3):227–248.

Rastogi, R. and Shim, K. (2000). Public: A decision tree classifier that integrates building and pruning. *Data Mining and Knowledge Discovery*, 4(4):315–344.

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning*, 13(1):135–143.

Zadeh, L. A. (1965). Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific.

Zhang, X.-D. (2020). Machine learning. In *A Matrix Algebra Approach to Artificial Intelligence*, pages 223–440. Springer.

Uma classificação de vinhos baseada em regras fuzzy utilizando o algoritmo FARC-HD

Vítor M. Magalhães[1], Jair O. G. Carmona[1], Giancarlo Lucca[1,2], Helida Santos[1], Eduardo N. Borges[1]

[1]Programa de Pós-Graduação em Computação
Centro de Ciências Computacionais, Universidade Federal do Rio Grande – FURG

[2]Programa de Pós-Graduação em Modelagem Computacional
Centro de Ciências Computacionais, Universidade Federal do Rio Grande – FURG

{vitor.magalhaes,jogonzalezc,giancarlo.lucca}@furg.br

{helida,eduardoborges}@furg.br

Abstract. *A widely used application of supervised machine learning is in classification problems. There are different approaches, with different techniques and algorithms, in order to deal with this problem. Fuzzy Rule-Based Classification Systems are a consolidated and well known approach. In this sense, this work aims to present an analysis of this technique, as well as one of its classifier algorithms, called FARC-HD. Precisely, this algorithm was applied to a classic wine classification problem, demonstrating its high interpretability and accuracy.*

Resumo. *Uma aplicação bastante utilizada do aprendizado de máquina supervisionado ocorre em problemas de classificação. Existem diferentes abordagens, com técnicas e algoritmos diversos, para lidar com este problema. Sistemas de classificação baseados em regras fuzzy são uma abordagem consolidada e amplamente utilizada. Nesse sentido, este trabalho tem por objetivo apresentar uma análise sobre esta abordagem, bem como um de seus algoritmos classificadores, chamado FARC-HD. Mais precisamente, este algoritmo foi aplicado em um problema clássico de classificação de vinhos, demonstrando sua alta interpretabilidade e precisão.*

1. Introdução

O Aprendizado de Máquina (AM) [Michalski et al. 2013] está presente em nosso cotidiano resolvendo diversos problemas que, por muitas vezes, sequer podemos notar. Pode-se citar suas aplicações em diversas áreas, como gestão de resultados de praticantes de atividade física [da Silva et al. 2020]; análise de reportagens em redes sociais buscando identificar *fake news* [Radicchi et al. 2019]; e até no sistema judiciário há inúmeras possibilidades de aplicação destes conceitos [Pedreira et al. 2021].

A aplicação dos conceitos de aprendizado de máquina naturalmente ocorre por um sistema de aprendizado, cuja definição conceitual é a de um programa de computador que toma decisões baseado em experiências acumuladas através das soluções bem sucedidas de problemas anteriores [Monard e Baranauskas 2003].

O AM é uma subárea da Inteligência Artificial [Russell e Norvig 2002], podendo ser considerado como um ramo em evolução de algoritmos computacionais

projetados para emular a inteligência humana aprendendo com o ambiente ao redor [El Naqa e Murphy 2015]. Diversas técnicas podem ser usadas para se aplicar o AM para resolução de problemas. Tratando-se de aprendizado de máquina supervisionado [Tan et al. 2005], mais especificamente para resolver problemas de classificação [Kotsiantis et al. 2007], uma das abordagens é a *Fuzzy Rule-Based Classification System* (FRBCS), ou sistema de classificação baseado em regras *fuzzy*. Sua principal diferenciação frente a outros classificadores é o fornecimento de modelos interpretáveis para o usuário final, justamente pela possibilidade de aplicação da lógica *fuzzy* [Zadeh 1996], discorrida no presente trabalho.

Sabendo-se que a compreensão detalhada dos algoritmos de processamento de informações para o AM pode levar a um melhor entendimento das habilidades de aprendizagem humana [Mitchell 1997], este trabalho tem como objetivo contribuir com o entendimento do funcionamento de um FRBCS. Nesse sentido, foi considerado o uso do algoritmo *Fuzzy Association Rule-based Classification model for High Dimensional problems* (FARC-HD) [Alcalá-Fdez et al. 2011], pelo fato de ser considerado um algoritmo estado da arte. Além disso, para verificar o método frente ao problema de classificação, o algoritmo foi aplicado em uma base de dados conhecida na literatura, com o objetivo de classificar o tipo de vinho - branco ou tinto - de acordo com resultados de análises.

O trabalho está organizado na seguinte forma: na Seção 2, a fundamentação teórica do presente trabalho é explicitada, partindo da lógica *fuzzy*, passando pelos *Fuzzy Rule-Based Classification Systems* e pelo algoritmo FARC-HD. Na Seção 3, os trabalhos relacionados são discutidos. Na Seção 4, a metodologia é apresentada, bem como as razões da escolha do algoritmo, os datasets e seu pré-processamento, e a configuração utilizada pelo algoritmo. Os resultados obtidos são apresentados e analisados na Seção 5. Por fim, na Seção 6, as conclusões são apresentadas.

2. Fundamentação teórica

Esta seção aborda os principais conceitos utilizados no desenvolvimento deste trabalho.

2.1. Lógica Fuzzy

Na teoria clássica (ou *booleana*) dos conjuntos, também conhecida como conjuntos *crisp*, um elemento pode pertencer (1) ou não pertencer (0) a um determinado conjunto. Entretanto, o mundo não funciona de maneira exata, e muito menos o raciocínio humano. A lógica clássica funciona, em sua plenitude, quando é aplicada a termos exatos e fronteiras bem definidas (não-vagas). Entretanto, quando ela é aplicada em ambientes incertos ou imprecisos, seu funcionamento deixa a desejar. Um exemplo que explica claramente sua inaplicabilidade a estas situações é o conhecido *Paradoxo de Sorites*: em que momento um monte de areia deixa de ser um monte, se formos retirando grão por grão?

Então, buscando expressar o funcionamento do mundo aplicado à teoria dos conjuntos, surge a teoria dos conjuntos *fuzzy*, na qual, um elemento pode pertencer parcialmente a um conjunto, abrindo uma infinidade de possibilidades entre o 0 e o 1 através dos graus de pertinência destes mesmos elementos aos conjuntos.

Assim sendo, a teoria dos conjuntos *fuzzy* é a predecessora da lógica *fuzzy*. Ela permite que esta última suporte modos de raciocínio que são aproximados ao invés de exatos. A modelagem linguística *fuzzy* permite lidar com sistemas através da construção de um

modelo linguístico que pode se tornar interpretável por seres humanos [Gacto et al. 2011]. Então, a lógica *fuzzy* pode ser vista como uma tentativa de formalização/mecanização de duas capacidades humanas notáveis: primeiro, a capacidade de conversar, raciocinar e tomar decisões racionais em um ambiente de informações imperfeitas. E, segundo, a capacidade de realizar uma ampla variedade de tarefas físicas e mentais sem quaisquer medições e cálculos [Zadeh 2008].

2.2. Sistemas de classificação baseados em regras *fuzzy*

Sistemas de classificação baseados em regras *fuzzy* são ferramentas úteis para lidar com problemas de classificação, enfatizando a significância das variáveis linguísticas e da lógica *fuzzy* [Zadeh 1975] e tendo por principal vantagem a alta interpretabilidade dos modelos de saída [Sanz et al. 2010].

Na Figura 1, verifica-se que um sistema *fuzzy* possui alguns componentes principais: dada uma entrada, inicialmente ela é *fuzzificada* através de um *fuzzificador*. O *fuzzificador* contém as funções de pertinência das variáveis linguísticas de entrada, recebendo um valor do universo de discurso e retornando o grau de pertinência do valor ao respectivo conjunto *fuzzy*. O próximo componente é a *máquina de inferência*, que é responsável por realizar todos os cálculos necessários, recebidos do componente *conhecimento*. Este é composto pelo banco de dados e pela base de regras do sistema. Por último, há ainda o *defuzzificador*, que além de possuir as funções de pertinência das variáveis linguísticas de saída, ele recebe os graus de pertinência para uma variável linguística - de saída - e retorna um valor para essa variável.

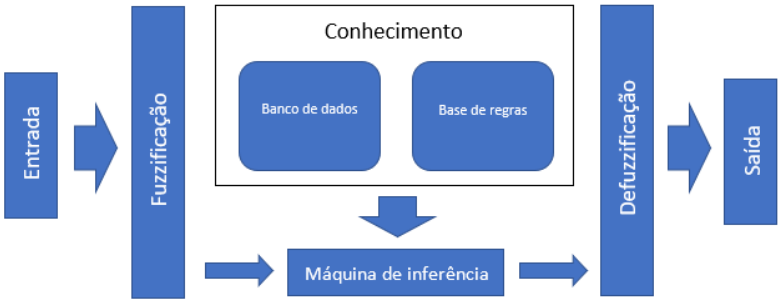


Figura 1. Componentes de um sistema *fuzzy*, adaptado de [Lucca 2018].

2.3. O algoritmo FARC-HD

A escolha por explorar o algoritmo FARC-HD em um problema de classificação deu-se pelo fato de o algoritmo basear-se em três grandes pilares de sustentação:

- obtenção das regras de associação *fuzzy* para classificação – uma árvore de busca é utilizada para listar todos os conjuntos possíveis de elementos *fuzzy* frequentes e para gerar regras de associação, limitando a profundidade dos ramos para encontrar um pequeno número de regras *fuzzy*;
- pré-seleção de regras candidatas – para diminuir o custo computacional do estágio de pós-processamento genético, o algoritmo usa a medida de acurácia relativa ponderada melhorada *wWRAcc* (sigla em inglês para *improved Weighted Relative Accuracy*) para pré-selecionar as regras mais importantes;

- seleção de regras genéticas e *tuning* lateral – um algoritmo genético é usado para selecionar e fazer a sintonia em um conjunto de regras de associação *fuzzy* com alta acurácia, para buscar a conhecida "sinergia positiva" apresentada por ambas as técnicas (seleção e *tuning*).

3. Trabalhos relacionados

Classificação *fuzzy* já foi empregada com sucesso em muitos contextos de problemas reais. No trabalho de [Assis Silva e Soares de Souza Lima 2009], foi utilizada para melhor visualizar as mudanças das classes de fertilidade do solo em cultivares de café, o que melhor definiu as zonas de transição gradual, ao invés de classificar as informações de forma exata. A lógica *fuzzy* foi aplicada para poder detectar e classificar falhas de curto-circuito em sistemas elétricos [Barros 2009]. Perturbações sutis diferenciavam o padrão em relação aos eventos caracterizados como de baixa e ou de média impedância. Um classificador baseado em regras *fuzzy* também já foi utilizado para combinar o conhecimento especializado do meteorologista com a velocidade e a objetividade de um computador, tendo suas regras formuladas através de conjuntos *fuzzy* para permitir a flexibilidade que o problema exigia [Bardossy et al. 1995]. Também pode-se citar a utilização de classificadores baseados em regras *fuzzy* para determinar o risco de um paciente sofrer de uma doença cardiovascular, fornecendo um modelo interpretável para explicar o resultado, servindo de base para a tomada de decisão da equipe médica [Sanz et al. 2014].

4. Metodologia

Esta seção apresenta a metodologia adotada para o desenvolvimento do trabalho, que baseou-se no processo de descoberta de conhecimento em bancos de dados (*Knowledge Discovery in Databases* - KDD) [Fayyad et al. 1996].

Após a definição do problema de classificação e consequente atributo-alvo, além das etapas de pré-processamento e transformação do *dataset*, é na etapa de *data mining* [Tan et al. 2005] que ocorre a aplicação do algoritmo FARC-HD. O método de funcionamento do algoritmo pode ser observado através da Figura 2.

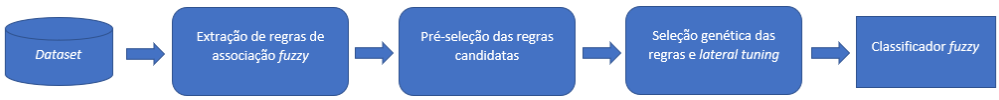


Figura 2. Método utilizado pelo algoritmo FARC-HD, adaptado de [Alcalá-Fdez et al. 2011].

Através da figura acima, verifica-se que, após receber o *dataset* como entrada, o primeiro passo realizado pelo algoritmo é a extração das regras de associação *fuzzy*. Após, é realizada a pré-seleção das regras candidatas e, por último, há a seleção genética das regras e o *lateral tuning*.

4.1. Datasets

Neste trabalho foram utilizados dois *datasets* distintos[1]. Um refere-se a resultados de análise para vinhos brancos, enquanto o outro analisa vinhos tintos. Os *datasets* possuem exatamente os mesmos 12 (doze) atributos, todos do mesmo tipo (ponto flutuante), diferenciando-se apenas pelo número de instâncias. O objetivo, então, é classificar o tipo de vinho - *branco* ou *tinto* - a partir dos resultados de análise.

4.2. Pré-processamento dos datasets

Ambos conjuntos de dados foram combinados em um mesmo *dataset*, tendo sido adicionado o atributo denominado *is red*, para identificar o tipo de vinho de acordo com o resultado de análise: 1 para vinho tinto e 0 para vinho branco, conforme a Tabela 1.

Tabela 1. Concatenação dos datasets considerados.

	acidez fixa	acidez volátil	ácido cítrico	açúcar residual	cloretos	dióx. de enx. livre	dióx. de enx. total	densidade	pH	sulfatos	álcool	qualidade	is red
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.99780	3.51	0.56	9.4	5	1
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.99680	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.99700	3.26	0.65	9.8	5	1
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.99800	3.16	0.58	9.8	6	1
4	7.4	0.66	0.00	1.8	0.075	13.0	40.0	0.99780	3.51	0.56	9.4	5	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...
5315	6.2	0.21	0.29	1.6	0.039	24.0	92.0	0.99114	3.27	0.50	11.2	6	0
5316	6.6	0.32	0.36	8.0	0.047	57.0	168.0	0.99490	3.15	0.46	9.6	5	0
5317	6.5	0.24	0.19	1.2	0.041	30.0	111.0	0.99254	2.99	0.46	9.4	6	0
5318	5.5	0.29	0.30	1.1	0.022	20.0	110.0	0.98869	3.34	0.38	12.8	7	0
5319	6.0	0.21	0.38	0.8	0.020	22.0	98.0	0.98941	3.26	0.32	11.8	6	0

Após, buscou-se eliminar instâncias que pudessem prejudicar o aprendizado em razão de sua baixa relevância. Não haviam instâncias com dados nulos, então foram apenas removidas as duplicadas. Após, foram analisadas as distribuições de alguns atributos e eliminadas instâncias pouco representativas. Por exemplo, a Figura 3 apresenta o histograma do atributo qualidade. Foram removidas as instâncias que possuíam valores inferiores a 5 e superiores a 7.

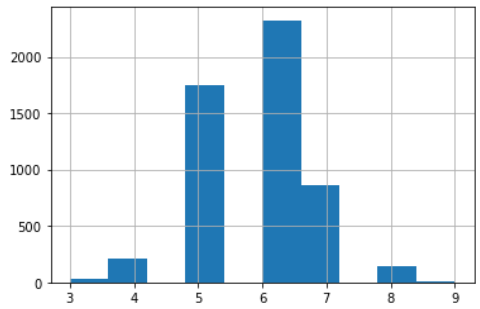


Figura 3. Histograma do atributo qualidade.

4.3. Configuração utilizada no FARC-HD

A técnica *K-Fold Cross Validation* (CV) é uma das abordagens mais usadas para validação de erro em classificadores [Tan et al. 2005]. Basicamente, consiste em dividir um conjunto de dados em *K* subconjuntos; então, iterativamente, *K-1* subconjuntos são usados

[1]Datasets do repositório de aprendizado de máquina - UCI, disponíveis em https://archive.ics.uci.edu/ml/datasets/wine+quality

para aprender o modelo, enquanto o restante é utilizado para avaliar o seu desempenho [Anguita et al. 2012]. Neste trabalho, foram utilizados 10 (dez) *folds* (subconjuntos).

Na configuração do algoritmo FARC-HD, para a obtenção dos resultados, foram utilizadas cinco variáveis linguísticas diferentes, com suporte mínimo de 0.05 (proporcionalidade de transações que contem o conjunto) e confiança máxima (estimativa de probabilidade) de 0.8. Destacamos a seguir os valores dos parâmetros usados no experimento, configurados no software KEEL[2]: *Seed = 1286082570; Number of linguistic values = 5; Minimun support = 0.05, Maximum confidence = 0.8, Depth of the trees (Depthmax) = 3, Parameter K of the prescreening = 2, Maximum number of evaluations = 15000, Population size = 50, Parameter alpha = 0.15, Bits per gen = 30, Type of inference = 1.*

5. Resultados

Após análise dos resultados obtidos pela saída do software KEEL, observou-se que justamente pela melhor definição dos limites de fronteiras entre as classes, característica de um sistema de classificação baseado em regras *fuzzy*, o algoritmo demonstrou um ótimo desempenho, obtendo uma acurácia média de 99.3% no subconjunto (partição) de treino e 98.9% no subconjunto (partição) de teste.

Por restrições de espaço, para demonstrar a assertividade do método, foi escolhido aleatoriamente um dos *folds*. Pode-se observar seus resultados abaixo:

- TotalNumberOfNodes: 4
- NumberOfLeafs: 5
- NumberOfAntecedentsByRule: 2.4

- NumberOfItemsetsTraining: 160
- NumberOfCorrectlyClassifiedTraining: 158
- PercentageOfCorrectlyClassifiedTraining: 98.75%
- NumberOfIncorrectlyClassifiedTraining: 2
- PercentageOfIncorrectlyClassifiedTraining: 1.25%

- NumberOfItemsetsTest: 18
- NumberOfCorrectlyClassifiedTest: 18
- PercentageOfCorrectlyClassifiedTest: 100.0%
- NumberOfIncorrectlyClassifiedTest: 0
- PercentageOfIncorrectlyClassifiedTest: 0.0%

Nos dados acima, verifica-se que, no *fold* supracitado, foram treinadas 160 instâncias, das quais 158 foram corretamente classificadas - uma acurácia do conjunto de treino de 98.75% - com apenas dois erros, um percentual de apenas 1.25%. Já no conjunto de teste, o resultado foi de 100% de acerto em todas as dezoito instâncias, demonstrando o desempenho excelente do classificador.

6. Conclusão

Problemas de classificação - aprendizado de máquina supervisionado - podem ser resolvidos com algoritmos de classificação baseados na lógica *booleana*. Nesse caso, os limites

[2]Ferramenta de *data mining* KEEL – https://www.keel.es

entre as diferentes classes de dados seriam mais facilmente identificados, tendo em vista que suas fronteiras são bem definidas. Todavia, em muitos problemas, a definição dessas fronteiras não é exatamente clara. Assim, objetivando predizer as classes às quais as instâncias pertencem através de um método de raciocínio *fuzzy*, considera-se infinitos valores no intervalo $[0, 1]$. Logo, as fronteiras entre as classes passam a ser definidas pelos graus de pertinência dos elementos, ou seja, o quanto um determinado elemento pertence a um determinado conjunto. Esta característica permite à lógica *fuzzy* trabalhar com termos linguísticos e, assim, a composição das regras se torna amigável ao entendimento humano, tornando os sistemas de classificação baseados em regras *fuzzy* muito úteis.

Neste trabalho, foi considerada a aplicação do algoritmo FARC-HD em um problema clássico de classificação de duas classes distintas de vinhos. Pode-se observar que tanto a delimitação dos triângulos das funções de pertinência, quanto o *lateral tuning* do algoritmo criaram fronteiras mais suaves e com limites mais bem definidos, resultando em uma acurácia de 98.75% para os dados de treino e 100% para os dados de teste, demonstrando de maneira objetiva a excelente performance neste problema.

Como sugestão para futuros trabalhos que complementem as avaliações aqui descritas, sugere-se um aprofundamento do estudo, diferenciando a presente abordagem da aplicação de técnicas de aprendizado de máquina tradicionais ou de agentes inteligentes e os impactos das possibilidades dos conjuntos *fuzzy* no mesmo problema, além da aplicação do mesmo algoritmo FARC-HD em diferentes problemas, explorando um pouco mais a alta interpretabilidade do modelo.

Agradecimentos

O presente trabalho foi realizado com apoio da CAPES (DS 88887.622570/2021-00, PNPD 464880/2019-00) e FAPERGS (19/2551-0001279-9).

Referências

Alcalá-Fdez, J., Alcala, R., e Herrera, F. (2011). A fuzzy association rule-based classification model for high-dimensional problems with genetic rule selection and lateral tuning. *IEEE Transactions on Fuzzy Systems*, 19(5):857–872.

Anguita, D., Ghelardoni, L., Ghio, A., Oneto, L., e Ridella, S. (2012). The 'k'in k-fold cross validation. In *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pages 441–446.

Assis Silva, S. d. e Soares de Souza Lima, J. (2009). Lógica fuzzy no mapeamento de variáveis indicadoras de fertilidade do solo. *Idesia (Arica)*, 27(3):41–46.

Bardossy, A., Duckstein, L., e Bogardi, I. (1995). Fuzzy rule-based classification of atmospheric circulation patterns. *Int. journal of climatology*, 15(10):1087–1097.

Barros, A. C. (2009). Detecção e classificação de faltas de alta impedância em sistemas elétricos de potência usando lógica fuzzy. Dissertação de Mestrado. Universidade Estadual Paulista (UNESP). Disponível em http://hdl.handle.net/11449/87092.

da Silva, H. d. B., Antoniazzi, R. L., Schuch, R. R., e Chicon, P. M. M. (2020). Gestão de atividades físicas com utilização de machine learning. *Anais do Seminário Interinstitucional de Ensino, Pesquisa e Extensão*.

El Naqa, I. e Murphy, M. J. (2015). What is machine learning? In *machine learning in radiation oncology*, pages 3–11. Springer.

Fayyad, U., Piatetsky-Shapiro, G., e Smyth, P. (1996). The kdd process for extracting useful knowledge from volumes of data. *Communications of the ACM*, 39(11):27–34.

Gacto, M. J., Alcalá, R., e Herrera, F. (2011). Interpretability of linguistic fuzzy rule-based systems: An overview of interpretability measures. *Information Sciences*, 181(20):4340–4360.

Kotsiantis, S. B., Zaharakis, I., Pintelas, P., et al. (2007). Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering*, 160(1):3–24.

Lucca, G. (2018). Aggregation and pre-aggregation functions in fuzzy rule-based classification systems. Tese de Doutorado. Universidad Pública de Navarra. Disponível em https://academica-e.unavarra.es/handle/2454/34775.

Michalski, R. S., Carbonell, J. G., e Mitchell, T. M. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.

Mitchell, T. M. (1997). Does machine learning really work? *AI magazine*, 18(3):11–11.

Monard, M. C. e Baranauskas, J. A. (2003). Conceitos sobre aprendizado de máquina. *Sistemas inteligentes-Fundamentos e aplicações*, 1(1):32.

Pedreira, M. R. G. et al. (2021). Inteligência artificial e machine learning no judiciário: as mudanças processuais e os impactos da tecnologia no sistema brasileiro. Trabalho de conclusão de curso. Universidade Católica do Salvador. Disponível em http://ri.ucsal.br:8080/jspui/handle/prefix/4441.

Radicchi, L. B. M., Barion, M. C., e Ferreira, A. (2019). Arquitetura de machine learning para análise de reportagens textuais em redes sociais para a detecção de fake news.

Russell, S. e Norvig, P. (2002). *Artificial intelligence: a modern approach*. Prentice Hall.

Sanz, J. A., Fernández, A., Bustince, H., e Herrera, F. (2010). Improving the performance of fuzzy rule-based classification systems with interval-valued fuzzy sets and genetic amplitude tuning. *Information Sciences*, 180(19):3674–3685.

Sanz, J. A., Galar, M., Jurio, A., Brugos, A., Pagola, M., e Bustince, H. (2014). Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. *Applied Soft Computing*, 20:103–111.

Tan, P.-N., Steinbach, M., e Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning – I. *Information Sciences*, 8(3):199–249.

Zadeh, L. A. (1996). Fuzzy sets. In *Fuzzy sets, fuzzy logic, and fuzzy systems: selected papers by Lotfi A Zadeh*, pages 394–432. World Scientific.

Zadeh, L. A. (2008). Is there a need for fuzzy logic? *Information sciences*, 178(13):2751–2779.

Visualização e Extensão de um Verificador de Modelos para a ferramenta Verigraph-GUI *

Arthur L. Fuchs, Rodrigo Machado, Leila Ribeiro

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil

alfuchs@inf.ufrgs.br, rma@inf.ufrgs.br, leila@inf.ufrgs.br

Abstract. *Verigraph-GUI is a graphical editor for the Verigraph tool, a graph transformation tool. This work reports on the construction of a graphical user interface for the model checking module of the Verigraph tool, as well as further planned extensions and further improvements.*

Resumo. *Verigraph-GUI é um editor gráfico desenvolvido para a ferramenta Verigraph, uma ferramenta de transformação de grafos. Esse trabalho relata sobre a construção de uma GUI para o módulo de verificação de modelos da ferramenta Verigraph, assim como extensões e outras melhorias planejadas.*

1. Introdução

O Verigraph-GUI [1] [Fuchs et al. 2019] é uma interface gráfica para a ferramenta de transformação de grafos Verigraph [2] [Azzi et al. 2018]. Essa interface inclui um editor de gramáticas de grafos e oferece acesso à execução e análise de especificações. Uma dessas funcionalidades é a verificação de modelos – técnica que permite verificar se um sistema satisfaz propriedades expressas por fórmulas de uma Lógica Temporal [Clarke 2008]. Tal módulo, contudo, estava até então restrito a uma interface de linha de comando. Este trabalho trata da construção de uma interface gráfica para a verificação de modelos no Verigraph-GUI e de melhorias planejadas para essa funcionalidade.

2. Gramáticas de Grafos e a ferramenta Verigraph-GUI

Gramáticas de grafos permitem especificar sistemas utilizando grafos para representar o estado do sistema, e regras de reescrita para descrever o comportamento do mesmo. Considerando a abordagem algébrica [Ehrig et al. 2006], no estilo *Double Pushout* (DPO), uma regra de grafos é representada por um span de (homo)morfismos totais de grafos $r : L \leftarrow K \rightarrow R$. Um morfismo de grafos é o mapeamento de vértices e arcos de um grafo G para vértices e arcos de um grafo H de forma a preservar os nodos de origem e o destino dos arcos. Nessa forma de representar regras, K serve de interface entre os grafos L e R e contém os elementos *preservados*. Os elementos de L e de R não inclusos nos morfismos K são, respectivamente, *deletados* e *criados* pela regra. Uma regra pode ser aplicada se for encontrado um *match* de L no grafo de estado G que queremos transformar. Essa ocorrência é modelada por um morfismo total de grafos, que faz o mapeamento de todos os elementos de L para G. A aplicação de uma regra é feita formalmente através

*Trabalho em andamento

[1]https://github.com/artFuchs/verigraph-GUI

[2]https://github.com/Verites/verigraph

de um diagrama de pushout duplo DPO. É comum o uso de mecanismos de tipagem de grafos para especificar diferentes categorias de nodos e arestas, e restringir as possibilidades de conexão de arestas nos grafos instância.

O Verigraph-GUI nasceu da necessidade de um editor de gramáticas de grafos para a ferramenta Verigraph, bem como da necessidade de uma interface para visualização de suas análises. Sua versão inicial foi apresentada em [Fuchs et al. 2019]. Desde então o editor de gramáticas de grafos foi estendido ter suporte à NACs e, recentemente, o Verigraph-GUI ofereceu suporte a módulos de análise já presentes no Verigraph. Até o momento, foram escritas mais de 7000 linhas no desenvolvimento do Verigraph-GUI.

A verificação de modelos foi implementada originalmente em [Becker 2014], em uma versão inicial do sistema Verigraph, apresentando somente uma interface de linha de comando. Este projeto tem como objetivo geral:

1. a construção de uma interface gráfica para o módulo de verificação de modelos CTL, com a possibilidade de geração e navegação pelo espaço de estados (incluindo questões relacionadas ao leiaute do espaço de estados);
2. a extensão deste mesmo para suportar outras Lógicas Temporais (LTL e CTL*);
3. a melhoria da performance deste verificador de modelos.

Este trabalho se encontra em andamento: neste artigo revisamos a verificação de modelos para gramáticas de grafos e relatamos resultados iniciais sobre o projeto acima, em particular a conclusão do primeiro objetivo. A versão atual do Verigraph-GUI (mostrada nas Figuras 1 e 2) já permite realizar a simulação da aplicação de regras no sistema, a análise de pares críticos, a visualização e manipulação do espaço de estados e a verificação de fórmulas CTL sobre esse espaço de estados.

3. Espaço de estados de uma gramática de grafos

A Figura 1 mostra uma gramática de grafos que representa um sistema de trens, originalmente descrito em [Becker 2014]. Os tipos de nodos representam trens, passageiros e estações. As arestas com rótulo in representam a localização de passageiros a trens e estações, assim como a posição de trens em estações. As arestas com rótulo destiny apontam a estação de destino do passageiro. A gramática consiste de três regras: a regra *moveTrain* simula a movimentação do trem entre estações; *embarkPassenger* simula um passageiro em uma estação entrando no trem; *disembarkPassenger* simula um passageiro descendo do trem para sua estação de destino. O grafo inicial descreve uma situação com três estações conectadas de forma circular, um único trem e um passageiro. No sistema Verigraph-GUI as regras são descritas através de um único grafo com marcações específicas para os elementos deletados (marcados em azul) e criados (marcados em verde) pela regra, de forma semelhante à edição de gramática no sistema Groove [Rensink 2004].

Gramáticas de grafos têm como características serem modelos de execução baseada em regras de reescritas, normalmente levando a não-determinismo e concorrência na sua execução. Essa característica pode tornar difícil a compreensão sobre os comportamentos possíveis do sistema a partir da descrição da gramática.

O espaço de estados é um sistema de transições, isto é, um grafo que descreve todos os estados do sistema gerados pela aplicação sucessiva das regras de reescrita. Assim como no sistema Groove, o Verigraph-GUI considera como as propriedades de um

dado estado as regras que possuem *matches* naquele estado em particulador. Transições são anotadas somente com o nome da regra utilizada, mesmo que haja mais de um *match* possível para tal regra. Dado que a abordagem algébrica para gramáticas de grafos apresenta não-determinismo de representação, ao calcular o espaço de estados nós identificamos grafos isomorfos. Desta forma, definimos cada nodo do espaço de estados como sendo uma classe de equivalência de grafos isomorfos, ao invés de grafos concretos. Para exemplificar, a Figura 2 mostra o espaço de estados gerado para a gramática da Figura 1. O estado 0, marcado em azul na figura, corresponde à classe de equivalência que inclui o grafo inicial da gramática.

Nota-se que gramáticas de grafos podem gerar espaços de estados infinitos, e, mesmo quando são finitos, o tamanho destes pode apresentar um número muito grande de estados, fato normalmente conhecido como *explosão do espaço de estados*.

4. Verificação de Modelos CTL

A ideia geral da técnica de verificação de modelos é: dado um modelo (sistema de transições) M e uma fórmula em Lógica Temporal *f*, achar todos os estados de M que satisfazem *f* [Clarke 2008]. A execução dessa técnica é automática e permite verificar se um sistema satisfaz a propriedade especificada pela fórmula. Ela foi desenvolvida para procurar por erros de concorrência em programas paralelos, erros tipicamente difíceis de reproduzir e encontrar em testes. Verificação de modelos pode ser usada para analisar o espaço de estados gerado por gramáticas de grafos, que como mencionado na seção anterior podem apresentar um número muito grande de estados ou até mesmo ser infinitos.

Lógicas Temporais permitem fazer afirmações sobre o tempo. Um dos tipos de Lógica Temporal mais conhecidos é a *Computer Tree Logic* (CTL). Essa lógica foi criada com o objetivo de sintetizar esqueletos de sincronização [Clarke 2008] e suas expressões permitem descrever propriedades com base nas possíveis ramificações e rotas do espaço de estados do sistema. Uma formula em CTL é definida pela seguinte sintaxe:

$$\varphi ::= \bot \mid \top \mid p \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \varphi \Rightarrow \varphi \mid \varphi \iff \varphi \mid (\varphi) \mid$$

$$AX\varphi \mid AF\varphi \mid AG[\varphi U \varphi] \mid EX\varphi \mid EF\varphi \mid EG\varphi \mid E[\varphi_1 U \varphi_2]$$

onde p é uma formula atômica. No caso de espaços de estados de gramáticas de grafos, as fórmulas atômicas consistem dos nomes de regras passíveis de serem aplicadas sobre o estado em questão. Os operadores $\bot, \top, \neg, \vee, \wedge, \rightarrow$ e $\iff$ funcionam da mesma forma que em lógica proposicional clássica. Os operadores A e E são quantificadores e expressam *todos os caminhos satisfazem* e *existe algum caminho que satisfaça*, respectivamente. O operadores X, F, G e U são *operadores temporais* : X verifica se uma propriedade é satisfeita no próximo estado; F verifica se uma propriedade é satisfeita eventualmente em um caminho; G verifica se uma propriedade é satisfeita por todos os estados em um caminho; U verifica se, em um caminho, todos os estados satisfazem φ_1 até que a fórmula φ_2 seja satisfeita.

Na verificação de modelos, cada estado de um modelo é analisado para ver se ele satisfaz a fórmula CTL especificada pelo usuário. Como exemplo, a Figura 2, mostra o resultado para a fórmula $A[moveTrain\, U\, embarkPassenger]$. Essa formula especifica a propriedade de que o trem deve sempre poder se mover entre estações até que seja possível

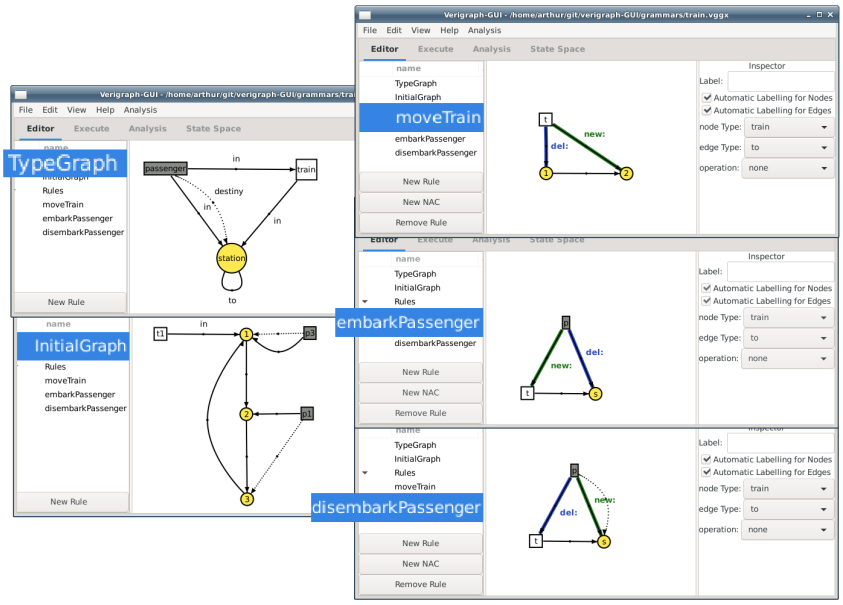


Figura 1. Gramática de exemplo especificada no Verigraph-GUI

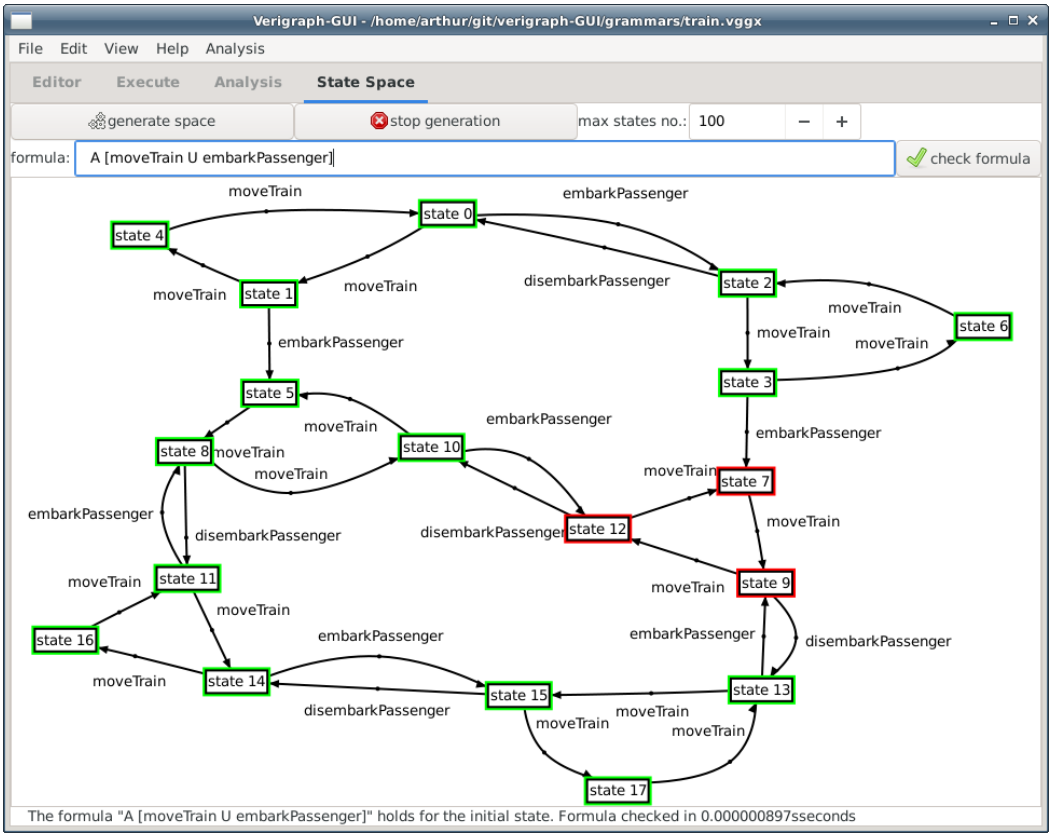


Figura 2. Espaço de estados gerado pela gramática de exemplo e resultado da verificação da fórmula CTL $A[moveTrain\ U\ embarkPassenger]$

que um passageiro embarque. Note que embora alguns estados não satisfaçam a fórmula, o estado 0 (estado inicial) o faz, sendo esta a verificação realizada pela ferramenta.

O algoritmo de satisfação de fórmulas CTL usado no Verigraph-GUI é descrito em [Huth and Ryan 2004], sendo utilizada a implementação feita em [Becker 2014]. O algoritmo funciona verificando quais estados satisfazem cada sub-fórmula de uma fórmula CTL ϕ. A avaliação das sub-fórmulas é feita em ordem crescente em relação aos seus tamanhos, deixando seus resultados disponíveis para serem usados nas sub-fórmulas maiores. Assim, por exemplo, a fórmula $A[moveTrain\ U\ embarkPassenger]$ tem o conjunto de sub-fórmulas $\{A[moveTrain\,U\,embarkPassenger], moveTrain, embarkPassenger\}$, e a execução do algoritmo encontra todos os estados que satisfazem $moveTrain$ e $embarkPassenger$ antes de verificar quais estados satisfazem $A[moveTrain\,U\,embarkPassenger]$.

5. Status do projeto e conclusões

Nesta seção, é relatado o estado atual do trabalho e os próximos passos a serem dados neste projeto.

O módulo de verificação de modelos havia sido implementado no Verigraph com uma interface textual [Becker 2014]. Neste trabalho foi desenvolvida uma GUI para este módulo no Verigraph-GUI, que pode ser vista na figura 2. Essa GUI, assim como as GUIs de outros módulos do Verigraph-GUI, foi projetada utilizando a ferramenta Glade, que permite a construção de GUIs de forma visual. Ela permite gerar e visualizar o espaço de estados da gramática, e também especificar uma formula CTL e visualizar os estados que a satisfazem. Sobre o que foi feito no desenvolvimento dessa GUI:

- O algoritmo de geração do espaço de estados implementado no trabalho de [Becker 2014] foi modificado para:
  - Fazer a utilização de busca em largura ao invés de busca em profundidade;
  - Limitar a busca a um número máximo de estados (a busca anteriormente era limitada à um nível máximo de profundidade)
  - Rodar em paralelo com a geração da visualização do espaço de estados. Isso permite que o usuário pare a geração do espaço de estados mantendo o modelo gerado até o momento.
- O algoritmo de satisfação implementado em [Becker 2014] foi integrado, permitindo que uma fórmula CTL especificada pelo usuário seja verificada caso o espaço de estados tenha sido gerado.
- Para visualização do espaço de estados,
  - É usado um algoritmo de leiaute que organiza os nodos do espaço de estado em níveis, gerando uma estrutura de diagrama. Os níveis são organizados de forma que o primeiro nível contenha apenas o estado inicial, e os outros níveis contenham os estados que possam ser alcançados apenas a partir de um estado do nível anterior. O leiaute gerado por esse algoritmo para a gramática da Figura 1 pode ser visto na Figura 3.
  - Essa estrutura é desenhada utilizando os módulos de renderização e de definição de *callbacks* para o *canvas* (área principal da janela), os quais são também utilizado nos módulos de edição e simulação. O resultado do

algoritmo de satisfação da formula CTL é usado para definir cores de destaque para os nodos do grafo: verde se a fórmula é satisfeita, e vermelho caso contrário.

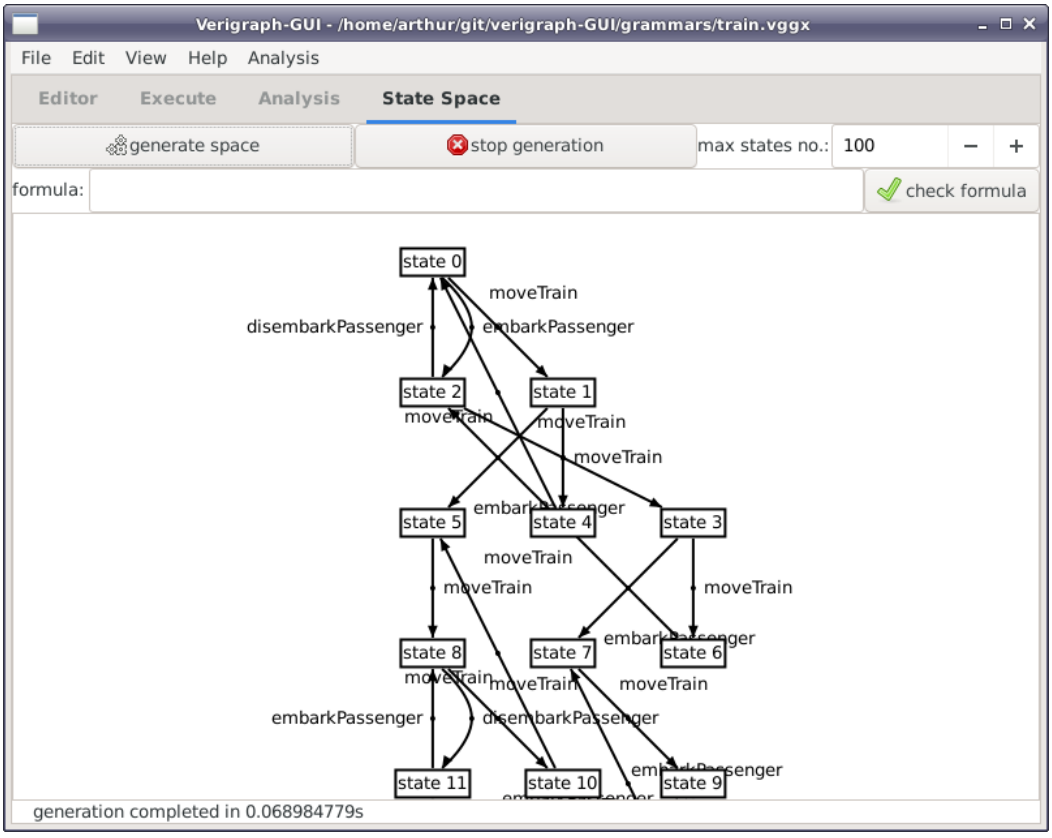


Figura 3. Leiaute gerado durante a construção do espaço de estados. O leiaute mostrado na Figura 2 é o resultado da organização deste. Nesta figura, os nodos não têm sombreamento porque a verificação não foi executada (foi somente executada a geração do espaço de estados).

Atualmente está em progresso a avaliação da performance do algoritmo de verificação de modelos através de *benchmarks*. Um fator importante a ser considerado para a performance do verificador de modelos é a geração do espaços de estados. Por conta de cada estado ser considerado uma classe de equivalência de grafos, a verificação de isomorfismo de grafos atualmente realizada é custosa e pretendemos propor melhorias nesse sentido. Pretendemos também:

- Melhorar a visualização do espaço de estados, pois atualmente não é possível visualizar qual classe de equivalência de grafos isomorfos um estado representa. Uma possibilidade é integrar o espaço de estados com o módulo de simulação, de forma que quando um estado seja selecionado, esse estado se torne o estado atual do simulador (similar ao sistema Groove [Rensink 2004]).
- Estender o verificador de modelos para aceitar outras lógica temporais, em particular LTL e CTL* [Huth and Ryan 2004]. Essas lógicas permitem expressar propriedades diferentes:

– Diferentemente de CTL, LTL não permite verificar pela existência de um caminho. LTL permite expressar propriedades sobre os caminhos de forma universal. Isso porque LTL não faz uso de quantificadores (A, E). No entanto, LTL permite selecionar um subconjunto de caminhos descrevendo-os com uma fórmula, algo que não é possível de fazer em CTL pois os operadores temporais em CTL são sempre associados a quantificadores.
– CTL* é uma lógica temporal que combina as expressividades de CTL e LTL. Isso é feito removendo as restrições de que todo operador temporal deve ser associado a um quantificador. O lado negativo da lógica CTL* é que sua verificação é mais custosa computacionalmente.

Referências

Azzi, G. G., Bezerra, J. S., Ribeiro, L., Costa, A., Rodrigues, L. M., and Machado, R. (2018). *The Verigraph System for Graph Transformation*, pages 160–178. Springer International Publishing, Cham.

Becker, T. R. (2014). VeriGraph: A Tool For Model Checking Graph Grammars. Monografia (Bacharel em Ciência da Computação), UFRGS (Universidade Federal do Rio Grande do Sul), Rio Grande do Sul, Brazil.

Clarke, E. M. (2008). *25 Years of Model Checking: History, Achievements, Perspectives*. Springer Berlin Heidelberg, Berlin, Heidelberg.

Ehrig, H., Ehrig, K., Prange, U., and Taentzer, G. (2006). *Fundamentals of Algebraic Graph Transformation (Monographs in Theoretical Computer Science. An EATCS Series)*. Springer-Verlag, Berlin, Heidelberg.

Fuchs, A. L., Machado, R., and Ribeiro, L. (2019). Verigraph-gui: A gui for the verigraph tool. In *Anais WEIT 2019*, pages 268–277. Universidade de Passo Fundo (UPF).

Huth, M. and Ryan, M. (2004). *Logic in computer science - modelling and reasoning about systems (2. ed.)*.

Rensink, A. (2004). The groove simulator: A tool for state space generation. In Pfaltz, J. L., Nagl, M., and Böhlen, B., editors, *Applications of Graph Transformations with Industrial Relevance*, pages 479–485, Berlin, Heidelberg. Springer Berlin Heidelberg.

An approach for consensual analysis on Typical Hesitant Fuzzy Sets via extended aggregations and fuzzy implications based on admissible orders

Monica Matzenauer[1], **Advisor Renata Reiser** [1], **Coadvisor Helida Santos**[2]

[1]Federal University of Pelotas (UFPel)
Pelotas, RS – Brazil

[2]Federal University of Rio Grande (FURG)
Rio Grande, RS – Brazil

{monica.matzenauer,reiser}@inf.ufpel.edu.br, helida@furg.br

Abstract. *Typical Hesitant Fuzzy Logic (THFL) is founded on the theory of Hesitant Fuzzy Sets, which consider as membership degrees the finite and non-empty subsets of the unit interval, called Typical Hesitant Fuzzy Elements (THFE). THFL provides the modelling for situations where there exists not only data uncertainty, but also indecision or hesitation among experts about the possible values for preferences regarding collections of objects. In order to reduce the information collapse for comparison and/or ranking of alternatives in the preference relationships, this thesis develops new ideas on THFL connectives, investigated under the scope of three admissible orders. In particular, properties of negations and aggregations are studied, as t-norms and OWA operators, with special interest in the axiomatic structures defining the implications and preserving their algebraic properties and representability. As the main contribution, we present a model that formally builds consensus measures on THFE through extended aggregation functions and fuzzy negation, using admissible orders for comparison and further, differentiating an analysis of consistency over preference matrices. Main theoretical results are submitted to multiple expert and mutiple criteria decision making problems.*

Keywords: *Typical hesitant fuzzy sets, Consensus measures, Fuzzy implications, Extended aggregation functions, Admissible orders.*

1. Introduction

Fuzzy Set Theory (FS), presented by [Zadeh 1965], has yielded several extensions over the years. Among the most relevant contributions in this research area, we highlight the following works:

- the extension known as Type-2 Fuzzy Sets (T2FS), introduced in [Zadeh 1971, Zadeh 1975], which considers the membership functions as FS on the unit interval $[0, 1]$;
- the Set-Valued Fuzzy Sets (SVFS), introduced by [Grattan-Guinness 1976] expressing the membership degrees as subsets of the unit interval $[0, 1]$;
- Atanassov's Intuitionistic Fuzzy Sets (IFS) in [Atanassov 1986], where the definition of a fuzzy set considers not only the membership function, but also its dual construction providing the non-membership degree; and

- Hesitant Fuzzy Sets (HFS), proposed by [Torra and Narukawa 2009] as another extension for fuzzy sets, in which the membership degree of an hesitant fuzzy set is also given as a subset of $[0, 1]$.

Note that despite the existence of different extensions in order to handle imprecise information, there are relationships among them as showed in [Bustince et al. 2016]. This work explored the inclusion relationships between some types of fuzzy sets and it also concluded that, in fact, the concepts of HFS and SVFS are equivalent. However, the results achieved in [Torra 2010] presented an explicit definition for the union and intersection operations on HFS, which was not the research focused in [Grattan-Guinness 1976].

Later, [Bedregal et al. 014a] noticed that in most applications, Typical Hesitant Fuzzy Elements (THFE) are used, i.e. finite and non-empty hesitant fuzzy degrees. Then, the notion of Typical Hesitant Fuzzy Logic (THFL) appears, which is based on Typical Hesitant Fuzzy Sets (THFS) conceived taking THFE as membership degrees.

Relevant research in decision making has been supported by the HFS theory, since it was introduced in 2010. See, for instance, the studies in [Zeng et al. 2020, Farhadinia et al. 2020, Rezaei and Rezaei 2020, Farhadinia and Xu 2020, Wang et al. 2021] considering the logical study of HFS. In particular, several weighted average (WA) and ordered weighted average (OWA)-like operators have been proposed to be used in multi-criteria and decision making (MCDM) problems dealing with multiple attributes and multiple specialists [Bedregal et al. 014a, Xia and Xu 2011, Zhu et al. 2012, Matzenauer et al. 2021a].

A frequent issue in the context of MCDM problems is that it is not always possible to find a consensus among a group of experts. So, it seems more appropriate to consider a set of possible values taking into account everyone's opinion. For instance, in order to provide a membership degree for an element of the universe, HFS can be useful to express this membership degree through a set of THFE, which will consider all the opinions given by the group of experts.

However, some research questions arise from this setting:

(i) How much do these elements agree with each other?
(ii) Is it possible to combine these elements into a single output?
(iii) Is the result reliable and does it reflect the opinions provided by the group?

The consolidated research on consensus measures provide relevant results contributing to answer all these questions, which have been applied in different contexts. In the current literature, we can find works on fields like consensual processes [Unzu and Vorsatz 2011], consensual measures and aggregations [Beliakov et al. 2014], majority decisions [Garcí a Lapresta and Llamazares 2010] and preference intensities group decision and negotiation [García-Lapresta and Llamazares 2001, Llamazares et al. 2013].

Apart from other results, we provide a general idea of up to what extent the expert inputs agree with one another based on our approach using the theory of THFS. In our proposal, we present a model that formally constructs consensus measures by means of aggregations functions, fuzzy implication-like functions and fuzzy negations, using admissible orders to compare the THFE, and also providing an analysis of consistency

on them. Our theoretical results are applied into a problem of decision making with multi-criteria illustrating our methodology to achieve consensus in a group of experts working with typical hesitant fuzzy sets.

2. Main contributions

As the main contribution, the thesis introduces a model to provide semantic interpretation for consensus setting on Typical Hesitant Fuzzy Sets, namely the $CC_{\mathbb{H}}$-Model: Consensus Measures on Typical Hesitant Fuzzy Sets (THFS) based on Extended Aggregation and Implication Operators.

The main properties proposed in the literature of consensus measures are studied here from the setting of inputs on the class of THFS defined over $\mathbb{H}$, which is the set of all finite and non-empty subsets of the unit interval $[0, 1]$, mainly according to the approach in [Beliakov et al. 2014]. However, the studies are restricted to the fact that the agreement among evaluations has the same relevance.

Going beyond, among several partial orders defined over $\mathbb{H}$, the proposal represents an extension that considers an admissible partial order $\leq_A$ on $\mathbb{H}$; based on an extended aggregation operator A, as a binary relation promoting comparison even between THFE with different cardinalities.

Main contributions achieved in the thesis are listed below:

(i) Starting with fuzzy extensions of consensus measures from U to $\mathbb{H}$ and taking into account so many distinct partial orders for THFS, this work introduces a new admissible order based on a hesitant aggregation function $\mathcal{A}$, refining not only the restricted consensual order $<_{RH}$ but many other ones, providing a comparison between HFS which do not have the same cardinalities;

(ii) This research considers the concepts of admissible orders obtained from hesitant aggregation functions and fuzzy negations, providing methods to generate comparisons (ordering) of typical hesitant fuzzy elements [Bustince et al. 2013, Miguel et al. 2016, Lima 2019];

(iii) Extension of the main concepts of fuzzy connectives (fuzzy negations, aggregation functions, t-norms and t-conorms, fuzzy implications) are considered, discussing their properties regarding admissible orders [Bustince et al. 2020, Matzenauer et al. 2021a];

(iv) The work also considers a formal definition of consensus measures extending this concept to the typical hesitant fuzzy sets. Actually, various applications can be found in the literature concerning consensus measures [Li et al. 2018, Rodríguez et al. 2018], and also an attempt of a formal mathematical definition was given in [Beliakov et al. 2014].

(v) This study explores methodologies based on the $CC_{\mathbb{H}}$-Model, a construction of consensus measures through different implications, exploring main properties of fuzzy implications which are required, regarding admissible/total orders on $\mathbb{H}$.

3. Objectives

The main objective of the thesis is to extend Beliakov's work by applying consensus measures to typical hesitant fuzzy sets based on aggregation functions and admissible orders.

The specific objectives are described as follows:

- Contribute for the study of hesitant fuzzy sets and admissible linear orders, considering their relevance in multi-valued fuzzy sets by allowing comparison and ordering relations;
- Collaborate to the study of hesitant fuzzy aggregators, exploring the main properties, analysing their relevance to consensus measurement methodologies and practical applications;
- Study fuzzy consensus measures and provide the application of related methodology in the decision making based on multi-criteria and -attribute from many specialists;
- Introduce the study of main classes of hesitant fuzzy implications, exploring the main properties and constructors of duality and conjugation;
- Propose the $CC_{\mathbb{H}}$-Model, based on an axiomatic definition of consensus measures consistent with studies of admissible linear orders in typical hesitant fuzzy sets.

4. Final Considerations

In the thesis, new ideas in THFE are investigated and developed under the scope of an arbitrary order, allowing the possibility of comparisons of THFE with different cardinalities. A class of admissible linear $\langle \mathbb{H}, \preceq_A^f \rangle$-orders is also presented, when A is an increasing aggregation function and f satisfies the injective-cardinality property. This admissible $\langle \mathbb{H}, \preceq_A^f \rangle$-orders is a family of total orders that refine the partial $\langle \mathbb{H}, \leq_{RH} \rangle$ -order. The theorem presented below, can be seen in with more details in the thesis and in [Matzenauer et al. 2021b].

Theorem 4.1 *Let $A^* : \mathbb{H} \to [0,1]$ be a function such that A^* is increasing w.r.t. $\leq_{RH}$, $A^*(\mathbf{0}_{\mathbb{H}}) = 0$ and $A^*(\mathbf{1}_{\mathbb{H}}) = 1$ and $f^* : \mathbb{H} \to \mathbb{R}$ be a function such that the property:*

$IC : f^*(X) = f^*(Y) \Rightarrow \#X = \#Y$ *(injective-cardinality property)*

is satisfied. The relation $\preceq_{A^*}^{f^*}$ on $\mathbb{H}$ defined by

$$X \preceq_{A*}^{f^*} Y \Leftrightarrow \begin{cases} X = Y, \text{ or} \\ A^*(X) < A^*(Y), \text{ or} \\ A^*(X) = A^*(Y) \text{ and } f^*(X) < f^*(Y), \end{cases} \tag{1}$$

is a total admissible order on $\mathbb{H}$ whenever, for each $n \in \mathbb{N}^+$, $A_n^* = A^* \restriction \mathbb{H}^{(n)}$ is injective.

Two other admissible linear orders are also considered: $\langle \mathbb{H}, \preceq_{Lex1} \rangle$- and $\langle \mathbb{H}, \preceq_{Lex2} \rangle$-orders, that allowed us to introduce a formal definition of hesitant fuzzy operators, such as $\langle \mathbb{H}, \preceq \rangle$-aggregation functions and $\langle \mathbb{H}, \preceq \rangle$-negations, including their respective important properties. Emphasizing as main results, we have the generation of $\langle \mathbb{H}, \preceq \rangle$-negations from fuzzy negations, also presenting some interesting examples.

The contextual theoretical research on the properties of $\langle \mathbb{H}, \preceq \rangle$-implications is related to the monotonicity analysis, which is restricted to the first place antitonicity and second place isotonicity, but also including the identity and exchange principles, the left and right boundary conditions, the contrapositive symmetry and ordering properties. Some additional examples are also presented, mainly related to natural negations obtained from $\langle \mathbb{H}, \preceq \rangle$-implication functions.

Based on such classes of $\langle \mathbb{H}, \preceq \rangle$-orders, expressions for the main examples of aggregation functions and fuzzy implications are presented. Furthermore, the representability of such operators is obtained from generation of $\langle \mathbb{H}, \preceq \rangle$-implications as an order-preserving structure of main implication properties. Besides, is introduced a formal definition for the representability of those negations, by constructing a method to obtain $\langle \mathbb{H}, \preceq \rangle$-implications from $\langle \mathbb{H}, \preceq \rangle$-aggregations.

Another relevant contribution illustrating our theoretical results is an algorithmic solution for an ME-MCDC problem, which used $\langle \mathbb{H}, \preceq \rangle$-operators and took into account the selection of a CIM-software. By applying the Łukasiewicz implication, the example reported an ME-MCDM problem in a CIM-application, which could be analysed from three distinct comparisons based on $\langle \mathbb{H}, \preceq \rangle$-operators.

The work also presents improved consensus-based procedures based on admissible $\langle \mathbb{H}, \preceq \rangle$-orders, handling Multi Expert-Multi Criteria Decision Making (ME-MCDM) problems and using consistent $\langle \mathbb{H}, \preceq \rangle$-preference relations (HFPR). At the first level, the consistence analysis considers the weak transitivity and ordinal consistency properties in $\langle \mathbb{H}, \preceq \rangle$-orders, also extending the notion of (restricted) $\max\text{-}\max$ and $\min\text{-}\max$ transitivity. Subject to such results on consistency analysis, normalised additive hesitant fuzzy preference relations introduce two strategies to obtain a consensus-based model.

Then, we formally defined the generalised notion of consensus measures from $([0,1], \leq)$ to a bounded poset $\mathbb{H} = \langle \mathbb{H}, \preceq \rangle$, also studying the corresponding extensions of aggregations, implications and fuzzy negations. As one of the main contribution, the $\mathcal{CC}_{\mathcal{AI}}$- and $\mathcal{CC}_{\min,\mathcal{I}}$-consensus models are presented as new methodologies of consensus preserving main properties in the context of Typical Hesitant Fuzzy Sets, by exploring properties of admissible $\langle \mathbb{H}, \preceq \rangle$-aggregation and admissible $\langle \mathbb{H}, \preceq \rangle$- implications.

This first consensus measures method $\mathcal{CC}_{\mathcal{AI}}$ is based on a typical hesitant extended aggregation function $\mathcal{A}$ and an $\langle \mathbb{H}, \preceq \rangle$-implication $\mathcal{I}$. The theorem presented below, can be seen with more details in the thesis and in [Matzenauer et al. 2021b].

Theorem 4.2 *Let $\mathcal{A}$ be an extended $\langle \mathbb{H}, \preceq \rangle$-aggregation function satisfying some aggregations properties and $\mathcal{I}$ be an $\langle \mathbb{H}, \preceq \rangle$-implication verifying some implications properties. Then the operator $\mathcal{CC}_{\mathcal{AI}} \colon \bigcup_{n=2}^{\infty} \mathbb{H}^n \to \mathbb{H}$ given by*

$$\mathcal{CC}_{\mathcal{AI}}(X_1, \ldots, X_n) \;=\; \mathcal{A}^n_{i,j=1,i\neq j}(\mathcal{I}(X_i, X_j)), \tag{2}$$

is an $\langle \mathbb{H}, \preceq, \mathbf{0}_{\mathbb{H}}, \mathbf{1}_{\mathbb{H}} \rangle$-valued consensus measure on $\mathbb{H}$,

This second method, $\mathcal{CC}_{\min,\mathcal{I}}$-consensus model, considers the minimum typical hesitant aggregation ($\mathcal{T}_{\min} = \min$) and an $\langle \mathbb{H}, \preceq \rangle$-implication $\mathcal{I}$, and is presented with more details in the thesis and in [Matzenauer et al. 2021b].

Theorem 4.3 *Let $\mathcal{A}$ be an extended $\langle \mathbb{H}, \preceq \rangle$-aggregation function satisfying some aggregations properties and $\mathcal{I}$ be an $\langle \mathbb{H}, \preceq \rangle$-implication verifying some implications properties. The operator $\mathcal{CC}_{\min,I} \colon \bigcup_{n=2}^{\infty} \mathbb{H}^n \to \mathbb{H}$ given by*

$$\mathcal{CC}_{\min,I}(X_1, \ldots, X_n) \;=\; \begin{cases} \mathcal{I}(X_1, X_2) \wedge \mathcal{I}(X_2, X_1), & \text{if } n = 2 \\ \mathcal{CC}_{\min,I}(X_1, \mathcal{CC}_{\min,I}(X_2, \ldots, X_n)), & \text{if } n > 2, \end{cases} \tag{3}$$

is an $\langle \mathbb{H}, \preceq \rangle$-valued consensus measure on $\mathbb{H}$.

5. Future works

Ongoing works are focusing on other classes of implications, such as (S, N)-implications [Zanotelli et al. 2020] and including the residuation principle related to R-implications and the left-continuity of t-norms, in the context of admissible $\langle \mathbb{H}, \preceq \rangle$-orders. And, in order to show the advantage of the proposed method, further work extends case studies in cloud computing [Schneider et al. 2020], for hesitant fuzzy environments, based on the theoretical results achieved in this step of the research on admissible linear orders.

We also intend to explore new group of strategies to obtain consensus measures, mainly connected to the class of operators, satisfying commutative, nondecreasing aggregations with $\mathbf{1}_{\mathbb{H}}$-annihilator.

6. Publications

This section reports the publication of the main results connected with the thesis.

1. MATZENAUER, M. L.; SANTOS, H.; BEDREGAL, B.; BUSTINCE, H.; REISER, R.. On Admissible Total Orders for Typical Hesitant Fuzzy Consensus Measures. in: International Journal of Intelligent Systems (IJIS, 2021), p. 1-23, 2021.
2. MATZENAUER, M. L.; REISER, R.; SANTOS, H.; BEDREGAL, B.; BUSTINCE,H.. Strategies on Admissible Total Orders Over Typical Hesitant Fuzzy Implications Applied to Decision Making Problems. In: International Journal of Intelligent Systems. (IJIS, 2021), p. 1-50, 2021.
3. MATZENAUER, M. L.; REISER, R.; SANTOS, H.; PINHEIRO, J.; BEDREGAL, B.. An Initial Study on Typical Hesitant (T,N)-Implication Functions. In: 18th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems. CCIS 1238, p. 747-760, 2020.
4. MATZENAUER, M. L.; REISER, R.; SANTOS, H.; BEDREGAL, B.. Typical Hesitant Fuzzy Sets: Evaluating Strategies in GDM Applying Consensus Measures. In: Conference of the International Fuzzy Systems Association and the European Society for Fuzzy Logic and Technology. (EUSFLAT 2019), 2019., 2019/08. Anais. Atlantis Press, 2019/08.
5. MATZENAUER, M. L.; REISER, R.; SANTOS, H.; BEDREGAL, B.; BUSTINCE, H.. Typical Hesitant Fuzzy Implications Functions. In: Workshop Escola de Informática Teórica (WEIT2019), 2019, Passo Fundo. Anais do Workshop Escola de Informática Teórica (WEIT2019). PF: ed.UFSM, 2019. v. 1. p. 222-230.

References

Atanassov, K. T. (1986). Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 20(1):87–96.

Bedregal, B., Reiser, R., Bustince, H., Lopez-Molina, C., and Torra, V. (2014a). Aggregation functions for typical hesitant fuzzy elements and the action of automorphisms. *Information Sciences*, 255:82 – 99.

Beliakov, G., Calvo, T., and James, S. (2014). Consensus measures constructed from aggregation functions and fuzzy implications. *Knowledge-Based Systems*, 55:1 – 8.

Bustince, H., Barrenechea, E., Pagola, M., Fernández, J., Xu, Z., Bedregal, B. R. C., Montero, J., Hagras, H., Herrera, F., and Baets, B. D. (2016). A historical account of types of fuzzy sets and their relationships. *IEEE Transactions Fuzzy Systems*, 24(1):179–194.

Bustince, H., Fernandez, J., Kolesárová, A., and Mesiar, R. (2013). Generation of linear orders for intervals by means of aggregation functions. *Fuzzy Sets and Systems*, 220:69 – 77. Theme: Aggregation functions.

Bustince, H., Marco-Detchart, C., Fernández, J., Wagner, C., Garibaldi, J. M., and Takác, Z. (2020). Similarity between interval-valued fuzzy sets taking into account the width of the intervals and admissible orders. *Fuzzy Sets Syst.*, 390:23–47.

Farhadinia, B., Aickelin, U., and Khorshidi, H. A. (2020). Uncertainty measures for probabilistic hesitant fuzzy sets in multiple criteria decision making. *Int. J. Intell. Syst.*, 35(11):1646–1679.

Farhadinia, B. and Xu, Z. (2020). A novel distance-based multiple attribute decision-making with hesitant fuzzy sets. *Soft Comput.*, 24(7):5005–5017.

Garcí a Lapresta, J. L. and Llamazares, B. (2010). Preference intensities and majority decisions based on difference of support between alternatives. *Group Decision and Negotiation*, 19(6):527–542.

García-Lapresta, J. and Llamazares, B. (2001). Majority decisions based on difference of votes. *Journal of Mathematical Economics*, 35(3):463–481. cited By 41.

Grattan-Guinness, I. (1976). Fuzzy membership mapped onto intervals and many-valued quantities. *Mathematical Logic Quarterly*, 22(1):149–160.

Li, C., Rodríguez, R. M., Martínez, L., Dong, Y., and Herrera, F. (2018). Personalized individual semantics based on consistency in hesitant linguistic group decision making with comparative linguistic expressions. *Knowledge-Based Systems*, 145:156–165.

Lima, A. A. d. (2019). *Multidimensional Fuzzy Sets*. PhD thesis, Federal University of Rio Grande do Norte, Center for Exact and Earth Sciences, Department of Informatics and Applied Mathematics, Graduate Program in Systems and Computing, Natal, Rio Grande do Norte, Brazil.

Llamazares, B., Pérez-Asurmendi, P., and García-Lapresta, J. (2013). Collective transitivity in majorities based on difference in support. *Fuzzy Sets and Systems*, 216:3–15. cited By 9.

Matzenauer, M., Reiser, R., Santos, H., Bedregal, B., and Bustince, H. (2021a). Strategies on admissible total orders over typical hesitant fuzzy implications applied to decision making problems. *International Journal of Intelligent Systems*, pages 1–50. . In press.

Matzenauer, M., Santos, H., Bedregal, B., Bustince, H., and Reiser, R. (2021b). On admissible total orders for typical hesitant fuzzy consensus measures. *International Journal of Intelligent Systems*, pages 1–23.

Miguel, L. D., Bustince, H., Fernandez, J., Induráin, E., Kolesárová, A., and Mesiar, R. (2016). Construction of admissible linear orders for interval-valued atanassov

intuitionistic fuzzy sets with an application to decision making. *Information Fusion*, 27:189 – 197.

Rezaei, K. and Rezaei, H. (2020). New distance and similarity measures for hesitant fuzzy sets and their application in hierarchical clustering. *J. Intell. Fuzzy Syst.*, 39(3):4349–4360.

Rodríguez, R. M., Xu, Y., Martínez-López, L., and Herrera, F. (2018). Exploring consistency for hesitant preference relations in decision making: Discussing concepts, meaning and taxonomy. *Multiple-Valued Logic and Soft Computing*, 30(2-3):129–154.

Schneider, G. B., Moura, B. M. P., Yamin, A. C., and Reiser, R. H. S. (2020). Int-FLBCC: Model for load balancing in cloud computing using fuzzy logic type-2 and admissible orders. *RITA*, 27(3):102–117.

Torra, V. (2010). Hesitant fuzzy sets. *Int. Journal of Intelligent Systems*, 25:529–539.

Torra, V. and Narukawa, Y. (2009). On hesitant fuzzy sets and decision. In *FUZZ-IEEE 2009, IEEE Int. Conference on Fuzzy Systems, Korea, 2009, Proceedings*, pages 1378–1382.

Unzu, J. A. and Vorsatz, M. (2011). Measuring consensus concepts, comparisons, and properties. In *Consensual Processes*, pages 195–211. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wang, Z., Wu, J., Liu, X., and Garg, H. (2021). New framework for fcms using dual hesitant fuzzy sets with an analysis of risk factors in emergency event. *Int. J. Comput. Intell. Syst.*, 14(1):67–78.

Xia, M. and Xu, Z. (2011). Hesitant fuzzy information aggregation in decision making. *International Journal of Approximate Reasoning*, 52(3):395 – 407.

Zadeh, L. (1971). Quantitative fuzzy semantics. *Information Sciences*, 3:159 – 176.

Zadeh, L. (1975). The concept of a linguistic variable and its application to approximate reasoning – I. *Information Sciences*, 8(3):199 – 249.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control*, 8(3):338–353.

Zanotelli, R. M., Reiser, R., and Bedregal, B. R. C. (2020). n-dimensional (S, N)-implications. *Int J Approx Reason*, 126:1–26.

Zeng, W., Ma, R., Yin, Q., and Xu, Z. (2020). Similarity measure of hesitant fuzzy sets based on implication function and clustering analysis. *IEEE Access*, 8:119995–120008.

Zhu, B., Xu, Z., and Xia, M. (2012). Hesitant fuzzy geometric bonferroni means. *Information Sciences*, 205:72 – 85.

n-Dimensional Fuzzy Implications: Analytical, Algebraic and Applicational Approaches

Rosana Medina Zanotelli[1],
Advisor Renata H. Sander Reiser[1], **Coadvisor Benjamín René C. Bedregal**[2]

[1]Center of Tecnological Development – CDTec
Federal University of Pelotas – (UFPel)
CEP 96010-610 – Pelotas – RS – Brazil

[2]Department of Informatics and Applied Mathematics – DIMAp
Federal University of Rio Grande do Norte – (UFRN)
CEP 59072-970 – Natal – RN – Brazil

{rzanotelli,reiser}@inf.ufpel.edu.br,

bedregal@dimap.ufrn.br

Abstract. *The study of n-dimensional fuzzy logic contributes to overcome the insufficiency of traditional FL in modeling imperfect and imprecise information coming from different experts. Based on representability, we extend results from fuzzy connectives to n-dimensional approach. This research on n-dimensional fuzzy implications (n-DI) pass through the next steps: (i) analytical studies;(ii) algebraic aspects;(iii) n-dimensional approach of fuzzy implication classes represented by fuzzy connectives as (S, N)-implications and QL-implications; (iv) studies of n-dimensional R-implications (n-DRI); (v) constructive method obtaining n-DRI based on n-dimensional aggregation operators and (vi) an introductory study considering an n-DI in modeling approximate reasoning.*

1. Introduction

The n-dimensional interval studies provide a new possibility to model not only the imprecision in the calculations, but also to describe the (ordered and possible repeated) uncertainty opinion provided by many specialists related to their preference relations faced to multi-criteria data and decision making systems.

In order to contribute with the theoretical research on n-dimensional intervals, this thesis aims to study of n-dimensional fuzzy sets (n-FS), not only as an extension of fuzzy sets, but also investigating their properties, exploring main classes of n-dimensional aggregations (n-DA) and also reporting two applications.

More specifically, this research proposes a theoretical study of n-DI considering the following approaches:
(i) the analytical study, defining n-DI, presenting the most desirable properties;
(ii) the algebraic aspects related to the left and right continuity n-dimensional t-norms and properties of natural n-dimensional fuzzy negations;
(iii) generation of n-DI by analysing their representability from fuzzy implications, also investigating the conjugation operation by action of n-dimensional automorphisms;
(iv) the n-DI classes explicitly/implicitly represented by fuzzy connectives, such as

(S, N)-implications and QL-implications;
(v) the study of n-DRI, analyzing the Residuation Principle and characterizing n-DRI based on continuous n-dimensional Triangular Norms (n-DT);
(vi) providing constructive methods to obtain n-DRI via n-dimensional aggregations.

Thus, these potential results illustrate applications of n-DRI in two fields:
(i) in solving problems of Multiple Criteria and Decision Making (MCDM), considering multiple alternatives in the selection of Computer-Integrated Manufacturing (CIM) software; and,
(ii) in modelling the approximate reasoning (AR) of fuzzy inference schemes, considering n-dimensional extension of intervals, fuzzy statements and IF-THEN rules.

1.1. Main Motivations

This large field of theoretical research and technological applications related to n-dimensional fuzzy set theory motivates the extensive studies developed in this thesis. In particular, it is reported in [Zanotelli 2020] that the class of fuzzy implications plays an important role in inference fuzzy systems, since they are applied in fuzzy control, imprecision analysis from natural language and soft-computing techniques. They provide the mathematical basis for the generation of powerful techniques for AR involving uncertain, vague and nebulous processes. Such theoretical results underlie applications in areas as pattern recognition, image processing, data mining and mathematical morphology, modeling the relevance of frequency in repetitive data information.

In addition, the extension of fuzzy implication arises in many situations in various branches of computer science and technological development, see, e.g., controlling a mobile robot; providing prediction of the survival time of myeloma patients in hybrid systems for genetic algorithms; modelling expert system for shopping via web; developing systems for analysis and estimation of the survival time of wireless sensor networks.

As another relevant motivation, we consider to consolidate projects connected to Laboratory of Ubiquitous and Parallel Systems (LUPS), Formal Methods and Mathematical Fundamentals of Computer Science (MFFMCC), LoLiTA (UFRN/Brazil) and GIARA (Navarra/Spain), research groups in multi-valued fuzzy logics showing the conditions under which the main properties of fuzzy implications are preserved by representable n-DI.

1.2. Contextual approach of the research proposal and related works

In 1983, the Atanassov's intuitionistic fuzzy sets [Atanassov 1986] (AFS) were defined as a special type of Type-2 Fuzzy Set (T2FSs), in such a way that each element is associated with a pair of real numbers, representing the membership and non-membership degrees, which is not necessarily defined by the complementary relationship.

Basic ideas of interval-valued fuzzy sets (IVFS) were simultaneously defined by Zadeh [Zadeh 1975] and Sambuc [Sambuc 1975]. Relevant works in IVFS were preliminarily studied by Mizumoto and Tanaka [Mizumoto and Tanaka 1976] also considering Dubois and Prade's [Dubois and Prade 2005] and [Deschrijver and Kerre 2005] research. And, the notion of interval-valued Atanassov intuitionistic fuzzy sets (IVAFS) is introduces in [Atanassov and Gargov 1998], formalizing AFS in the field of T2FSs.

Vicenç Torra proposes hesitant fuzzy sets (HFS) as an extension of the fuzzy sets in [Torra 2010] with the goal of considering fuzzy sets where the membership degree can

be expressed as a set of values, enabling the description of distinct opinions of several experts. Later, in [Bedregal et al. 2014] the notion of typical hesitant fuzzy sets (THFS) considers only finite and nonempty hesitant fuzzy values.

In 2010, n-dimensional fuzzy sets (n-DFS) were conceived by [Shang et al. 2010] as an extension of fuzzy sets, including IVFS and IVAFS. As the main difference related to HFS, n-DFS allow you to explore repetition of ordered elements on the membership degrees. So, as the main idea, n-DFS consider several uncertainty levels in the memberships degrees to model different membership degrees which are obtained as a consensus to unify or even aggregate these values [Bedregal et al. 2011].

See Table 1 summarizing the main characteristics of the selected research papers on $L_n(U)$, reporting techniques, main contributions and the applied approach areas.

1.3. Methodological structure of the doctoral thesis

This thesis is organized in eight Chapters, starting from preliminary aspects of fuzzy logic and evolving to more specific ones in order to develop the research.

In Chapter 2, the contributions of the type-n fuzzy sets to the systems development area are treated, a comparative and hierarchical analysis of the fuzzy sets is made showing the different interpretations of information and a literature review reporting relevant studies on the n-dimensional intervals.

The fundamental concepts of fuzzy logic is shown in Chapter 3, starting with automorphisms, connectives in fuzzy logic such as negations, triangular norms and conorms and implications with the main classes.

Fundamental concepts of n-dimensional fuzzy logic (n-DFL), such as order relations, the main properties of continuity of functions, automorphism, and an extension of the fuzzy sets are reported in Chapter 4.

In sequence, in Chapter 5, the n-dimensional fuzzy implications are reported, showing the main properties as well as representability. And finally, the expansion of the classes (S, N)-implication and QL-implication.

In Chapter 6, it deals with the extension of residual fuzzy implication, showing its main properties and also the action of automorphism in n-DRI. We present the characterization of n-DRI. Still how to obtain from n-dimensional aggregation operators an n-DRI and finally the extension of a fuzzy application to n-dimensional fuzzy.

Focusing on the R-implications, the discussions extending the residuation property based on Moore-metric, continuity and monotonicity properties with respected to the usual partial orders on $L_n(U)$ are considered in Chapter 7. Extending results from interval-valued fuzzy set theory to n-dimensional simplex, the class of n-DRI is constructed based on n-DA aggregation operators, given as the minimum operator and left-continuous t-norms. Some topics refer to the residuation property, conjugate operators, characterization of n-DRI, application of of n-DA to obtain n-DRI and main properties preserved by such methodology. Some basic concepts of AR are extended to the n-dimensional fuzzy approach, showing that n-dimensional fuzzy deduction rules generalize fuzzy rules of classical logic.

Thus, in conclusion, final considerations are presented, highlighting the original-

Table 1. Related papers on n-dimensional fuzzy sets

	Technique	Contribution	Class
1	Study on n-Dimensional R-implications [Zanotelli et al. 2021]	Study of the properties that characterize the class of R-**implications on** $L_n(U)$.	TR[1] DMP[2]
2	Conjuntos Fuzzy Multidimensionais [Lima 2019]	Presents the concept of **multidimensional fuzzy sets** as a generalization of n-DFS.	TR
3	n-Dimensional Interval Uninorms [Mezzomo et al. 2019]	Presents conjugate based on n-**dimensional interval uninorms** preserving their main properties.	TR
4	n-Dimensional Intervals and Fuzzy S-implications [Zanotelli et al. 2020]	Study main properties characterizing the class of S-**implications on** $L_n(U)$.	TR
5	Towards the study of main properties of n-Dimensional QL-implicators [Zanotelli et al. 2018]	Introduces the concept of n-**dimensional** QL-**implicators** considering duality and conjugation operators.	TR
6	Equilibrium Point of Representable Moore Continuous n-Dimensional Interval Fuzzy Negations [Mezzomo et al. 2018a]	Studies the class of representable (Moore-continuous) n-**dimensional interval fuzzy negations**.	TR
7	Moore Continuous n-Dimensional Interval Fuzzy Negations [Mezzomo et al. 2018b]	Characterizes the notion of (continuous) n-dimensional interval Moore-metric and n-**dimensional interval fuzzy negations**.	TR
8	n-Dimensional Fuzzy Negations [Bedregal et al. 2018]	Investigates an extension of n-representable fuzzy **negations on** $L_n(U)$ which are continuous and monotone.	MEDM[3]
9	Natural n-dimensional fuzzy negations for n-dimensional t-norms and t-conorms [Mezzomo et al. 2017]	Considers n-dimensional fuzzy negations studying n-**dimensional triangular norms and triangular conorms**.	TR
10	An algorithm for group decision making using n-dimensional fuzzy sets, admissible orders and OWA operators [De Miguel et al. 2017]	Introduces the concept of **admissible order for** n-**dimensional fuzzy set** as well as a construction method for these orders.	GDMP[4]
11	On n-dimensional strict fuzzy negations [Mezzomo et al. 2016]	Investigate the class of representable n-**dimensional strict fuzzy negations**.	TR
12	Weighted Average Operators Generated by n-dimensional Overlaps and an Application in Decision Making [Silva et al. 2015]	Provides a way to generate a class of **weighted average operator from** n-**dimensional** overlap functions and aggregation functions.	MAGDMP [5]
13	A class of fuzzy multisets with a fixed number of memberships [Bedregal et al. 2012]	Define a generalization of Atanassov's operators for n-dimensional fuzzy values (called n-dimensional intervals).	DMP
14	A Characterization Theorem for t-Representable n-Dimensional Triangular Norms [Bedregal et al. 2011]	Generalize the notion of t-representability for n-**dimensional t-norms** and provide a characterization theorem for that class of n-dimensional t-norms.	TR
15	The n-dimensional fuzzy sets and Zadeh fuzzy sets based on the finite valued fuzzy sets [Shang et al. 2010]	Defines cut sets on n-DFS studying the **decomposition and representation theorems**.	TR

[1]Theoretical Research [2]Decision Making Problem [3]Multi-Expert Decision Making
[4]Group Decision Making Problems [5]Multiple Attribute Group Decision Making Problems

ity and relevance of the thesis, the main contributions, the publications made during the doctorate course and the possibilities for future work to be developed.

2. n-Dimensional Fuzzy Logic: Fundamental Concepts

An n-dimensional fuzzy set considers tuples of size n whose components are valued in $U = [0;1]$ and ordered in increasing form, called n-dimensional intervals. Generally, these sets contribute in modeling situations involving decision-making where, each n-dimensional interval represents the opinion of n specialists on the degree to which an alternative meets a given criterion or attribute for this problem. The membership values of n-DFS are n-tuples of real numbers in $U = [0,1]$, called n-dimensional intervals, which are ordered in an increasing order. Formally, let $\chi \neq \emptyset$ and $\mathbb{N}_n = \{1, 2, \ldots, n\}$. By [Shang et al. 2010], an n-dimensional fuzzy set $A_{L_n(U)}$ is expressed as

$$A_{L_n(U)} = \{(\mathrm{x}, \mu_{A_{L_n(U)}}(\mathrm{x})) \colon \forall \mathrm{x} \in \chi\}, \tag{1}$$

where $\forall \mathrm{x} \in \chi$, $\mu_{A_{L_n(U)}}(\mathrm{x}) = (x_1, x_2, \ldots, x_n)$ such that $x_1 \leq x_2 \leq \ldots \leq x_n$. So, an n-dimensional fuzzy set B over χ can also be given as $B_{L_n(U)} = \{(\mathrm{x}, \mu_{B1}(\mathrm{x}), \ldots, \mu_{Bn}(\mathrm{x})) : \mathrm{x} \in \chi\}$, where all membership functions of B, denoted as $\mu_{Bi} : \chi \to U$, $\forall i \in \mathbb{N}_n$ verifies the condition $\mu_{B1}(\mathrm{x}) \leq \ldots \leq \mu_{Bn}(\mathrm{x})$, $\forall \mathrm{x} \in \chi$. The n-dimensional upper simplex is given as $L_n(U) = \{\mathbf{x} = (x_1, \ldots, x_n) \in U^n : x_1 \leq \ldots \leq x_n\}$, and an element $\mathbf{x} \in L_n(U)$ is called n-dimensional interval.

2.1. Improving the residuum analysis on $L_n(U)$

The following results show the necessary and sufficient conditions under which the pair of functions $(\mathcal{I}_\mathcal{T}, \mathcal{T})$ verifies the residuation property on $\mathcal{L}_n(U)$.

Theorem 1 *Let $\mathcal{T}$ be an n-DT. The following statements are equivalent:*
1. *$\mathcal{T}$ safisties the left-continuity property: $\mathcal{LC}$: $\lim_{n\to\infty} \mathcal{T}(\mathbf{x}, \mathbf{y}_n) = \mathcal{T}(\mathbf{x}, \lim_{n\to\infty} \mathbf{y}_n)$;*
2. *$(\mathcal{T}, \mathcal{I}_\mathcal{T})$ is an adjoint pair, verifying the residuation property: $\mathcal{RP}$: $\mathcal{T}(\mathbf{x}, \mathbf{z}) \leq \mathbf{y} \Leftrightarrow \mathcal{I}_\mathcal{T}(\mathbf{x}, \mathbf{y}) \geq \mathbf{z}$;* ***3***. *$\mathcal{I}_\mathcal{T}(\mathbf{x}, \mathbf{y}) = \max\{\mathbf{z} \in L_n(U) \colon \mathcal{T}(\mathbf{x}, \mathbf{z}) \leq \mathbf{y}\}$.*

The characterization of n-DRI is formalized from left-continuous n-DT considering the method of obtaining n-DT from n-dimensional fuzzy implications by a residuation principle. Since each $\mathcal{I} \in \mathcal{I}(L_n(U))$ verifies the right boundary condition, meaning that $\mathcal{I}(\mathbf{x}, /1/) = /1/$, the function $\mathcal{T}_\mathcal{I} : (L_n(U))^2 \to L_n(U)$,

$$\mathcal{T}_\mathcal{I}(\mathbf{x}, \mathbf{y}) = inf\{\mathbf{t} \in L_n(U) \colon \mathcal{I}(\mathbf{x}, \mathbf{t}) \geq \mathbf{y}\}, \tag{2}$$

is a well-defined function on $L_n(U)$.

Proposition 1 *For $\mathcal{I} \in \mathcal{I}(L_n(U))$ the following statements are equivalent:*
1. *$\mathcal{I}$ is right-continuous w.r.t. the second variable.*
2. *$(\mathcal{T}_\mathcal{I}, \mathcal{I})$ is an adjoint pair.*
3. *The infimum in Eq.(2) is the minimum, i.e., $\mathcal{T}_\mathcal{I}(\mathbf{x}, \mathbf{y}) = \min\{\mathbf{t} \in L_n(U) \colon \mathcal{I}(\mathbf{x}, \mathbf{t}) \geq \mathbf{y}\}$, where the right side exists for all $\mathbf{x}, \mathbf{y} \in L_n(U)$.*

Theorem 2 *If a function $\mathcal{I} : (L_n(U))^2 \to L_n(U)$ satisfies $\mathcal{I}2$, $\mathcal{I}5$, $\mathcal{I}13$ and verifies the right-continuity w.r.t. the second variable, then $\mathcal{T}_\mathcal{I}$ is a left-continuous n-DT. Moreover $\mathcal{I} = \mathcal{I}_{\mathcal{T}_\mathcal{I}}$, meaning that $\mathcal{I}(\mathbf{x}, \mathbf{y}) = \max\{\mathbf{t} \in L_n(U) \colon \mathcal{T}_\mathcal{I}(\mathbf{x}, \mathbf{t}) \leq \mathbf{y}\}$.*

And, in the next results, n-DRI can be obtained from n-DA operators.

Proposition 2 *Let* $T_1, \ldots, T_n : U^2 \to U$ *be left-continuous t-norms such that* $T_1 \leq ... \leq T_n$. *Then, for all* $\mathbf{x} = (x_1, ..., x_n)$, $\mathbf{y} = (y_1, ..., y_n) \in L_n(U)$ *the function* $\mathcal{T}_{T_1,...,T_n} : (L_n(U))^2 \to L_n(U)$ *given as*

$$\mathcal{T}_{T_1...T_n}(\mathbf{x}, \mathbf{y}) = \left(\min_{i=1}^{n} T_1(x_{n-i+1}, y_i), \min_{i=2}^{n} T_2(x_{n-i+2}, y_i), \ldots, T_n(x_n, y_n) \right), \quad (3)$$

verifies the t-subnorm conditions and $\mathcal{LC}$ *and* $\mathcal{RP}$ *properties on* $L_n(U)$.

The characterization of an $\mathcal{I}_{\mathcal{T}_{T_1,...,T_n}}$ operator is given in the following theorem.

Theorem 3 *Let* $T_1, \ldots, T_n : U^2 \to U$ *be left-continuous t-norms on* U *such that* $T_1 \leq \ldots \leq T_n$ *and* $I_1, \ldots, I_n$ *be their corresponding residual implications. Then the operator* $\mathcal{I}_{I_1,...,I_n} = \mathcal{I}_{\mathcal{T}_{T_1,...,T_n}}$ *where* $\mathcal{I}_{I_1,...,I_n} : (L_n(U))^2 \to L_n(U)$ *defined as*

$$\mathcal{I}_{I_1,...,I_n}(\mathbf{x}, \mathbf{y}) = \left(\min_{i=1}^{n} I_i(x_i, y_i), \min_{i=2}^{n} I_i(x_i, y_i), \ldots, I_n(x_n, y_n) \right) = \left(\min_{i=k}^{n} I_i(x_i, y_i) \right)_{k \in \mathbb{N}_n^*}.$$

Corollary 1 *Let* $T_1, \ldots, T_n : U^2 \to U$ *be left-continuous t-norms on* U *such that* $T_1 \leq \ldots \leq T_n$ *and* $I_1, \ldots, I_n$ *be their corresponding residual implications. Then* $(\mathcal{I}_{I_1,...,I_n}, \mathcal{T}_{T_1...T_n})$ *is an adjoint pair on* $L_n(U)$.

See now, in the following, a constructive method of an n-DI.

Proposition 3 *Let* $I_1, \ldots, I_n : U^2 \to U$ *be implications. Then, function* $\mathcal{I}_{I_1,...,I_n} : (L_n(U))^n \to L_n(U)$ *is an* n*-DI:*

$$\mathcal{I}_{I_1,...,I_n}(\mathbf{x}, \mathbf{y}) = \left(\min_{i=1}^{n} I_i(x_i, y_i), \min_{i=2}^{n} I_i(x_i, y_i), \ldots, I_n(x_n, y_n) \right), \quad (4)$$

3. Relevance and originality of the thesis

The study of n-dimensional intervals provide a new possibility to model not only the imprecision in the calculations, but also to describe the (possible repeated) uncertainty opinion provided by many specialists related to their preference relations faced to multi-criteria data. In these contexts, n-DFL approach contributed for the development of applications based on MEDM problems.

Such ordered structures are extensions of FS as old as HFS but less explored, in theoretical and applied research. Thus, the new investigations promoted by this thesis research intended to consolidate this extension of FL in both senses:

- theoretical foundations of main results in n-DFL,
- main results considering the extended study of n-DI, their main properties and classes contributing to explore other research fields taking advantages of logical structure of n-DFL: (i) formalizing IF-THEN rules of approximate reasoning; (ii) structuring new methods to measure similarity and consensus n-DFS; (iii) extracting parameters as correlation coefficient and entropy supporting the data analysis of applications.

And, see in [Zanotelli et al. 2020] partial relevant thesis results.

4. Conclusion

As a major contribution, the consolidating the research of fuzzy implications on $L_n(U)$ is achieved, considering: (i) analytical studies, defining n-DI, presenting the most desirable properties as neutrality, ordering, (contra-)symmetry, exchange and identity principles, and also discussing their interrelationships and exemplifications; (ii) algebraic aspects mainly related to left- and right-continuity of representable n-dimensional fuzzy

t-norms and the generation of n-DI from existing fuzzy implications; (iii) n-dimensional approach of fuzzy implication classes explicitly represented by fuzzy connectives, as (S, N)-implications and QL-implications; (iv) prospective studies of n-DRI, analyzing extended conditions to verify the residuation principle and their characterization based on n-dimensional t-norms and n-DRI; (v) constructive method obtaining n-DRI based on n-dimensional aggregation operators, presenting an exemplification in the solution of a problem involving CIM-MCDM, based on Łukasiewicz n-DRI; and also including (vi) an introductory study considering an n-DI in modeling AR of inference schemes, dealing with the effecting role of such connectives in based-rule fuzzy systems. In addition, taking into account the action of automorphism and fuzzy negations on $L_n(U)$, conjugate and dual operators of n-DI can be respectively obtained, preserving algebraic and analytical properties.

Further work intends to characterize other classes of n-DI, considering their application in the AR models based on inference schemes of deductive systems and construction of admissible linear orders providing new analysis of main properties of fuzzy implications on $L_n(U)$. To sum up, the methodology considered to achieve the main results of this thesis can be extended to multi-dimensional fuzzy interval.

References

Atanassov, K. and Gargov, G. (1998). Elements of intuitionistic fuzzy logic. part i. *Fuzzy Sets and Systems*, 95(1):39–52.

Atanassov, K. T. (1986). Intuitionistic fuzzy sets. *Fuzzy Sets and Systems*, 20(1):87–96.

Bedregal, B., Beliakov, G., Bustince, H., Calvo, T., Fernández, J., and Mesiar, R. (2011). A characterization theorem for t-representable n-dimensional triangular norms. In *Eurofuse 2011*, pages 103–112. Springer.

Bedregal, B., Beliakov, G., Bustince, H., Calvo, T., Mesiar, R., and Paternain, D. (2012). A class of fuzzy multisets with a fixed number of memberships. *Information Sciences*, 189:1–17.

Bedregal, B., Mezzomo, I., and Reiser, R. H. S. (2018). n-dimensional fuzzy negations. *IEEE Transactions on Fuzzy Systems*, 26(6):3660–3672.

Bedregal, B., Reiser, R., Bustince, H., Lopez-Molina, C., and Torra, V. (2014). Aggregation functions for typical hesitant fuzzy elements and the action of automorphisms. *Information Sciences*, 255:82–99.

De Miguel, L., Sesma-Sara, M., Elkano, M., Asiain, M., and Bustince, H. (2017). An algorithm for group decision making using n-dimensional fuzzy sets, admissible orders and owa operators. *Information Fusion*, 37:126–131.

Deschrijver, G. and Kerre, E. E. (2005). Implicators based on binary aggregation operators in interval-valued fuzzy set theory. *Fuzzy Sets and Systems*, 153(2):229–248.

Dubois, D. and Prade, H. (2005). Interval-valued fuzzy sets, possibility theory and imprecise probability. In *EUSFLAT Conf.*, pages 314–319.

Lima, A. A. d. (2019). *Conjuntos Fuzzy Multidimensionais*. PhD thesis, UFRN, Dep. of Infor. and Applied Mathematics, Grad. Prog. in Systems and Computing, Natal, RN, Brazil.

Mezzomo, I., Bedregal, B., and Milfont, T. (2018a). Equilibrium point of representable moore continuous n-dimensional interval fuzzy negations. In *North American Fuzzy Information Processing Society Annual Conf.*, pages 265–277. Springer.

Mezzomo, I., Bedregal, B., and Milfont, T. (2018b). Moore continuous n-dimensional interval fuzzy negations. In *2018 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

Mezzomo, I., Bedregal, B., Milfont, T., da Cruz Asmus, T., and Bustince, H. (2019). n-dimensional interval uninorms. In *2019 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6. IEEE.

Mezzomo, I., Bedregal, B., and Reiser, R. (2017). Natural n-dimensional fuzzy negations for n-dimensional t-norms and t-conorms. In *2017 IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE)*, pages 1–6.

Mezzomo, I., Bedregal, B. C., Reiser, R. H., Bustince, H., and Paternain, D. (2016). On n-dimensional strict fuzzy negations. In *Fuzzy Systems (FUZZ-IEEE), 2016 IEEE Int. Conf. on*, pages 301–307. IEEE.

Mizumoto, M. and Tanaka, K. (1976). Some properties of fuzzy sets of type 2. *Information and Control*, 31(4):312–340.

Sambuc, R. (1975). *Fonctions and Floues: Application a l'aide au Diagnostic en Pathologie Thyroidienne*. Faculté de Médecine de Marseille.

Shang, Y.-g., Yuan, X.-h., and Lee, E. S. (2010). The n-dimensional fuzzy sets and zadeh fuzzy sets based on the finite valued fuzzy sets. *Computers & Mathematics with Applications*, 60(3):442–463.

Silva, I., Bedregal, B., and Bustince, H. (2015). Weighted average operators generated by n-dimensional overlaps and an application in decision making. In *16th World Congress of the International Fuzzy Systems Association,(Eindhoven, The Netherlands)*, pages 1473–1478.

Torra, V. (2010). Hesitant fuzzy sets. *Intl Journal of Int Systems*, 25(6):529–539.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning—i. *Information sciences*, 8(3):199–249.

Zanotelli, R., Reiser, R., and Bedregal, B. (2018). Towards the study of main properties of n-dimensional ql-implicators. In *SBMAC*, pages 1–8. Federal University of Ceará.

Zanotelli, R. M. (2020). *n-Dimensional Fuzzy Implications: Analytical, Algebraic and Applicational Approaches*. PhD thesis, UFPEL, CDTec, Pelotas, RS, Brazil.

Zanotelli, R. M., Reiser, R., and Bedregal, B. R. C. (2020). n-dimensional (S, N)-implications. *Int. J. Approx. Reason.*, 126:1–26.

Zanotelli, R. M., Reiser, R., and Bedregal, B. R. C. (2021). On the residuation principle of n-dimensional r-implications. *Int. J. Soft Computing*, 1:1–30. under revision.

Uma Tradução de Redes de Petri Fuzzy Generalizadas para Gramáticas de Grafos com Atributos*

Júlia Krüger Vieira[1], **Luciana Foss**[1], **Simone André da Costa Cavalheiro**[1]

[1]Centro de Desenvolvimento Tecnológico
Programa de Pós-Graduação em Computação
Universidade Federal de Pelotas (UFPel) – Pelotas, RS – Brasil

{jkvieira,lfoss,simone.costa}@inf.ufpel.edu.br

Abstract. *In this paper, a method for translating a Generalized Fuzzy Petri Net to an Attributed Graph Grammar is presented. With this translation, the aim is to obtain a generic model to specify fuzzy systems which provides computational tools that allow to carry out property verifications and theorem proofs.*

Resumo. *Neste artigo é apresentado um método para traduzir uma Rede de Petri Fuzzy Generalizada para uma Gramática de Grafos com Atributos. Com esta tradução visa-se obter um modelo genérico para especificar sistemas fuzzy o qual disponibiliza ferramentas computacionais que permitem realizar verificações de propriedades e provas de teoremas.*

1. Introdução

No projeto descrito na dissertação apresentou-se uma tradução entre modelos partindo de um tipo de Rede de Petri Fuzzy (RPF) chamado de Generalizada (RPFG) para se obter uma Gramática de Grafos com atributos (GGA) com o intuito de fornecer um novo método de especificação para sistemas fuzzy. Nesta tradução, obtém-se uma GGA Fuzzy que descreve o mesmo sistema de produções fuzzy que a RPFG modela.

Dado que existem poucas ferramentas que permitem a análise de propriedades para sistemas modelados por RPF – são exemplos, a PACE e a PNeS que apenas simulam o sistema e analisam as propriedades referentes à estrutura da rede [GMBH 2008, Suraj 2018] – a contribuição central da tradução reside no fato de se obter um novo modelo que permite usar as ferramentas de análise disponíveis para GGA com o fim de provar propriedades referentes às classes das computações e realizar provas de teoremas utilizando a ferramenta Rodin [Cavalheiro et al. 2017, Cavalheiro 2010].

Para mostrar que a tradução está correta, deve-se garantir que a semântica foi preservada pela codificação, isto é, que a RPFG e a GGA possuem comportamentos equivalentes.

A fim de verificar-se o estado da arte com relação ao tema abordado, foram analisados trabalhos sobre traduções já realizadas com outros tipos de redes de Petri e sobre Gramáticas de Grafos Fuzzy (GGs Fuzzy) que estão disponíveis. Ao que se refere às traduções já realizadas, a estratégia básica adotada é sempre a mesma, isto é, o grafo inicial da gramática é obtido a partir da marcação inicial da rede, cada transição gera uma

*O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

regra da gramática e, para gramáticas tipadas, o grafo tipo é composto pela representação de todos os lugares da rede. O que difere uma tradução da outra é o mapeamento dos elementos específicos presentes em cada tipo de Rede de Petri para a gramática gerada. Na tradução aqui proposta, a característica principal é a presença dos elementos e das operações da Lógica Fuzzy. Já com relação às GGs Fuzzy analisadas, as principais características destas gramáticas são considerar um tipo específico de Lógica Fuzzy e não permitir a escolha de diferentes operadores fuzzy. A GGA Fuzzy derivada da tradução proposta difere-se das demais por fornecer um modelo mais genérico que pode ser estendido com outros tipos de Lógica Fuzzy de forma simplificada, além disso, o novo modelo permite incluir situações não fuzzy na modelagem.

O restante deste trabalho está organizado da seguinte forma: as seções 2, 3 e 4 referem-se ao referencial teórico utilizado neste trabalho; na Seção 5 é apresentada a tradução propriamente dita; e a Seção 6 discorre sobre as considerações finais.

2. Lógica Fuzzy

A Lógica Fuzzy estende a lógica clássica inserindo a incerteza presente no mundo real. Assim, a função de pertinência é uma extensão da teoria dos conjuntos clássica que define o quanto um elemento pertence a um dado conjunto. Um conjunto fuzzy A em um universo de discurso U é caracterizado pela expressão $A = \{(x, u_A(x)) \mid x \in X\}$, onde $u_A : X \rightarrow [0, 1]$. Esta representação permite a extensão dos conjuntos clássicos provendo o mapeamento da pertinência no intervalo entre 0 e 1. Definições e conceitos referentes à Lógica Fuzzy e a sistemas fuzzy podem ser encontrados em [Jafelice et al. 2005].

Sistemas fuzzy são formularizados pelas etapas de fuzzificação, de inferência (base de regras fuzzy) e defuzzificação. Neste projeto, o foco é dado na etapa de inferência. Os sistemas fuzzy operam os elementos dos conjuntos fuzzy através dos conectivos fuzzy e das regras fuzzy. Os principais conectivos fuzzy são: as T-normas (E fuzzy), por exemplo, o mínimo $T_M(x, y) = min(x, y)$ e o produto algébrico $T_P(x, y) = x.y$; as T-conormas (OU fuzzy), por exemplo, de Lukasiewicz $S_{LK}(x, y) = min(x + y, 1)$); e a negação fuzzy.

As regras fuzzy são produções do tipo *Se* x *é* A *então* y *é* B, computadas por uma operação de implicação fuzzy. O fato de se afirmar x é A ou y é B é representado por uma proposição fuzzy, por exemplo, a temperatura é baixa. As regras fuzzy podem ser classificadas em tipos considerando a posição dos conectivos (E/OU) fuzzy conforme a listagem a baixo [Chen et al. 1990].

- Tipo 1: R_i: **Se** d_j **então** d_k
- Tipo 2: R_i: **Se** d_{j1} E d_{j2} E... E d_{jn} **então** d_k
- Tipo 3: R_i: **Se** d_j **então** d_{k1} E d_{k2} E...E d_{kn}
- Tipo 4: R_i: **Se** d_{j1} OU d_{j2} OU...OU d_{jn} **então** d_k

Cada regra pode ser vinculada a um fator de certeza que indica o grau de credibilidade da mesma. Nesta abordagem, o fator de certeza é considerado na computação de cada regra [Reddy 2017].

3. Gramática de Grafos com Atributos

GGA é uma extensão da gramática de grafos convencional que comporta variáveis e operações em suas definições. A abordagem algébrica utilizada é a **Double Pushout** que

restringe a aplicação das regras utilizando a condição de colagem. Definições e conceitos referentes à GGAs podem ser encontradas em [Cavalheiro et al. 2017].

Cada GGA é definida com relação a uma especificação algébrica que descreve de forma abstrata os tipos de dados que podem ser utilizados nos seus componentes. Uma álgebra que implementa uma especificação algébrica $SPEC$ é chamada de $SPEC$-álgebra. Os grafos com atributos são compostos por vértices, arestas (dirigidas) e atributos que permitem associar a cada vértice valores de uma $SPEC$-álgebra. Uma GGA é composta por um grafo tipo, um grafo inicial e um conjunto de regras. O grafo tipo define as restrições para a estrutura gráfica e para os tipos de valores dos atributos os quais podem ser usados durante a especificação do sistema na gramática. O grafo inicial é um grafo estado com atributos que define o estado inicial do sistema. Uma regra de transformação de grafos descreve como um grafo estado pode ser alterado para alcançar um novo estado na gramática. As regras são caracterizadas por um span de morfismos $r : L \overset{\alpha L}{\hookleftarrow} K \overset{\alpha R}{\hookrightarrow} R$, onde o grafo L é o lado esquerdo da regra (LHS) e define quais elementos devem estar presentes no grafo estado para que esta seja aplicada; o grafo K é a interface que define quais elementos devem ser preservados durante a aplicação da regra; e o grafo R é o lado direito da regra (RHS) indicando qual o efeito da aplicação da regra. O morfismo αL especifica quais elementos devem ser deletados (elementos que estão em L mas não estão em K) e o morfismo αR define quais elementos serão criados (elementos que estão em R mas não estão em K). Além disso, as regras podem ter equações que especificam as condições para que essa possa ser aplicada. A semântica de uma GGA é dada pela aplicação de uma regra sobre um grafo estado do sistema. Uma regra está habilitada e pode ser aplicada se existir uma ocorrência m do LHS no grafo estado, a qual deve satisfazer todas as suas equações. A deleção, a preservação e a criação dos elementos deve levar em conta os atributos, preservando a álgebra.

Na Figura1 é dado o exemplo da especificação do jogo Pacman descrito por uma GGA. Na especificação algébrica TPacman é definido o tipo (*sort*) de dado Nat (conjunto dos números Naturais) e a operação que pode ser realizada sobre este tipo é a add (soma). No grafo tipo T o vértice representado pelo círculo preto e a aresta conectada a ele definem o tipo dos vértices dos lugares do tabuleiro e a forma como eles estão conectados. A maçã, o fantasma e o Pacman podem estar conectados apenas a vértices de lugar e o vértice ContaMaçãs possui o atributo CM do tipo Nat. O grafo inicial $G0$ define o estado inicial do jogo e possui um tabuleiro com doze lugares conectados entre si, um Pacman, um fantasma e uma maçã. O atributo CM é inicializado com 0 (nenhuma maçã foi consumida). A regra r é formada pelos grafos L, K e R e modela a ação de comer a maçã realizada pelo Pacman. Note que r está habilitada, já que existe uma ocorrência (indicada na Figura1 por um retângulo tracejado em vermelho) de L em $G0$ que satisfaz a equação da regra. Na construção da ocorrência o atributo CM de L cujo valor é a variável $cont$ será casado com o atributo CM de $G0$ cujo valor é 0, atribuindo o valor 0 à variável $cont$. O efeito da aplicação desta regra em $G0$ será: (i) a aresta que conecta a maçã ao seu lugar e o atributo CM serão deletados; (ii) uma nova aresta será criada conectando a maçã em outro lugar e o atributo CM será recriado com o valor da variável $cont1$ definido pela equação $cont1 = add(cont, 1)$.

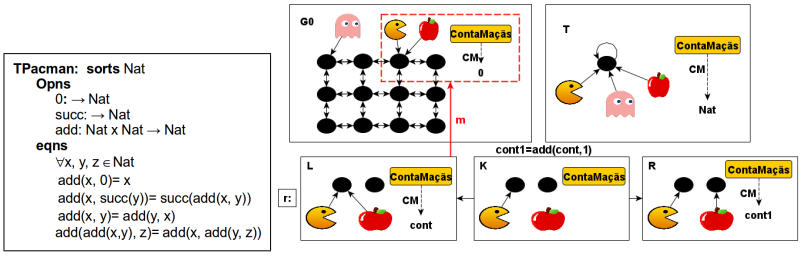


Figura 1. Exemplo reduzido da especificação do jogo Pacman

4. Redes de Petri Fuzzy Generalizadas

A RPFG é uma extensão da RPF que vincula triplas de operadores fuzzy às transições visando uma flexibilidade na escolha das operações fuzzy usadas. Definições e conceitos referentes às RPFGs podem ser encontrados em [Suraj 2012].

Uma RPFG é formada por um conjunto de lugares, um conjunto de proposições, transições, um conjunto de operadores e pelas marcações. Considere a RPFG utilizada como exemplo mostrada na Figura 2. Os círculos são os lugares da rede e vinculados à cada lugar tem-se: uma única proposição fuzzy d_i; um único rótulo p_i que indica o nome do lugar; e um único token (pontos pretos) cujo valor é definido no intervalo $[0, 1]$ o qual determina o valor verdade da proposição associada. Os retângulos modelam as transições. Cada transição da rede representa uma regra fuzzy cujos tipos foram enumerados na Seção 2. Na Figura 2 é ilustrado o conjunto de regras fuzzy que a RPFG do exemplo modela através da sua configuração. Vinculados à cada transição tem-se: os lugares de entrada ($I(t)$ pré-condições); os lugares de saída ($O(t)$ pós-condições); um fator de certeza (μ_i); um limiar (λ_i) usado para (des)habilitar uma transição em uma dada marcação; e uma tripla de operadores fuzzy (Op_{IN}, Op_{Out1}, Op_{Out2}), onde, Op_{IN} define a operação de agregação das entradas, Op_{Out1} define a operação de aplicação do fator de certeza e Op_{Out2} define a operação de agregação de cada saída. Os operadores usados no exemplo são: min e $*$ (T-normas mínimo e produto algébrico, respectivamente), o operador S_LK (T-conorma de Lukasiewicz) e o operador ID (identidade, usado como operador de entrada para regras do Tipo 1). As marcações definem os valores dos tokens para cada estado do sistema, lugares que não possuem tokens têm valor igual a 0. A primeira disposição dos tokens na rede é chamada de marcação inicial, neste exemplo, $M_0 = (0.6, 0.4, 0.7, 0.5, 0, 0, 0)$. A semântica da rede dá-se pela dinâmica de suas marcações que evoluem conforme os disparos das transições. Uma transição está habilitada em uma dada marcação M e pode disparar se o valor de saída do operador de entrada é positivo e maior ou igual ao valor limiar, isto é, $Op_{In}(M(p_1), \ldots, M(p_i)) \geq \lambda(t) > 0, \forall p_i \in I(t)$. Os disparos atualizam os valores das proposições e somente transições que não compartilham lugares podem disparar paralelamente. Alterações nos demais lugares da rede não são realizadas. Para os disparos das transições são considerados dois modos. O Modo 1 não preserva os tokens das pré-condições, já o Modo 2, preserva os tokens das pré-condições, e a computação da saída para ambos os modos é dada por $M'(p) = Op_{Out2}(Op_{Out1}(Op_{In}(M(p_1), ..., M(p_i)), \mu(t)), M(p))$, Se $p \in O(t)$. Analisando a semântica na rede do exemplo, inicialmente as transições $t1$, $t2$ e $t3$ estão habilitadas na marcação M_0. Pois, para $t1$ tem-se $min(0.6, 0.4) \geq 0.4$, para $t2$ tem-se $ID(0.7) \geq 0.2$ e para $t3$ tem-se $S\_LK(0.5, 0.6) \geq 0.2$. Porém as transições compartilham lugares e

somente uma pode disparar. Considerando os dois modos, se $t1$ dispara, um novo estado da rede definido pela marcação M' é alcançado. De acordo com o Modo 1, o novo estado alcançado está exemplificado na Figura 2 (a) e a computação de M' é realizada da seguinte forma: $M'(p_1) = 0$, $M'(p_2) = 0$, $M'(p_5) = S\_LK(*(min(0.6, 0.4), 0.4), 0) = 0.16$, $M'(p_6) = S\_LK(*(min(0.6, 0.4), 0.4), 0) = 0.16$. Demais lugares da rede permanecem inalterados. Já de acordo com o Modo 2, o novo estado alcançado está exemplificado na Figura 2 (b) e a computação de M' é realizada da forma quase análoga, a única diferença é que os tokens presentes em p_1 e p_2 não são consumidos durante o disparo.

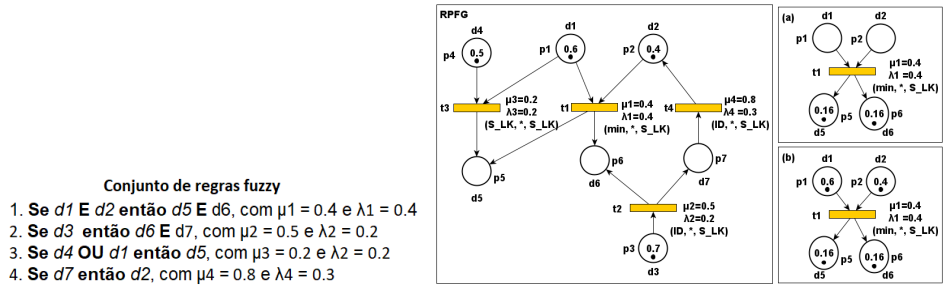


Figura 2. Regras fuzzy; RPFG de exemplo; (a) $t1$ após o disparo (Modo 1); (b) $t1$ após o disparo (Modo 2).

5. Tradução da Rede de Petri Fuzzy Generalizada para Gramática de Grafos com Atributos

Uma GGA é obtida a partir de uma RPFG, extraindo-se os elementos da GGA a partir da estrutura da RPFG. Tais elementos são: a especificação algébrica, o grafo tipo, o grafo inicial e o conjunto de regras.

A especificação algébrica TFuzzy ilustrada na Figura 3 é uma construção genérica que usa a especificação algébrica NatBool (álgebra dos números naturais e dos booleanos), também ilustrada na Figura 3. Em TFuzzy é definido o tipo de dados $[0, 1]$, o qual é uma discretização (números racionais) do conjunto dos valores fuzzy referentes ao intervalo $[0, 1]$ de reais. As assinaturas das operações fixas estão definidas em **opns** e são: as constantes 0 e 1 (bordas do intervalo); a constante undef (valor indefinido); a operação $./.$ (divisão entre dois números naturais que permite realizar a discretização); as operações $. \leq_R .$ e $. \neq_R .$ (comparações entre valores); e a operação $. + .$ (soma). As propriedades algébricas das operações devem ser enumeradas em **eqns** (omitido por questões de espaço). A especificação Tfuzzy para uma tradução é obtida adicionando-se as definições das operações fuzzy da rede. Na Figura 4 é ilustrada a especificação TFuzzy estendida com os operadores da rede de exemplo. Em **opns** foram inseridas as assinaturas dos operadores min, $*$, ID e S_LK. Em **eqns** deve-se adicionar as propriedades algébricas de tais operadores.

A construção do grafo tipo da GGA dá-se por meio da função $Trad_T(N)$ cujo argumento N é a RPFG. Todos os grafos da GGA são tipados sobre o grafo tipo construído, onde o morfismo de tipagem é injetor. Isto é, em cada grafo existe apenas uma instância de cada tipo de elemento. Um grafo tipo é obtido a partir de uma RPFG da seguinte forma: o conjunto de vértices do grafo é formado pelo conjunto de lugares da rede; o conjunto de

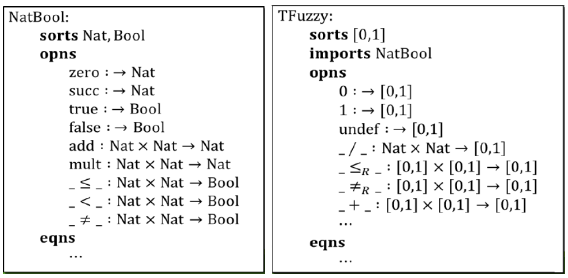


Figura 3. Especificações algébricas: NatBool (naturais e booleanos) e TFuzzy (fuzzy)

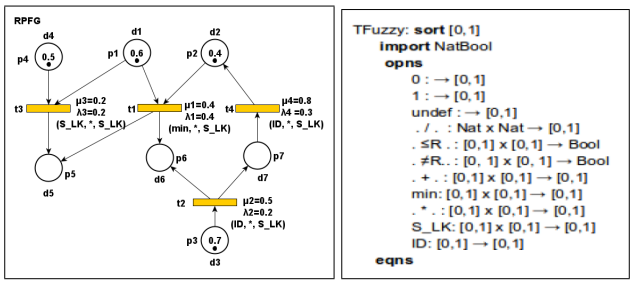


Figura 4. RPFG de exemplo e TFuzzy estendida com as operações da rede

arestas é vazio (grafos discretos); os atributos vinculados aos vértices são definidos pelas proposições; vértices e atributos são associados de acordo com o mapeamento bijetivo existente entre lugares e proposições; os atributos estão associados ao seu tipo de dado construído na especificação algébrica TFuzzy ($[0, 1]$). Na Figura 5 é ilustrado o grafo tipo T obtido da rede de exemplo. Pode-se ver que cada lugar da rede transformou-se em um vértice do grafo, pois, cada tipo de token é definido pelo lugar da rede no qual se encontra, já que cada lugar contém um único token. As proposições da rede transformaram-se nos atributos do grafo e cada atributo está vinculado ao tipo $[0, 1]$.

A construção de um grafo estado da GGA dá-se por meio da função $Trad_G(M, N)$ cujos argumentos são a marcação M e a RPFG. Um grafo estado é um grafo tipado com atributos obtido a partir de uma marcação da rede como segue: o conjunto de vértices do grafo é formado pelos lugares da rede; o conjunto de arestas é vazio (grafos discretos); os atributos vinculados aos vértices são obtidos através das proposições da rede; vértices e atributos são associados de acordo com o mapeamento bijetivo existente entre lugares e proposições; e os valores associados aos atributos são obtidos a partir da marcação da rede e representam os valores dos tokens. O grafo estado é tipado sobre o grafo tipo, onde cada elemento é mapeado para ele mesmo no grafo tipo. O Grafo inicial é um grafo estado obtido a partir da marcação inicial da rede. Na Figura 5 é ilustrado o grafo inicial $G0$ derivado da rede de exemplo. Nele pode-se constatar que os valores dos atributos definem exatamente os valores determinados pela marcação M_0 (primeiro estado) da rede.

Para cada transição da rede é gerada uma regra da gramática. O grafo L e o grafo R são compostos por vértices obtidos das pré-condições ($I(t)$) e das pós-condições ($O(t)$) de cada transição. O grafo de interface K é vazio, pois, todos os elementos serão consumidos e recriados pela aplicação da regra. Nos grafos das regras os valores dos atributos

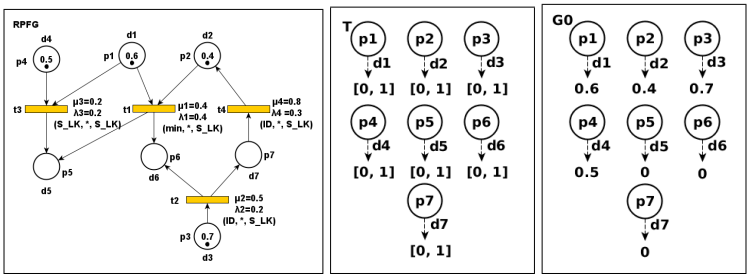


Figura 5. RPFG de exemplo; grafo tipo T; grafo inicial G0

são variáveis, e estas, distinguem os elementos. As variáveis x_i representam os valores obtidos das pré-condições e as y_i representam os valores obtidos das pós-condições. De acordo com o modo de disparo da RPFG obtêm-se dois conjuntos de regras distintos que se diferenciam apenas pela construção do conjunto de equações. Para o conjunto de regras que simula o Modo 1 as equações são construídas como segue: o primeiro tipo de equação é o $\lambda(t_i) \leq Op_{In}(x_{d_1}, ..., x_{d_n}) = true$ que testa se o limiar da rede é satisfeito; o segundo tipo de equação é o $y'_d = Op_{Out2}(Op_{Out1}(Op_{In}(x_{d_1}, ..., x_{d_n}, \mu(t)), y_d)$, um para cada lugar de saída da transição, que define os novos valores das proposições derivadas das pós-condições da transição; e o terceiro tipo de equação é o $x'_d = 0$, um para cada lugar de entrada da transição, que zera os valores das proposições derivadas das pré-condições da transição. Para o conjunto de regras que simula o Modo 2 as equações são construídas de forma quase análoga diferenciando-se apenas no terceiro tipo de equação, o qual é o $x'_d = x_d$, que mantêm os valores das proposições derivadas das pré-condições. Na Figura 6 é mostrada a regra $r1$ obtida da tradução da transição t_1 de acordo com o Modo 1. As pré-condições de $t1$ (p_1 e p_2) e as pós-condições (p_5 e p_6) se transformaram nos vértices dos grafos $L1$ e $R1$. O grafo K é omitido. Na construção do conjunto de equações da regra, as ocorrências de μ e λ foram substituídas pelos seus respectivos valores definidos em t_1. Os operadores fuzzy também foram substituídos pelos seus respectivos símbolos. A primeira equação da regra corresponde à condição que habilita a transição t_1 na rede, a segunda e a terceira atualizam os valores dos atributos derivados das pós-condições (y'_5 e y'_6). E as duas últimas equações zeram os valores dos atributos derivados das pré-condições.

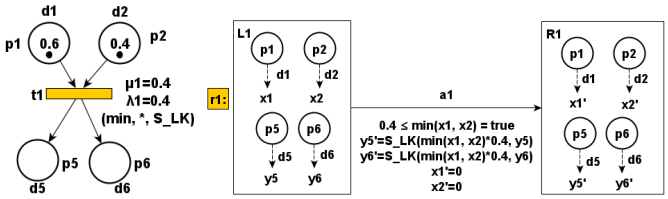


Figura 6. Transição $t1$ da RPFG de exemplo e regra $r1$ obtida da tradução

A semântica das RPFGs deve ser preservada pela tradução, independente do modo de disparo utilizado. Intuitivamente, considerando um mesmo estado em ambos os modelos, se na RPFG há uma transição habilitada neste estado cujo disparo gera uma nova marcação, então a regra da GGA correspondente à essa transição, também está habilitada no mesmo estado, e sua aplicação gera um grafo estado que corresponde à marcação ob-

tida pelo disparo da transição na RPFG. A relação inversa também deve ser válida. O Teorema 1 garante que a GGA obtida da tradução proposta preserva a semântica da rede generalizada. A prova deste teorema pode ser encontrada em [Vieira 2021].

Teorema 1 ***(Preservação da Semântica):*** *Dada uma* ***RPFG*** $N = (P, Tr, D, I, O, \mu, \beta, \lambda, Op, \delta, M_0)$ *e sua tradução para* ***GGA*** *(considerando o Modo 1 ou 2 de disparo)* $GG = (T, G0, Rules)$*, uma transição* $t \in Tr$ *está habilitada em uma marcação* M *de* N *e seu disparo deriva* M' *(considerando o Modo 1 ou 2 de disparo) se e somente se existe uma ocorrência* m *para a regra* $p_t \in Rules$ *em* G *(um grafo estado da GGA) resultando em um grafo* H*, de modo que* $Trad_G(M, N) = G$ *e* $Trad_G(M', N) = H$*.*

6. Conclusões e Trabalhos Futuros

Neste resumo foi apresentada a tradução de uma RPFG para GGA. Nesta tradução, definiu-se a construção de uma GGA fuzzy cujo conjunto de regras pode ser obtido de acordo com os dois modos de disparo da rede. Além disso, foi apresentada a prova da preservação da semântica de forma sucinta. Durante a prova da preservação da semântica foram identificadas algumas incorreções na tradução, as quais foram corrigidas, o que ressalta a importância deste tipo de prova. Como trabalhos futuros propõem-se construir um novo modelo de AGGs Fuzzy, usando como base a tradução definida neste trabalho e utilizar a ferramenta Rodin, mais especificamente o provador de teoremas, para realizar a verificação de propriedades dos sistemas especificados com este novo modelo. Além disso, pretende-se propor novas especificações algébricas para considerar outros tipos de Lógica Fuzzy.

Referências

Cavalheiro, S. A. C. (2010). Relational approach of graph grammars. Master's thesis, Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

Cavalheiro, S. A. d. C., Foss, L., and Ribeiro, L. (2017). Theorem proving graph grammars with attributes and negative application conditions. *Theoretical Computer Science*, 686:25–77.

Chen, S.-M., Ke, J.-S., and Chang, J.-F. (1990). Knowledge representation using fuzzy petri nets. *IEEE Transactions on Knowledge and Data Engineering*, 2(3):311–319.

GMBH, I. (2008). Cookbook pace 2008. Accessed in 10 of january of 2020.

Jafelice, R. S. d. M. et al. (2005). Notas em matemática aplicada 17. Sociedade Brasileira de Matemática Aplicada e Computacional, São Carlos/SP.

Reddy, P. V. S. (2017). Fuzzy logic based on belief and disbelief membership functions. *Fuzzy Information and Engineering*, 9(4):405–422.

Suraj, Z. (2012). Generalised fuzzy petri nets for approximate reasoning in decision support systems. In *CS&P*, pages 370–381. Citeseer.

Suraj, Z. (2018). Pnes: Tools for the design and analysis petri nets. In *2018 5th International Conference on Control, Decision and Information Technologies (CoDIT)*, pages 391–396.

Vieira, J. K. (2021). Uma tradução de redes de petri fuzzy generalizadas para gramáticas de grafos com atributos. Dissertação (mestrado em ciência da computação), Universidade Federal de Pelotas.