UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO BORN VIEIRA

CaTReS - Ferramenta de Apoio à Pesquisa e Ensino em Teoria das Categorias

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Paulo Fernando Blauth Menezes
Orientador

Porto Alegre, março de 2006

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Vieira, Rodrigo Born

CaTReS - Ferramenta de Apoio à Pesquisa e Ensino em Teoria das Categorias / Rodrigo Born Vieira. – Porto Alegre: PPGC da UFRGS, 2006.

110 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2006. Orientador: Paulo Fernando Blauth Menezes.

1. Simuladores Computacionais. 2. Teoria das Categorias. 3. Ensino a Distância. 4. Matemática. I. Menezes, Paulo Fernando Blauth. II. Título.

AGRADECIMENTOS

À Universidade Federal do Rio Grande do Sul por ter me oferecido a oportunidade ímpar de adquirir formação superior pública de qualidade.

Ao Instituto de Informática da UFRGS - meu lar de aprendizado e pesquisa -, seus professores e funcionários, pelo empenho e pela paixão com que exercem suas atividades e pela boa vontade com a qual sempre me atenderam.

Ao meu orientador e amigo, Paulo Fernando Blauth Menezes, por sempre ter sido uma pessoa solícita e por ter me ajudado a encontrar o norte no meu mestrado quando precisei.

A minha noiva e companheira, Daisy Cristina Castilhos Martins, por sempre ter estado ao meu lado com seu amor e carinho, me apoiando e tolerando minha ausência quando necessário foi.

Aos meus colegas do Laboratório de Fundamentos da Computação pela parceria e pela troca de experiências e conhecimentos que tanto me engrandeceram.

Aos meus pais, Nicomedes Vieira e Eclair Born Vieira, por todo o amor, apoio e por terem valorizado a minha formação e investido ao longo de toda a minha vida na minha educação, bem de valor inestimável que liberta.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ANSI	American National Standards Institute
CAPES	Fundação Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CASE	Compoter Aided Software Engineering
CaTLeT	Category Theory Learning Tool
CaTReS	Category Theory Researching Software
CGI	Common Gateway Interface
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico
CTDT	Category Theory Database Tools
DBC	A Database of Categories
EAD	Ensino a Distância
FAPERGS	Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul
GDCT	Graphical Database for Category Theory
GML	Graph Modelling Language
HTML	Hypertext Markup Language
IMS	Instructional Management Systems
JVM	Java Virtual Machine
LFC	Laboratório de Fundamentos da Computação
MEC	Ministério da Educação
PPGC	Programa de Pós-Graduação
SDK	Software Development Kit
SGEAD	Sistema de Gerenciamento para Ensino a Distância
SQL	Structured Query Language
UFRGS	Universidade Federal do Rio Grande do Sul
UML	Unified Modelling Language
VGJ	Visualizing Graphs with Java
WWW	World Wide Web

LISTA DE SÍMBOLOS

$f:A \to B$	Morfismo com Domínio em A e Contradomínio em B
$f \circ g$	Morfismo f composto com o morfismo g
$A \times B$	Produto cartesiano do conjunto A com o conjunto B
$A \times B$	Produto categorial do objeto A com o objeto B
$A \times B$	Multiplicação entre os operandos A e B
$A \subseteq B$	Conjunto A está contido no conjunto B
i1 | i2	Operação lógica “ou” bit a bit entre os inteiros i1 e i2
$A=\{1,2\}$	Conjunto A cujos elementos são 1 e 2
$a \in A$	Elemento a pertence ao conjunto A
$\langle a,b \rangle$	Par ordenado formado por a e por b
f(x)	x aplicado ao morfismo f

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Teoria das Categorias é uma ramificação da Matemática Pura com campo científico aparentemente distinto daquele que é objeto de estudo e pesquisa para a Ciência da Computação. Entretanto, algumas características dessa teoria matemática demonstram sua utilidade na pesquisa computacional.

Dentre essas características podemos citar independência de implementação, dualidade, herança de resultados, possibilidade de comparação da expressividade de formalismos, notação gráfica e, sobretudo, expressividade das construções categoriais. Sua expressividade é explicitamente destacada pelo MEC nas Diretrizes Curriculares de Cursos da Área de Computação e Informática, onde afirma-se que "Teoria das Categorias possui construções cujo poder de expressão não possui, em geral, paralelo em outras teorias".

Entretanto, Teoria das Categorias tem encontrado obstáculos para ser efetivamente aplicada na Ciência da Computação. A baixa oferta de bibliografia - predominantemente de língua inglesa - e a falta de uniformidade na exposição do que sejam os tópicos introdutórios convergem e potencializam outro grande empecilho à sua propagação: a baixa oferta de cursos com enfoque em Teoria das Categorias.

A fim de transpor essas dificuldades, Fábio Victor Pfeiff desenvolveu o CaTLeT, um aplicativo de interface visual que tinha como objetivo facilitar o acesso aos conceitos introdutórios de Teoria das Categorias. Com inspiração fortemente educacional, CaTLeT somente é capaz de representar objetos e morfismos atômicos, o que o limita a servir somente aos conceitos iniciais. Em 2003, o CaTLeT foi ampliado e os objetos e morfismos, antes atômicos, passaram a representar conjuntos e relações, respectivamente.

Este projeto consiste em uma ampliação tanto do CaTLeT quanto dos objetivos que justificaram sua criação. Esta dissertação trata de um projeto de simulador categorial e de sua respectiva implementação as quais visam fornecer suporte computacional a fim de facilitar o acesso a conceitos intermediários de Teoria das Categorias e servir como suporte à pesquisa na área. A construção desse simulador possui três critérios de avaliação como parâmetro: boa acessibilidade, alta relevância das estruturas implementadas e alta cobertura.

A nova ferramenta - denominada CaTReS - deve manter a acessibilidade a usuários leigos que sua predecessora possui e ampliar significativamente as estruturas suportadas, além de incluir tratamento à conceitos funtoriais. Dessa maneira, este projeto vem para auxiliar na superação dos obstáculos anteriormente mencionados.

Palavras-chave: Simuladores Computacionais, Teoria das Categorias, Ensino a Distância, Matemática.

CaTReS: Computational Support for Researching and Teaching of Category Theory

ABSTRACT

Category Theory is a branch of Pure Mathematics with a scientific field apparently distinct from Computer Science ones. However, some of its properties show its utility in computational research.

Among these characteristics we can mention implementation independence, duality, inheritance of results, ability to compare the expressiveness of other formalisms, strong basis on graphical notation and, above all, the expressiveness of its constructions. Its expressiveness is explicitly mentioned by MEC in the Curricular Directives of Courses in Computation and Informatics Areas, in which it's affirmed that "Category Theory has constructions whose expressiveness power isn't found, in general, in other theories".

However, Category Theory has been facing some obstacles to become effectively applied in Computer Science. The lack of published work - mostly available in English - and the heterogeneity in the presentation of introductory topics have lead to another great obstacle to its propagation: the small number of educational courses that emphasize it.

In order to surpass these difficulties, Fábio Victor Pfeiff developed CaTLeT, a visual interface application whose goal was making easier the access to the Category Theory introductory concepts. With strong educational inspiration, CaTLeT is able to represent only atomic objects and morphisms, what restrict its usage to the learning of introductory concepts. In 2003, CaTLeT was extended and the objects and morphisms, which were atomic elements, become structured, representing sets and relations respectively.

This project consists of an extension both of CaTLeT and the goals that inspired its creation. This master's thesis deals with the project of a categorical simulator and its implementation which objectives providing computational support to make the access to intermediate concepts of Category Theory easier and providing support for researches in the area. The construction of this simulator was guided by three evaluation criteria: good accessibility, high relevance of the implemented structures and high coverage.

The new tool - named CaTReS - was projected to maintain the accessibility to laymen, that CaTLeT has, and to extend significantly the supported structures, besides including functorial support. So, this project comes to collaborate on surpassing the obstacles mentioned before.

Keywords: Computational Simulators, Category Theory, Distance Education, Mathematics.

1 INTRODUÇÃO

Esta dissertação, submetida à avaliação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação junto ao Programa de Pós-Graduação (PPGC) da Universidade Federal do Rio Grande do Sul, trata de um projeto de simulador categorial e de sua respectiva implementação as quais visam fornecer suporte computacional a fim de facilitar o acesso a conceitos intermediários de Teoria das Categorias e servir como suporte à pesquisa na área.

1.1 Contexto do Trabalho

Há pelo menos quatro áreas com as quais este projeto contribui e em que ele ancora-se: Teoria das Categorias, simuladores computacionais, ensino a distância e processos cognitivos.

1.1.1 Teoria das Categorias

Tida como a teoria das construções matemáticas [PIA92], Teoria das Categorias é uma ramificação da Matemática Pura desenvolvida por Samuel Eilenberg e Saunders Mac Lane em 1945 [EIL45], como uma decorrência de seus trabalhos em topologia algébrica.

Apesar de considerada como uma área de pesquisa essencialmente teórica, Teoria das Categorias surge na Ciência da Computação como uma perspectiva de solução de um de seus gargalos no campo aplicado: a baixa capacidade de expressão das ferramentas descritivas disponíveis. Tal limitação pode ser observada nos códigos fontes de ferramentas de escritório mais utilizadas nos dias de hoje, onde para implementar, por exemplo, um editor de textos, milhões de linhas de código fonte são escritas.

A altíssima prolixidade dos códigos fonte de grande parte dos programas acaba tornando impossível para um ser humano - ou mesmo para uma equipe de desenvolvedores - ter uma alta compreensão a respeito do produto implementado. Em virtude disso, problemas decorrentes de erros de codificação acabam ocorrendo com elevada freqüência.

Considerando-se a complexidade dos sistemas computacionais atuais, verifica-se que, de certa forma, o desenvolvimento de soluções para os problemas propostos está limitado à capacidade do ser humano de expressar os problemas e suas soluções. Assim, quando mais expressivo for o formalismo usado, mais avanços podem ser esperados. Inclusive, formalismos mais expressivos auxiliam não só nas especificações e provas, mas, principalmente, em um melhor entendimento dos problemas, bem como em uma maior simplicidade e clareza nas soluções [MEN2001].

O potencial de aplicação de Teoria das Categorias em Ciência da Computação é reconhecido pelo MEC [BRA2005], como segue:

> *Teoria das Categorias possui construções cujo poder de expressão não possui, em geral, paralelo em outras teorias . Esta expressividade permite formalizar idéias mais complexas de forma mais simples bem como propicia um novo ou melhor entendimento das questões relacionadas com toda a Ciência da Computação. Como Teoria das Categorias é uma ferramenta nova, para exemplificar, vale a pena estabelecer um paralelo com a linguagem Pascal: Teoria das Categorias está para a Teoria dos Conjuntos assim como Pascal está para a linguagens Assembler.*

Pelo fato de Teoria das Categorias possuir uma história recente se comparada à outras teorias matemáticas - como, por exemplo, Cálculo Diferencial e Integral -, ainda há uma série de questões práticas a serem amadurecidas para que seu conteúdo seja mais acessível aos interessados.

Embora venha aumentando significativamente a oferta de bibliografia a respeito do assunto, ainda é bastante restrita e de formatação pouco padronizada. Não só os tópicos abordados variam bastante de uma obra para a outra como também a linguagem matemática utilizada não segue uma linha uniforme.

Por tratar-se de uma construção matemática peculiar e com potencial de aplicabilidade em uma área da Ciência de Computação a qual carece de estudos científicos mais elaborados e, sobretudo, cuja falta de soluções compromete gravemente a confiabilidade e o alcance das soluções computacionais desenvolvidas nos dias de hoje, este autor acredita ser questão de tempo para que Teoria das Categorias adquira maior acessibilidade.

1.1.2 Simuladores Computacionais

Entende-se por simulador computacional um aplicativo que reproduz o comportamento e as características de um objeto ou de um sistema (geralmente não-computacional). Muitas vezes, esses softwares são criados a fim de facilitar o aprendizado.

Talvez o exemplo mais conhecido desse tipo de programa seja os simuladores de vôo, nos quais se busca criar virtualmente todo o ambiente e os perigos que envolvem pilotar um avião sem, no entanto, expor o aluno aos riscos reais da atividade. Esses sistemas são muito utilizados também para simular comportamentos moleculares e celulares, facilitando a compreensão dos fenômenos químicos e biológicos.

Obviamente, não se pretende que um programa reproduza na plenitude as nuanças do sistema simulado. Um simulador de vôo é uma ferramenta eficiente para o aprendizado de pilotagem, mas não substitui a experiência de voar. Nesse sentido, um simulador é semelhante a um modelo: trata-se de uma simplificação útil da realidade. Tal situação é bastante conhecida na Ciência da Computação: trata-se das limitações em representar estruturas e resolver problemas em um computador.

Tais limitações são perceptíveis quando se fala em reproduzir Teoria das Categorias em um computador. Apesar de a simulação computacional de Teoria das Categorias ser uma área de pesquisa com poucos trabalhos realizados, é fácil notar que não é possível construir um simulador capaz de representar plenamente Teoria das Categorias. Conseqüentemente, não é possível implementar a verificação de uma propriedade que funcione para qualquer tipo de categoria.

Entretanto, projetar e implementar estruturas categoriais relevantes em um computador é uma missão viável - como, por exemplo, representar categorias finitas com objetos representando conjuntos finitos e morfismos sendo relações. Esse tipo de simulador é uma forma eficiente de tornar mais palpáveis conceitos categoriais muito abstratos, tornando,

assim, esses conceitos teóricos mais próximos de pesquisas aplicadas.

Há uma carência de pesquisas nessa área. Os simuladores categoriais existente lidam somente com conceitos categoriais básicos [ROS98] [FLE95] ou são inacessíveis para leigos [RYD88].

Observando esse cenário, o grupo de pesquisa LFC - do qual este autor faz parte - vêm pesquisando a respeito de tecnologias para implementar construções categoriais e implementando simuladores.

1.1.3 Ensino a Distância

Atualmente, uma das vertentes de pesquisa mais efervescentes é a que se preocupa com Ensino a Distância (EAD). Seu objetivo é proporcionar acesso a materiais instrucionais para um número de alunos maior do que é possível em aulas presenciais. Adicionalmente, esses alunos potencialmente podem estar geograficamente dispersos por grandes distâncias.

O caráter multidisciplinar de EAD reserva grande parcela de contribuição a ser provida por Ciência da Computação, especialmente após os adventos da Internet, da *World Wide Web* e do hipertexto.

Muito tem sido investido no estudo e exploração da rede de computadores como uma ferramenta eficiente para o ensino [OWS97]. O material desenvolvido com essa finalidade, entretanto, ainda apresenta problemas. As principais deficiências apontadas por pesquisadores [IMS98] são a falta de integração e compatibilidade entre os materiais desenvolvidos e a necessidade de gerenciamento da enorme quantidade de informação disponibilizada na Internet. Uma forma de buscar a solução para tais problemas está sendo pesquisada com a criação dos Sistemas de Gerenciamento para Ensino a Distância - SGEADs.

SGEADs, ou do inglês IMS - *Instructional Management Systems* -, são um conjunto de ferramentas para a construção do material instrucional. Esse conjunto não inclui apenas ferramentas para a manipulação de textos e gráficos, mas também ferramentas administrativas, acompanhamento do desenvolvimento do aluno, aplicação e correção de testes, avaliações e trabalhos extraclasse, etc. Enfim, tudo o que se faz necessário em um ambiente de ensino.

Tal classe de sistemas permite, por exemplo, que novos conhecimentos cheguem aos alunos isolados dos grandes centros de ensino e que professores sejam compartilhados eficientemente por diversos alunos localizados em diferentes locais.

Um exemplo de SGEAD é o sistema Hyper-Automaton [MAC2000], o qual utiliza autômatos finitos com saída como um modelo de cursos na web, utilizando hipertextos.

1.1.4 Processos Cognitivos

Quando falamos de processos cognitivos estamos lidando com a teoria do conhecimento, o que nos remete à pergunta: “Como é possível o conhecimento”, a qual se desdobrou numa pluralidade de problemas, referentes à natureza e às condições preliminares do conhecimento lógico-matemático, do conhecimento experimental de tipo físico, etc [PIA73].

Nesse campo de pesquisa destacou-se Jean Piaget, renomado psicólogo e filósofo suíço, conhecido por seu trabalho pioneiro no campo da inteligência infantil. Piaget passou grande parte de sua carreira profissional interagindo com crianças e estudando seu processo de raciocínio. Seus estudos tiveram um grande impacto sobre os campos da psicologia e pedagogia.

O livro *Morphisms and Categories* [PIA92], de Jean Piaget, é uma publicação póstuma. Os manuscritos originais são do início dos anos 80, enquanto a primeira publicação apenas ocorreu em 1990, na França, possivelmente demonstrando tanto a precocidade com que Piaget percebeu a importância de Teoria das Categorias para o desenvolvimento do raciocínio quanto a marginalidade com que esse assunto tem sido tratada por muitos estudiosos.

Nesse trabalho, Piaget mostra, através de experiências realizadas com crianças, que a idéia de composição, típica de sistemas categorias, está na gênese no ser humano. De fato, o homem naturalmente tende a pensar em propriedades simples de esquemas, e então, sucessivamente, operações similares em esquemas de esquemas, e assim por diante, formando uma cadeia de composições de propriedades.

Os resultados obtidos por Piaget nos conduzem à conclusão de que o raciocínio humano é categorial. Tal colocação encontra base no seguinte trecho de seu livro:

> *Eu quis tornar plausível, em linhas gerais, a idéia de que Teoria das Categorias, considerada como a teoria das construções matemáticas, reflete a constituição genética das ferramentas cognitivas humanas, ou seja, o destacamento de esquemas transferíveis para um conjunto de ações, então operações similares nestes esquemas, então operações similares em esquemas de esquemas, e assim por diante. Assim sendo, me parece claro que o estilo categorial, como uma forma de vislumbrar um aspecto importante da gênese das faculdades cognitivas humanas, não é um estilo imposto à epistemologia da gênese de fora para dentro, mas é um estilo que, por natureza, é adequado para descrever as construções descobertas pela epistemologia da gênese.*

Os resultados obtidos no trabalho de Piaget não apenas mostram que Teoria das Categorias poderia ser lecionado antes do curso de graduação mas também mostram que isso deveria ser feito. O desenvolvimento do pensamento formal lógico em crianças permite desenvolver habilidades e linhas de raciocínio que normalmente não são desenvolvidas na infância, apesar de serem requeridas em diversas áreas do conhecimento. Esse tipo de habilidade, e em particular o pensamento geral e unificado, típico de Teoria das Categorias, apesar de ser intuitivo para os seres humanos, é desencorajado durante o processo de aprendizado.

Teoria das Categorias normalmente é estudado em alguns cursos de pós-graduação e, menos freqüentemente, cursos de graduação. Seus princípios matemáticos, que poderiam ser fáceis, naturais e intuitivos, tornam-se densos e abstratos em virtude da forma que se estimula os alunos do ensino fundamental e médio a raciocinar.

Pesquisar meios de tornar essa teoria abstrata um pouco mais concreta e intuitiva é um desafio e uma interessante área de pesquisa. Uma forma que o grupo de pesquisa LFC vislumbrou para auxiliar nesse processo é a construção de simuladores categoriais.

Embora os frutos deste trabalho sejam potencialmente úteis para um estudo mais detalhado sobre o impacto de simuladores categoriais na reconstrução do raciocínio categorial perdido, não é foco deste trabalho realizar tal estudo, assim como não é feito um estudo mais detalhado que interligue os resultados aqui obtidos aos resultados do trabalho de Piaget.

1.2 Objetivos

Este trabalho vem essencialmente para contribuir com a missão de tornar Teoria das Categorias mais acessível aos alunos e aos pesquisadores. Para tanto, buscou-se projetar e elaborar uma ferramenta computacional que seja capaz tanto de auxiliar ao aluno que busca adquirir conhecimentos intermediários de Teoria das Categorias quando ao pesquisador que deseja utilizar um software categorial para, por exemplo, realizar testes a fim de evidenciar a veracidade de uma hipótese.

Dentro desse contexto, o software gerado como fruto deste projeto foi elaborado observando três metas:

- **Boa Acessibilidade:** Entende-se aqui por acessibilidade como sendo a capacidade de um determinado usuário conseguir operar o programa com o mínimo de conhecimento prévio possível e com o menor esforço possível. Interface gráfica e acesso via Internet são requisitos importantes para atingir boa acessibilidade. Entenda-se, portanto, que o conceito de acessibilidade utilizado neste trabalho nada tem a ver com o conceito muitas vezes utilizado na Ciência da Computação, no que se refere a funcionalidades visando usuários portadores de necessidades especiais. Aqui, trabalha-se com a idéia de facilidade de uso e facilidade de chegar até a ferramenta;

- **Alta Relevância das Estruturas Implementadas:** como foi dito anteriormente, é inviável simular toda a Teoria das Categorias em um computador. Portanto, é necessário selecionar as estruturas categoriais a serem reproduzidas. Dessa forma, os elementos escolhidos deverão ser os mais relevantes possíveis - sob a ótica do aprendizado de conceitos categoriais e sob a ótica de contribuição para a pesquisa;

- **Alta Cobertura:** apesar de haver limitações em representar conceitos categoriais, é importante que, dentro do escopo definido como relevante e, portanto, alvo da implementação em questão, o software seja o mais completo possível. Portanto, quanto mais conceitos categoriais foram acrescentados no programa dentro do escopo estabelecido, mais útil tende a ser o programa.

Observando esses três critérios, essa dissertação deve produzir como resultados:

- Uma análise do estado da arte no que se refere à produção de simuladores computacionais e uma avaliação desses programas quanto à acessibilidade, relevância das estruturas implementadas e cobertura;

- Uma análise dos conceitos matemáticos utilizados no projeto;

- Um projeto UML do software, contendo diagrama de pacotes e diagrama de classes;

- Um programa, denominado CaTReS (acrônimo de “Category Theory Researching Software”), o qual deve possuir:

  - Boa usabilidade (quesito acessibilidade);
  - Capacidade de representar categorias cujos objetos sejam conjuntos e os morfismos sejam ou relações ou funções totais ou funções parciais, de acordo com o interesse do usuário, além de capacidade de representar estruturas funtoriais (quesito relevância);

  - Capacidade de avaliar propriedades categoriais e funtoriais (quesito cobertura).

- Uma avaliação da capacidade de integração do CaTReS com o ambiente Hyper-Automaton.

1.3 Apresentação da Dissertação

Esta dissertação está dividida em 8 capítulos e 2 apêndices, através dos quais são apresentadas as propostas do trabalho, seus objetos, as ferramentas teóricas utilizadas, o modelo do software implementado, a contribuição para o estado da arte e os artigos gerados a partir deste trabalho.

O capítulo 1 apresenta a introdução deste trabalho, onde são explicitadas as áreas do conhecimento nas quais este trabalho se insere e os objetivos traçados para esta dissertação.

O capítulo 2 apresenta o estado da arte no qual este trabalho se insere, apresentando uma análise a respeito dos diferentes simuladores categoriais encontrados, suas características, suas diferenças e uma avaliação no que diz respeito aos aspectos considerados importantes no contexto deste trabalho.

O capítulo 3 apóia-se nos estudo realizado no capítulo 2 e apresenta as funcionalidades e as características que o simulador categorial projetado neste trabalho deve atender, sob a luz dos mesmos critérios utilizados na avaliação das ferramentas do capítulo 2.

No capítulo 4 são apresentados os principais algoritmos e conceitos matemáticos apresentados no trabalho. Aqui, não é foco a definição de conceitos matemáticos tidos como pré-requisitos deste trabalho - como a definição de categoria, por exemplo - e sim as propriedades matemáticas das estruturas utilizadas que se tornaram úteis para implementar algumas funcionalidades e como estruturas funtoriais foram representadas no software.

O capítulo 5 apresenta o projeto do software implementado, focando nos aspectos relacionados aos diagramas UML e às diferenças entre o projeto do CaTLeT - aplicativo antecessor do software desenvolvido nesta dissertação - e o projeto do CaTReS.

O capítulo 6 apresenta, dentro dos propósitos deste trabalho de apresentar uma ferramenta que possa ser de utilidade para o ensino a distância de Teoria das Categorias, um esboço do que poderia ser a integração do software desenvolvido neste trabalho e do sistema de gerenciamento de ensino a distância Hyper-Automaton.

O capítulo 7 é o manual do usuário. Nele, é mostrado, através de telas e cenários, como é que o usuário interage com o software.

O capítulo 8 apresenta a conclusão da dissertação, observando resultados obtidos no desenvolvimento deste trabalho, avaliando se estes atendem aos objetivos traçados, analisando se os critérios estabelecidos como importantes nesta dissertação foram atendidos pelo simulador categorial, as publicações oriundas deste trabalho e apresentando possibilidades de trabalhos futuros.

O apêndice A apresenta o resumo estendido publicado no Eurocast 2005, no qual é feito um estudo e apresentação de alternativas referentes a simulação computacional de construções categoriais.

O apêndice B apresenta o artigo completo publicado no mesmo evento, a partir do qual está sendo gerada uma publicação no Lecture Notes in Computer Science. O conteúdo apresentado nas duas publicações está diluído ao longo dos 7 capítulos.

2 ESTADO DA ARTE

Teoria das Categorias possui apenas sessenta anos de existência. É natural, portanto, que campos científicos oriundos dessa teoria matemática possuam poucos trabalhos desenvolvidos e um vasto espaço para pesquisas científicas. É o caso de sistemas computacionais que implementem construções categoriais.

Neste capítulo é feito um apanhado de trabalhos que tem como resultado modelos computacionais de estruturas categoriais.

2.1 Computational Category Theory [RYD88]

Trata-se de uma das primeiras experiências bem acabadas de representação computacional dos conceitos categoriais. Chama a atenção tanto pela quantidade de conceitos categoriais que aborda quando pela forma que demonstra ser possível representar tais conceitos com razoável facilidade e clareza utilizando o paradigma de programação funcional.

Computational Category Theory é uma implementação de conceitos e construções de Teoria das Categorias na linguagem de programação funcional ML padrão. Desenvolvido por David E. Rydeheard, da Escola de Ciência da Computação da Universidade de Manchester, e por Rod M. Burstall, da Escola de Informática da Universidade de Edimburgo, este programa é citado por seus autores como um precursor de trabalhos relacionados à mecanização de Teoria das Categorias.

O software desenvolvido deu origem ao livro [RYD88] - na página mencionada na referência, tal livro é tratado como manual -, o qual descreve a codificação utilizada na implementação. A forma como foi elaborado torna o livro mais um material didático com apoio de recursos computacionais do que um guia a respeito de como a ferramenta é implementada. Ele aborda item por item os conceitos categoriais utilizados na ferramenta, explica-os, expõe a definição matemática e mostra como este conceito foi implementado em ML padrão. Ou seja, esse material pode ser utilizado para aprender ou ensinar Teoria das Categorias, como referência alternativa de conceitos categoriais ou ainda para estudar como o aplicativo foi implementado. Como é possível observar, portanto, apesar do software ser a obra-prima e o livro ter sido criado apenas como apoio - para descrevê-lo -, é inegável que este ficou tão completo e tão didático que aquele pode ser visto, dependendo da situação, como mero aplicativo de suporte.

O programa não é diretamente disponibilizado aos usuários. Somente os fontes ML estão acessíveis, sendo necessário, portanto, um compilador ou um interpretador para utilizar o programa. Os autores recomendam a utilização do sistema Moscow ML[MOS2005].

Nesse sistema, os fontes são compilados dentro do ambiente de operação. Após realizada a compilação de sete arquivos, que representam o código fonte, criam-se então

arquivos de entrada a fim de definir categorias, realizar cálculos e operações e verificar propriedades. Para tanto, é absolutamente necessário saber detalhes do código fonte para poder operar o sistema, além de conhecimento de código funcional para escrever os arquivos utilizados como entrada.

A definição de um tipo a partir do qual possamos instanciar categorias é feita da seguinte maneira:

```
datatype ('o,'a)Cat =
   cat of ('a->'o)*('a->'o)*('o->'a)*('a*'a->'a)
```

O objeto é dado por `'o` (*object*), o morfismo é dado por `'a` (*arrow*) as funções origem e destino são definidas com a assinatura `'a->'o`, a função identidade é assinada como `'o->'a` e a função parcial composição é dada por `'a*'a->'a`.

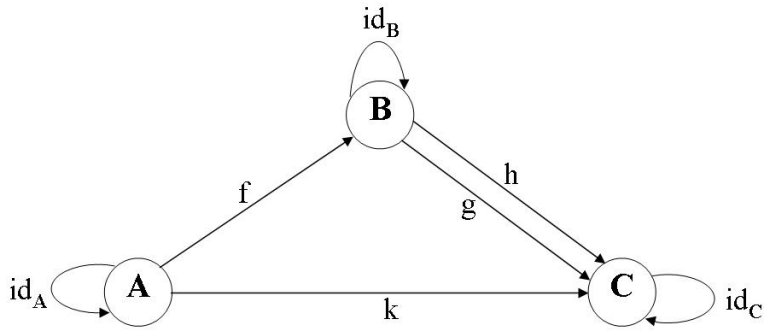


Figura 2.1: Exemplo de Categoria - id_A, id_B e id_C são as identidades de A, B e C, respectivamente, e g∘f=k e h∘f=k

A categoria da figura 2.1 temos a representação gráfica de uma categoria. Para representar essa categoria em ML, de modo a utilizá-la como entrada no Computational Category Theory, poderíamos entrar com os objetos e morfismos da seguinte maneira:

```
datatype Obj = A | B | C
datatype Morf = f | g | h | k | id of Obj
```

Nesse caso, objetos e morfismos são constantes. Eles estão sendo representados utilizando tipos enumerados. As identidades foram construídas utilizando objetos rotulados. Origem, destino e composição podem ser implementadas da seguinte maneira:

```
val categoria_exemplo
     let val o = fn f => A | g => B | h => B |
                 k => A | id(x) => x
         and d = fn f => B | g => C | h => C |
                 k => C | id(x) => x
         and ident = fn x => id(x)
         and comp =
           fn (id(x),u) => if d(u)=x then u
                      else raise nao_componivel
            | (u,id(x)) => if o(u)=x then u
                      else raise nao_componivel
```

```
          | (g,f) => k | (h,f) => k
          | _ => raise nao_componivel
    in cat(o,d,ident,comp) end
```

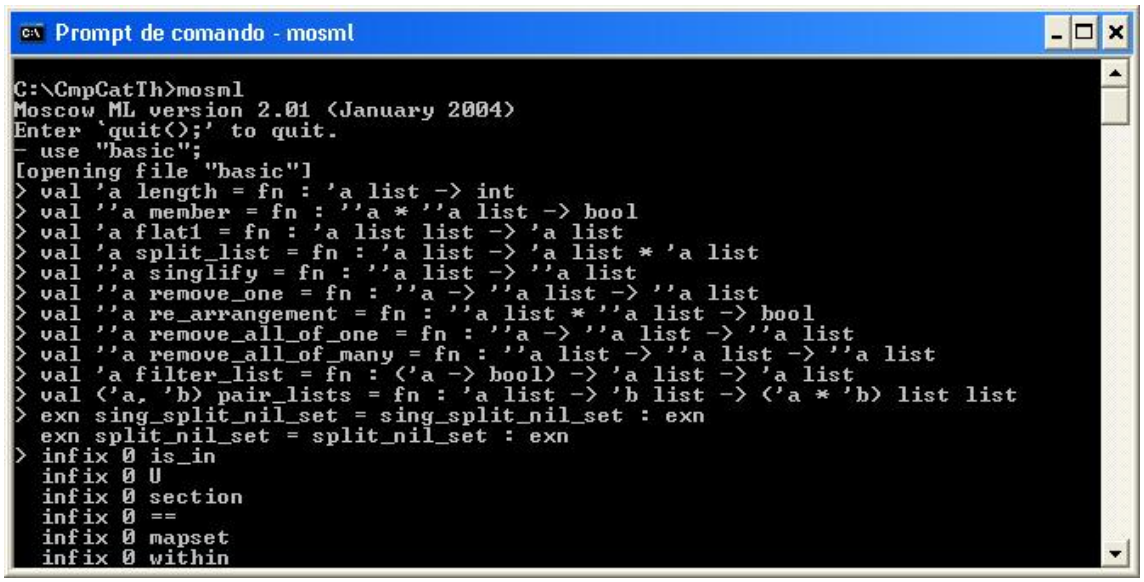


Figura 2.2: Sistema Moscow ML

A figura 2.2 nos mostra o ambiente Moscow ML, com o qual um usuário que deseje fazer uso do Computational Category Theory terá que lidar. A figura mostra a compilação do arquivo *basic* utilizando o comando *use "basic"*. Ao executar esse comando, será realizada a avaliação de todas as funções definidas no arquivo, as quais passarão a serem conhecidas pelo ambiente para, posteriormente, poderem ser utilizadas por outras funções.

O Computational Category Theory é bastante abrangente em relação aos conceitos categoriais que trata, implementando os conceitos de:

- categoria;
- subcategoria (cheia);
- monomorfismo e epimorfismo;
- isomorfismo;
- produto e coproduto;
- dualidade;
- limite e colimite;
- equalizador e coequalizador;
- produto fibrado e soma amalgamada;
- funtor;

- categoria das setas;
- transformação natural;
- adjunção;
- categoria cartesiana fechada;
- semitopos;
- topos.

Além disso, apresenta exemplos de categorias construídas neste sistema - como FinSet. Se acompanhamos o livro [RYD88], observamos o quanto o paradigma de programação funcional simplifica a definição de tais conceitos, os quais seriam bem menos simples de serem elaborados em uma linguagem de programação procedural ou orientada a objetos.

A grande limitação desse programa está, sem dúvida, em sua usabilidade. Para realmente servir para propósitos didáticos, esse software precisaria, no mínimo, sofrer adaptações de modo a permitir uma entrada de dados mais acessível ao usuário. Entretanto, mesmo utilizando-se de tal artifício, ainda assim seriam necessários ao usuário conhecimentos de operação em ambientes funcionais - um ambiente que não é, por si só, amigável.

2.2 A Database of Categories - DBC [FLE95]

Trata-se do primeiro simulador categorial desenvolvido com a participação de Robert Rosebrugh, do Departamento de Matemática e Ciência da Computação da Universidade de Mount Allison, no Canadá. A partir deste, foram desenvolvidos outros dois simuladores - citados posteriormente nesta dissertação - com melhorias especialmente na parte gráfica. Implementaram este simulador Michael Fleming e Ryan Gunther, ambos do Departamento de Ciência da Computação da Universidade de Waterloo, também no Canadá.

Esse software foi nominado DBC - acrônimo para *A Database of Categories*. No documento *A Database of Categories (pdf)*, disponível em [FLE95], os autores citam um e-mail escrito por David Rydeheard, um dos autores de Computational Category Theory, que trata de trabalhos posteriormente realizados na área de mecanização de Teoria das Categorias. Nesse e-mail, David Rydeheard coloca sua ferramenta e seu livro como ponto de partida para uma pesquisa nessa área.

Apesar de tal citação, quase nada há em comum entre DBC e Computational Category Theory - além do fato de ambas buscarem modelar estruturas categoriais. DBC foi implementado em linguagem C ANSI - utilizando, portanto, o paradigma de programação procedural.

Na figura 2.3 é possível ver algumas características do software. Trata-se de uma ferramenta textual, onde o usuário interage com a aplicativo através de um menu. Nota-se uma preocupação dos autores com a usabilidade (apesar de não se tratar de software gráfico) a qual não se verifica no Computational Category Theory.

O programa permite que se entre manualmente com novas categorias ou que as carregue de arquivos texto. Ele é capaz de armazenar na memória diversas categorias, podendo realizar cálculos e averiguar propriedades. Mais precisamente, o programa é capaz de:

Figura 2.3: Tela de Apresentação do DBC

- calcular a categoria dual e salvá-la em arquivo texto sem, no entanto, mantê-la carregada no programa;
- verificar se um dado objeto de uma categoria corresponde ao objeto inicial;
- verificar se um objeto e um par de morfismos dados correspondem a um coproduto binário;
- armazenar funtores.

Embora não sejam funcionalidades diretamente fornecidas pelo programa, este é capaz, através do cálculo da categoria dual, de realizar averiguações similares em relação ao objeto terminal e ao produto binário. Nesse aplicativo, é possível carregar um funtor sem que as categorias componentes tenham sido abertas.

A forma com que o programa trabalha com os conceitos de objeto inicial e coproduto binário evidencia o quão limitado ele é. A ferramenta não tem suporte para, por exemplo, apresentar um ou mais objetos iniciais de uma dada categoria. Apenas verifica se um objeto fornecido como entrada pelo usuário é o inicial. Dessa forma, o suporte que o software oferece é para a conferência de uma averiguação previamente feita pelo usuário.

Situação similar ocorre em relação ao coproduto binário. Entra-se com um objeto e um par de morfismos, e a partir daí é perguntado se estes formam de um coproduto binário. Observa-se que o programa informa que se trata de um coproduto, mas não informa entre quais objetos. A informação fica implícita, e cabe ao usuário saber que os objetos que dão origem ao produto calculado correspondem aos domínios dos respectivos morfismos dados como entrada.

Observa-se ainda uma imprecisão conceitual no diálogo com o usuário. Em uma dada categoria, caso fosse realizada uma consulta para verificar, por exemplo, se o objeto A e os morfismos f e g correspondem a um coproduto, o programa, em caso afirmativo, imprimiria a mensagem `A is a sum`. Em caso negativo, o programa responderia `A is not a sum`. Em outras palavras, o programa atribui ao objeto a propriedade de ser (ou de não ser) um coproduto, quando na verdade o coproduto é formado pelo objeto e pelas duas

imersões. Tal falha poderia induzir um aluno de Teoria das Categorias que tivesse utilizando a ferramenta como auxílio no aprendizado dos conceitos a uma percepção errada a respeito do que é coproduto.

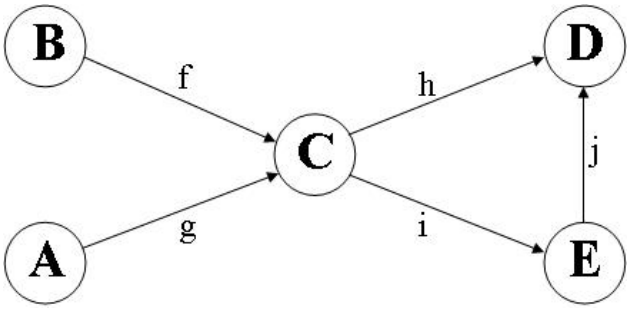


Figura 2.4: Exemplo de um grafo não-categorial - neste caso, h=j○i

Observa-se que os desenvolvedores não tiveram uma preocupação rigorosa em verificar se a estrutura recebida como entrada é, de fato, categoria. Um exemplo pode ser visto abaixo, onde o grafo não-categorial da figura 2.4 está codificada no formato reconhecido pelo DBC:

```
category
      Exemplo2
objects
      A, B, C, D, E.
arrows
      f:A->C, g:B->C, h:C->D, i:C->E, j:E->D.
relations
      ji = h.
```

O software recebe essa estrutura e trabalha com ela como se estivesse lidando com uma categoria. Entretanto, mesmo pressupondo que as identidades estão sendo omitidas, a estrutura representada não pode ser vista como categoria, posto que está faltando definir algumas composições - por exemplo, h○f e h○g. Não é possível pressupor que as composições também estejam sendo omitidas, posto que há uma composição representada em `relations` (`ji = h` significa h = j○i).

É bem claro que essa ferramenta é bem mais limitada do que àquela apresentada no item anterior. De fato, é a mais limitada dentre as apresentadas nesta dissertação. Entretanto, o programa já possui uma evolução no aspecto da acessibilidade. Não apenas por ser mais amigável em termos de interação com o usuário mas também por não requerer um ambiente de operação especial. O aplicativo é dependente de plataforma, posto que é implementado em C ANSI. Entretanto, não é difícil gerar uma versão executável para as diferentes plataformas tendo o código fonte disponível, posto que compiladores para a linguagem C padrão ANSI costumam ser uma dos mais disponibilizados (senão os mais disponibilizados) pelas diferentes plataformas.

2.3 Category Theory Database Tools - CTDT [ROS98]

Segunda ferramenta desenvolvida sob a supervisão de Robert Rosebrugh - desta vez, com a participação de Jeremy Bradbury, da Universidade de Mount Allison -, esta ferra-

menta é a sucessora imediata do DBC.

A primeira diferença em relação à antecessora que é importante de mencionar é a linguagem de programação utilizada na implementação. Enquanto DBC foi implementado em C ANSI, CTDT foi desenvolvido como um applet Java. Observando os códigos fontes, verifica-se que muitos algoritmos utilizados no DBC foram reaproveitados, assim como as estruturas de arquivos utilizadas.

Nos aspectos perceptíveis ao usuário, as alterações mostram-se mais claras no campo da interação homem-computador - embora menos do que se espera de uma aplicação gráfica, como veremos a seguir - do que no campo da funcionalidade. Os conceitos categoriais abordados no CTDT são praticamente os mesmos abordados pelo DBC - inclusive com as mesmas limitações e imprecisões conceituais já citadas.

O acréscimo funcional mais significativo refere-se à verificação do objeto inicial. Enquanto no DBC a verificação de objeto inicial restringia-se a informar se um objeto dado pelo usuário era ou não inicial, no CTDT é possível também solicitar que ele verifique tal propriedade em todos os objetos com um só comando, gerando assim uma lista de informações que associa cada objeto da categoria com a informação de que é ou de que não é um objeto inicial.

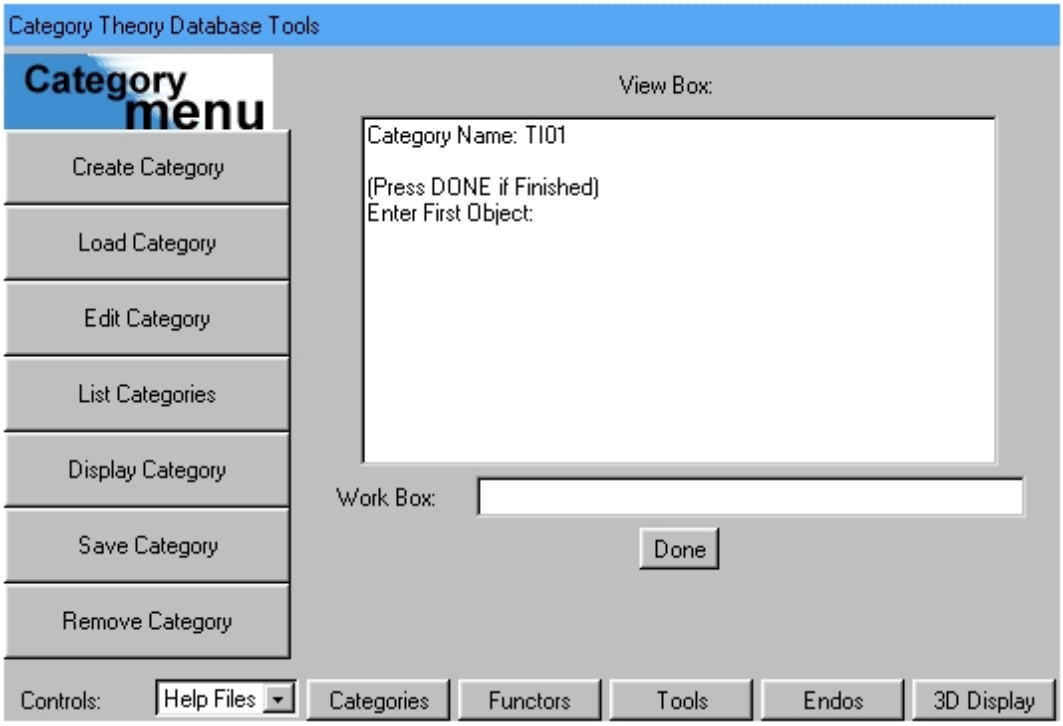


Figura 2.5: Tela do *Category Theory Database Tools* (CTDT)

A figura 2.5 nos permite ter uma idéia inicial de como ocorre a interação com o aplicativo. A partir dos cinco botões que aparecem na parte de baixo da tela é feito o acesso a todos os recursos do CTDT. Cada um desses botões disponibiliza um diferente menu - onde as opções possuem aspecto de botão - no lado esquerdo da tela. O menu visto à esquerda na figura aparece quando o botão `Categories` é pressionado.

Ao clicar em uma das opções do menu, o programa devolve um resultado ou um diálogo na caixa de edição intitulada `View Box`. Caso seja um diálogo, o usuário interage com esse ambiente através de outra caixa de edição intitulada `Work Box`. Nessa

caixa, o usuário digita a informação solicitada na outra caixa e pressiona “Enter”, em um tipo de interação claramente textual. Quando o usuário julgar que encerrou aquele ciclo de entrada de dados (por exemplo, quando já tiver informado todos os objetos os quais ele deseja que a categoria tenha) ele pressiona o botão `Done`.

Se compararmos o menu dessa figura com aquele outro que aparece na figura 2.3, observaremos a semelhança entre os diálogos realizados no CTDT e no DBC. Todas as opções apresentadas nesse menu aparecem no diálogo inicial do DBC, sendo que a opção `Category tools`, disponível no DBC, equivale ao botão `Tools` do CTDT e a opção `Functor menu` do DBC cumpre o mesmo papel do botão `Functors` do CTDT.

Tal semelhança, e sobretudo a maneira como o sistema interage com o usuário, evidenciam o quão pouco o grupo canadense se valeu de modernos recursos gráficos na elaboração do novo sistema. As principais interações entre usuário e software ocorrem através de duas caixas de texto. Portanto, o arcabouço gráfico implementado pelo grupo canadense pouco reflete em um ganho real em termos de usabilidade.

Tal ganho é mais eficientemente produzido por um módulo gráfico disponível na ferramenta, o qual é gerado quando pressionamos o botão `3D Display`. Ele é disponibilizado ao usuário assim que pelo menos uma categoria é carregada. Através desse módulo o usuário pode visualizar e editar a categoria de maneira gráfica - com objetos e morfismos visualmente representados. Tal recurso possui algumas limitações - por exemplo, não é possível especificar composições nesse modo -, mas sem dúvida é a inovação mais interessante do CTDT em relação ao DBC.

Convém ressaltar, entretanto, que esse módulo não foi criado pelo grupo canadense. Ele foi adaptado de modo a poder exibir e manipular categorias. O aplicativo é denominado *Visualizing Graphs with Java* (VGJ) [MCC2005] e foi criado por uma equipe de desenvolvedores do Departamento de Ciência da Computação e Engenharia de Software da Universidade de Auburn, Estados Unidos.

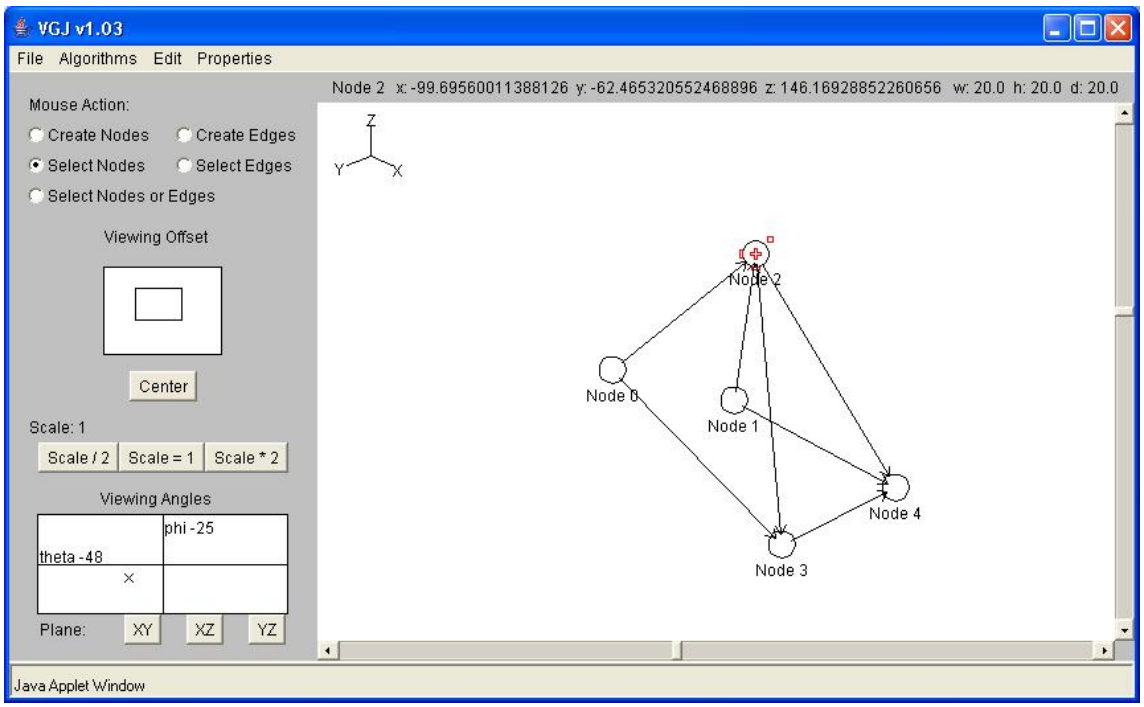


Figura 2.6: Tela do *Visualizing Graphs with Java* (VGJ)

O VGJ foi implementado utilizando a versão 1.0 do Java, mesma versão que foi utilizada na implementação do CTDT. A figura 2.6 nos mostra um grafo representado em VGJ e algumas das possíveis operações gráficas que podem ser executadas sobre ele. Apesar de tratar-se também de um applet, esta é bastante eficiente. É possível rotacionar um grafo nas três dimensões, alterar sua escala de visualização e transladá-lo.

O VGJ, dessa maneira, empresta ao CTDT um conjunto de alternativas de visualização e manipulação tridimensional de grafos que permitiu, com algumas alterações, um tratamento visual amigável com as estruturas categoriais representadas. Esse, sem dúvida, pode ser considerado o grande avanço do CTDT em relação ao DBC.

É possível que um dos motivos da escolha da linguagem de programação Java para implementar o CTDT tenha vindo do fato do VGJ ser também implementado nesta linguagem. Entretanto, desenvolver o CTDT como um applet Java traz ao software duas outras vantagens, ambas também ligadas ao quesito acessibilidade:

- Independência de Plataforma: o aplicativo roda em qualquer sistema que tenha máquina virtual Java instalada;
- Acessível pela Web: desenvolvido como um applet versão 1.0, tal software pode ser acessado por qualquer browser com suporte os applets, uma vez que os primeiros browsers com suporte a Java embutiam JVMs desta versão.

Tais características, combinadas com o suporte gráfico leve fornecido pelo VGJ, tornam CTDT uma ferramenta bem mais acessível ao usuário leigo do que o DBC, uma vez que este não precisará recompilar o código fonte caso saia de um sistema Windows para um Linux, não dependerá da disponibilização dos executáveis para sua plataforma específica - apenas precisará das classes compiladas do Java caso queira executar o programa localmente - e, se assim desejar, poderá abri-lo diretamente do seu browser sem necessitar de quaisquer conhecimentos de programação ou de ambiente de operação Java.

Dessa maneira, verifica-se que a escolha da linguagem de programação Java e da integração com o ambiente VGJ foram as grandes responsáveis pelos principais avanços no CTDT em relação ao DBC - tais avanços ligados ao aspecto acessibilidade. Não percebem-se avanços significativos em relação a aspectos funcionais, posto que não ampliaram-se os conceitos categoriais tratados pela ferramenta.

2.4 Graphical Database for Category Theory - GDCT [BRA2002]

Terceiro simulador categorial desenvolvido pela equipe de Robert Rosebrugh, esse é o mais recente dentre os desenvolvidos pelo grupo. A versão 1.1 do aplicativo foi desenvolvida de maio a agosto de 1999 por Bradbury e por Robert Rosebrugh - os mesmos autores do CTDT. A versão 2.0, disponibilizada no ano de 2002 - e sobre a qual versa esta avaliação -, contou também com a participação de Ian Rutherford e Matthew Graves.

Esse aplicativo, seguindo os passos do antecessor, foi desenvolvido em Java. Entretanto, deixou de ser implementado como um applet e passou a ser uma aplicação. Dessa maneira, há um retrocesso no quesito da acessibilidade, pois agora não é possível executar esse simulador categorial a partir de um browser. Agora, torna-se necessário instalar o aplicativo na máquina local, o que no CTDT era apenas uma opção.

Em outro aspecto, contudo, houve significativo avanço. Enquanto CTDT possui um esqueleto gráfico com fortes características textuais, aonde o verdadeiro avanço visual veio da integração com o VGJ, o GDCT possui características gráficas significativamente

mais consistentes, embora ainda conviva com tratamentos textuais. Se há um aspecto no qual se possa fazer uma análise evolutiva entre as ferramentas desenvolvidas pelo grupo canadense, este é o aspecto da transição de um software com traços mais textuais para outro, com traços menos textuais.

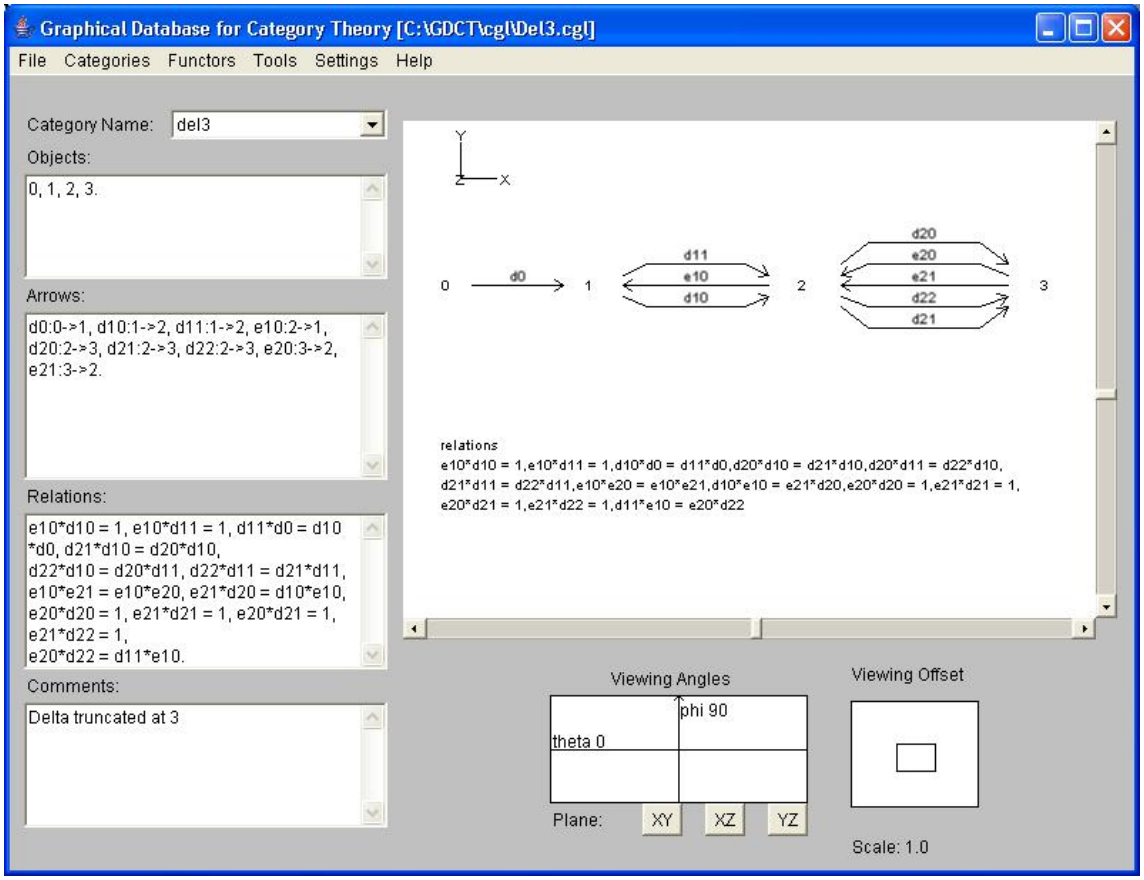


Figura 2.7: Tela do *Graphical Database for Category Theory* (GDCT)

Tal situação pode ser vista na figura 2.7. Nesse software já encontramos menus gráficos substituindo antigos menus textuais. De fato, o que se observa é o abandono do antigo arcabouço gráfico e a busca de uma maior integração dos elementos utilizados no DBC e no CTDT diretamente no ambiente do VGJ. Houve também um incremento nos recursos disponíveis, o que torna o GDCT um software sensivelmente mais robusto que seus antecessores.

Tanto o DBC quanto o CTDT possuem a capacidade de carregar categorias armazenadas em arquivos no formato CAT (formato criado pelo grupo canadense), o qual salva informações da categoria sem incluir informações gráficas. O GDCT continua trabalhando com esse formato de arquivo, mas incluiu uma alteração: antes, a composição $f \circ g = h$ era representada por `fg = h`. Agora, a mesma informação é representada por `f*g = h`. Como tal formato não inclui informações gráficas, o GDCT desenha a categoria em um formato padrão.

O VGJ permite o armazenamento em arquivo texto dos grafos criados. Para isso, ele utiliza um formato chamado de GML (*Graph Modelling Language*), o qual armazena informações ligadas à geometria espacial do grafo. O GDCT, a fim de utilizar tal recurso, criou o formato CGL, que é formado pela concatenação do formato CAT com o formato

GML. O software ainda permite que informações sejam salvas em formato GML - informações gráficas sem conteúdo categorial -, mas não permite abrir arquivos nesse formato.

Observando o grafo representada na figura 2.7 - disponível entre os arquivos de exemplo do GDCT - observa-se uma maneira muito peculiar de representar composições e de omitir identidades. O resultado da composição dos morfismos e10 e d10 foi definido como sendo a identidade do objeto 1. Essa informação é codificada na ferramenta como sendo `e10*d10 = 1`. Ao retornar um objeto como resultado de uma composição de morfismos, entende-se que está se referindo ao seu morfismo identidade. Podemos observar outra simplificação ao vermos uma composição representada por `d22*d11 = d21*d11`. Dessa maneira, sabemos que as composições d22∘d11 e d21∘d11 resultam no mesmo morfismo, o qual está omitido da estrutura.

Tais estratégias podem ser convenientes para o propósito de simplificar a estrutura representada, mas torna a criação de uma categoria um procedimento mais nebuloso e maximiza as possibilidades de representarmos uma estrutura não-categorial pensando tratar-se de uma categoria. Em termos educacionais, estratégias reducionistas como estas são bastante inadequadas, pois induzem o estudante a enxergar em uma categoria muito menos morfismos do que ela realmente deve possuir.

Tais problemas são os mesmos oriundos do DBC, posto que desde então não se verifica se uma determinada estrutura carregada é uma categoria. Portanto, evidencia-se que não é interesse do grupo canadense ter tal funcionalidade em suas ferramentas, e que se trabalha sempre com a presunção de que a estrutura carregada é, de fato, uma categoria.

Em termos funcionais, também houve um avanço significativo em relação as ferramentas anteriormente desenvolvidas pelo grupo. A nova ferramenta agrega:

- Isomorfismo: verifica se dois objetos - ou morfismos - selecionados são isomorfos; verifica se há na categoria objetos - ou morfismos - isomorfos a um selecionado; mostra todos os morfismos isomorfos não-triviais da categoria;
- Objeto Inicial: selecionados alguns objetos pertencentes a uma categoria, informa quais são iniciais e quais não são;
- Epimorfismo: verifica se um dado morfismo é epimorfismo; mostra todos os epimorfismos não triviais;
- Coequalizador: dados três morfismos, verifica se o terceiro é o morfismo resultante da coequalização dos dois primeiros; dados dois morfismos e um objeto, mostra todos os morfismos resultantes da coequalização dos morfismos dados que tenham destino no objeto dado;
- Coproduto: mostra os coprodutos existentes entre dois objetos dados;
- Objeto Terminal, Monomorfismo, Equalizador, Produto: funcionalidades duais às do objeto inicial, epimorfismo, coequalizador e coproduto, respectivamente.

Todas as funcionalidades acima estão diretamente disponíveis no menu `Tools`. Convém ressaltar que todas as funcionalidades disponíveis nas ferramentas anteriormente desenvolvidas pelo grupo canadense continuam existindo nesta.

A imprecisão conceitual também é ampliada. Em uma dada categoria onde A, B e C são objetos e f,g:A→B e h:B→C são morfismos, se ⟨B, h⟩ for o coequalizador de f,g:A→B, o programa retornará `h is a coequalizer` - ou seja, apenas o morfismo, e não o cone que representa o equalizador.

2.5 SimCat [MOU2001]

Trabalho desenvolvido no contexto de um Trabalho de Diplomação em Informática Teórica, SimCat é o primeiro simulador categorial desenvolvido no Laboratório de Fundamentos da Computação (LFC) - grupo de pesquisa no qual este autor encontra-se inserido -, um grupo de pesquisa do Instituto de Informática da UFRGS. Esse software foi desenvolvido por Tiago Mourão.

A base do SimCat vem de um trabalho desenvolvido por Tiago na disciplina de Teoria das Categorias do Instituto de Informática da UFRGS, onde foi solicitada a criação de uma software que permitisse representar os conceitos categoriais, de tal forma que fosse capaz de trabalhar:

- verificação se um grafo dado constitui uma categoria;
- criação da categoria dual;
- determinação de objetos inicial e terminal;
- determinação de morfismos mono e epi;
- cálculo de produtos de aridade máxima dois;
- cálculo de equalizadores.

Tiago Mourão desenvolveu um dos melhores aplicativos dentro do contexto exigido. No semestre seguinte ao que cursou a disciplina, iria se formar. Então, o grupo LFC apresentou-lhe o projeto e os objetivos de uma ferramenta categorial que estava começando a ser desenvolvida no grupo, chamada de CaTLeT - a qual será tratada mais adiante neste trabalho. Convidou-o então a integrar a equipe e tomar parte especificamente nesse projeto, propondo a ele um objetivo específico: o tema de seu Trabalho Final de Graduação seria a implementação de uma nova ferramenta cuja abrangência conceitual e funcionalidade seria superior à por ele desenvolvida anteriormente na disciplina. SimCat, portanto, nasceu como um laboratório experimental para o ideário conceitual e funcional do CaTLeT [PFE2002].

SimCat foi elaborado como um applet Java versão 1.3. Como já vimos ao falarmos do CTDT, tal escolha agrega ao software independência de plataforma e acesso via web. Entretanto, é necessária a instalação da JVM para que seja possível utilizar a ferramenta localmente. Para acessar o applet via web, é preciso instalar devido plug-in no browser desejado.

O ambiente de operação é gráfico, conforme pode ser observado na figura 2.8. A criação dos objetos se dá com um clique na área de trabalho e a dos morfismos, com um clique segurando o botão sobre um objeto e largando o botão sobre outro objeto. Se a correspondente opção estiver marcada, ao inserir um objeto, o software automaticamente insere o correspondente endomorfismo identidade. Caso uma outra específica opção estiver marcada, ao inserir um novo morfismo, o SimCat verificará se este não tornou-se componível com outros já existentes, inserindo, assim, a correspondente composição. Enquanto o grafo desenhado na área de trabalho corresponder a uma categoria, a ferramenta manterá escrita no segundo campo da barra de status a palavra `Categoria`. Caso contrário, nada estará escrito nesse campo, e no último campo aparecerá escrito o motivo - ou um motivo - o qual torna a estrutura representada um grafo não-categorial.

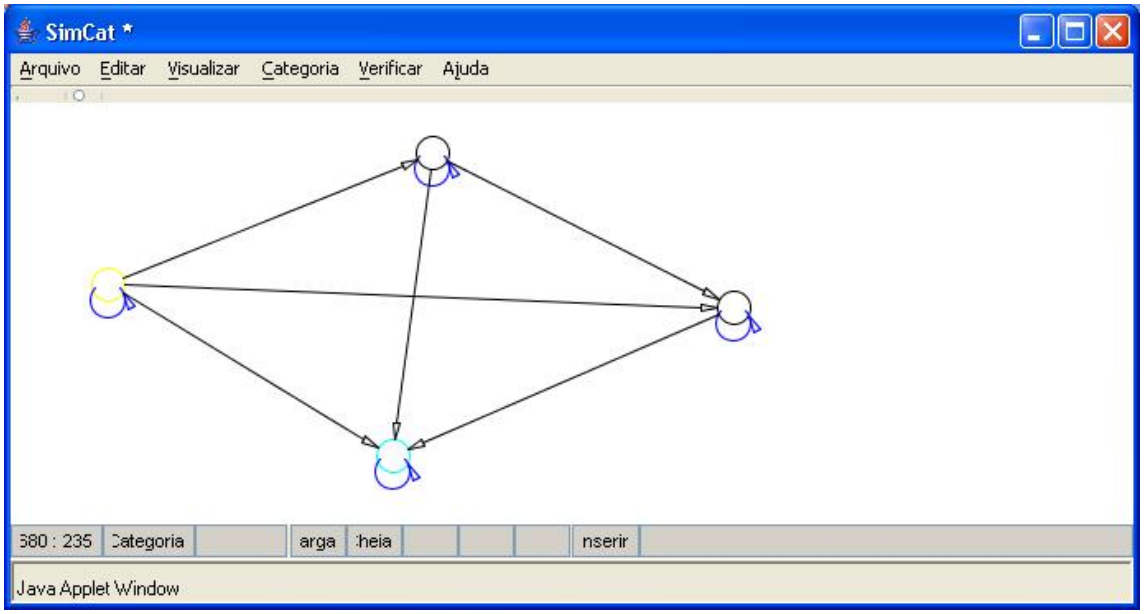


Figura 2.8: Tela do *SimCat*

Observa-se, portanto, que não só o SimCat é uma ferramenta que avalia permanentemente a estrutura que ele recebe de modo a avaliar se é uma categoria - algo que nenhuma das ferramentas do grupo canadense faz - como também possui instrumentos para induzir o usuário a realizar construções categoriais.

Além disso, a forma como o usuário interage com o sistema o leva a usufruir uma das grandes vantagens de Teoria das Categorias para Ciência da Computação, que é a notação gráfica. Tal situação facilita ao usuário a compreensão de seus componentes e de seus relacionamentos, ajudando na percepção visual intuitiva dos conceitos categoriais.

O SimCat explora os recursos visuais de modo a buscar informar ao usuário características categoriais em tempo de edição. A cada passo da construção categorial é possível observar quais são os objetos iniciais, terminais e zeros de acordo com a cor do contorno do objeto. O mesmo ocorre para destacar os morfismos identidade. Se abandonarmos o modo de inserção e utilizarmos o modo seleção, podemos começar a selecionar objetos e morfismos de modo a destacarmos uma subcategoria dentro da categoria representada. Enquanto as estruturas selecionadas não corresponderem a uma categoria, o software informa. Quando esta, de fato, constituir-se uma categoria, o software imediatamente informa na barra de status se esta subcategoria é larga ou cheia. Caso selecione-se apenas um morfismo, observam-se na barra de status suas propriedades (mono, epi e iso).

Além das verificações que o SimCat disponibiliza em tempo de edição, há uma série de propriedades que podem ser averiguadas no seu menu. Além dos itens solicitados no trabalho da disciplina de Teoria das Categorias - já listados anteriormente -, o SimCat oferece suporte para a verificação dos seguintes conceitos categoriais:

- objeto zero;
- isomorfismo;
- coequalizador;
- produto fibrado e soma amalgamada;

- cone e cocone;
- limite e colimite.

Para possibilitar a realização de alguns desses cálculos, o software permite abrir uma nova janela para realizar a edição de um diagrama. Dessa maneira, é possível construir cones e averiguar limites.

Se ainda trata-se de um software limitado e que permite ampliações que o tornem bem mais poderoso, é inegável que, em comparação aos aplicativos desenvolvidos pelo grupo canadense, há um ambiente de trabalho bastante mais amigável, com mais funcionalidades e - o que é mais importante - com precisão conceitual e com estruturas explícitas e, portanto, menos confusas - visto que o SimCat não omite identidades ou composições.

Se por um lado o caminho escolhido nos leva a ter um ambiente gráfico mais poluído quando se deseja representar categorias com maior número de elementos - a ponto de tornar-se inviável manipular elementos ou ter uma mínima visualização e operacionalidade -, o caminho seguido pelos canadenses leva a uma situação ainda mais alarmante: o se ser inviável saber se a estrutura representada é, de fato, uma categoria.

Cabe ressaltar, entretanto, duas deficiências de SimCat em relação a todas as ferramentas vistas até aqui: não trabalha com funtores e não é capaz de lidar com múltiplas categorias abertas (somente uma por vez).

2.6 Category Theory Learning Tool - CaTLeT [PFE2002][VIE2003]

O código fonte dessa ferramenta, a segunda elaborada no LFC, serviu de base para aquela desenvolvida no contexto desta dissertação. O CaTLeT foi desenvolvido por Fábio Victor Pfeiff como parte integrante de sua dissertação de mestrado e contou com resultados obtidos no desenvolvimento do SimCat - incluindo parte de seu código fonte - para servirem como base experimental que o auxiliaram a nortear seu projeto.

Vislumbrando o potencial de aplicabilidade que Teoria das Categorias poderia ter na área aplicada, mas que não atingia em virtude de diversas barreiras, Pfeiff decidiu elaborar uma ferramenta que servisse de auxílio para o aprendizado de seus conceitos introdutórios. A intenção era oferecer aos interessados um meio gráfico para facilitar o acesso aos conceitos básicos de Teoria das Categorias, de tal forma que servisse como uma ferramenta de propagação dos conceitos categoriais entre os leigos.

Implementada como um applet Java versão 1.3 - assim como sua antecessora -, CaTLeT é acessível pela Internet e independente de plataforma. O trabalho desenvolvido por Pfeiff levanta a possibilidade de integração com o Hyper-Automaton [MAC2000], um trabalho que apresenta uma forma de modelagem de cursos para o ambiente Web utilizando autômatos finitos determinísticos como saída. Esta técnica está direcionada para o suporte de cursos remotos em um sistema semi-automatizado para o ensino de Informática Teórica no Instituto de Informática da UFRGS. Segundo é explanado em [PFE2002], o CaTLeT possui características adequadas, por ser um applet Java, para prestar-se a integração em cursos dessa natureza.

Na figura 2.9 temos uma categoria desenhada no CaTLeT e, nela selecionados, um morfismo e seus objetos origem e destino. A estrutura selecionada está representada em separado como um diagrama, na janela intitulada `Diagram Viewport`. Assim, é que se torna possível realizar determinados cálculos que requerem um diagrama - como, por exemplo, limite. Tal representação é similar à disponível no SimCat, entretanto, com algumas vantagens: enquanto no SimCat o diagrama é montado somente a partir da seleção

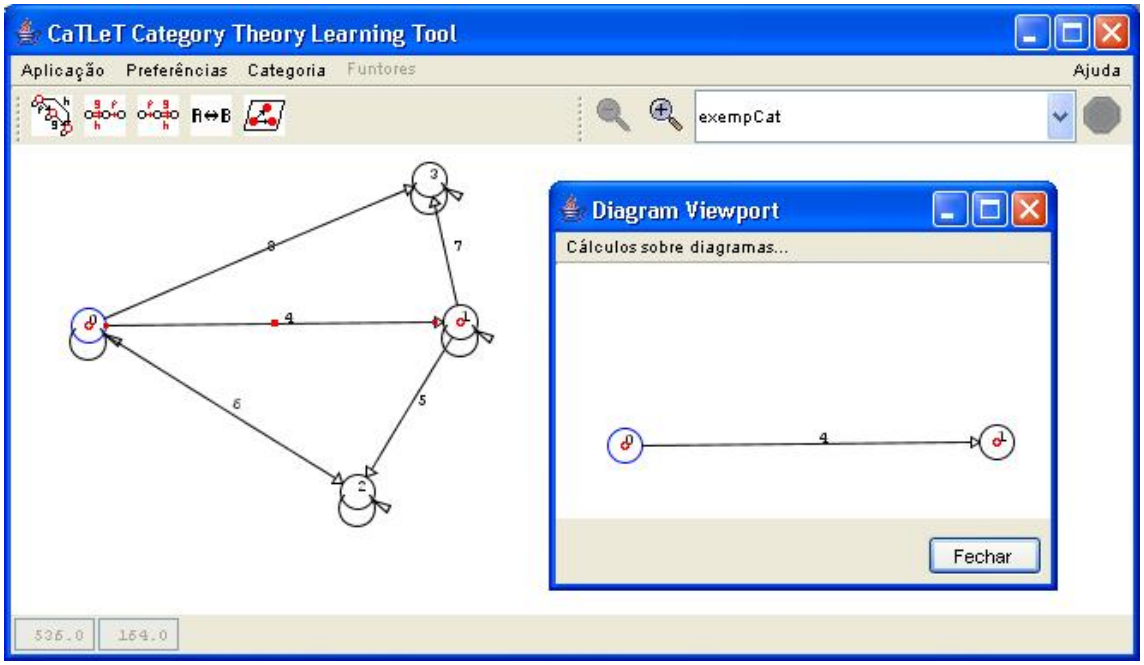


Figura 2.9: Tela do *Category Theory Learning Tool* (CaTLeT)

do objetos e morfismos da categoria e não é editável, no CaTLeT é possível editá-lo, tendo a possibilidade de repetir objetos e morfismos. Tal abordagem é mais adequada em termos conceituais, pois em um diagrama é possível repetir elementos e morfismos.

O CaTLeT é capaz de tratar múltiplas categorias. Tal situação também possibilitou a representação de funtores na ferramenta. Entretanto, como é reconhecido em [PFE2002], pouco tempo do projeto foi dedicado para o tratamento de funtores e, portanto, não há um tratamento mais aprimorado para a visualização de funtores. Da mesma maneira, não há verificação de propriedades funtoriais no CaTLeT.

Com exceção do Computational Category Theory, todas as ferramentas descritas até aqui trabalham somente com objetos e morfismos atômicos - sem estrutura interna. Em cima destes é construída uma categoria finita e se verificam propriedades. Em 2003, este autor realizou uma ampliação no CaTLeT em virtude da realização do Projeto de Diplomação [VIE2003]. A partir dessa ampliação, o CaTLeT passou a tratar objetos e morfismos como sendo conjuntos e relações entre conjuntos, respectivamente. Dessa maneira, foi criada a primeira ferramenta gráfica que simula categorias estruturadas do grupo LFC. Assim, se definirmos a categoria FinRel com sendo a categoria que possui como objetos todos os conjuntos finitos e como morfismos todas as possíveis relações entre conjuntos finitos, o CaTLeT ampliado passa a permitir a representação de subcategorias finitas de FinRel - dentro das limitações de memória e espaço da máquina.

Ao desenvolver essa ampliação, esse autor trabalhou com o fato de que, na categoria FinRel, é possível avaliar se dois objetos são isomorfos a partir de sua cardinalidade. É possível provar que, em FinRel, dois conjuntos de mesma cardinalidade sempre são isomorfos e dois conjuntos de cardinalidades diferentes nunca são isomorfos. Sabendo que, em Teoria das Categorias, objetos isomorfos são considerados essencialmente o mesmo, não sendo, em geral, distinguidos [MEN2001], decidiu-se representar a estrutura interna de um objeto apenas através da cardinalidade do conjunto que representa. Assim, ao criar-se um novo objeto, o CaTLeT passa a perguntar para o usuário qual o número de

elementos que este possui.

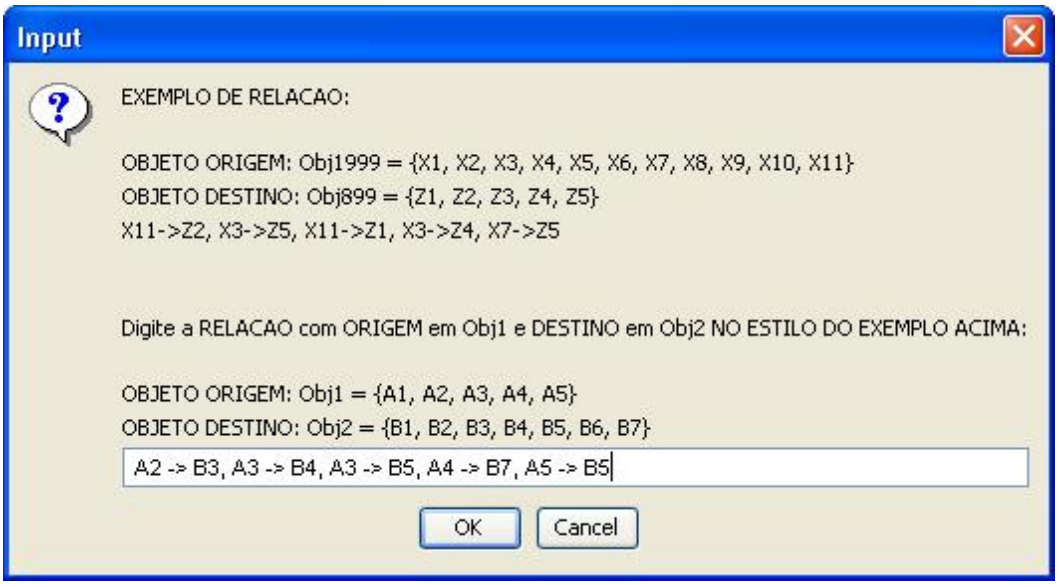


Figura 2.10: Tela de diálogo para entrada da relação entre os conjuntos Obj1 (de cardinalidade 5) e Obj2 (de cardinalidade 7)

Ao criar-se uma relação entre conjuntos, o software lida com os elementos do conjunto como sendo formados por uma letra derivada no número que identifica o objeto (cada número gera uma letra distinta) e um indexador. Assim, é possível estabelecer os pares ordenados da relação entre os conjuntos. O diálogo utilizado para tal entrada está representado na figura 2.10. O software realiza verificações de consistência - por exemplo, não é possível entrar com dois pares ordenados iguais.

Outro avanço do CaTLeT que pode ser percebido em relação ao SimCat refere-se às mensagens que o aplicativo retorna ao detectar que uma estrutura criada não é uma categoria. No SimCat, essa mensagem aparece na barra de status, e caso sejam vários os motivos que levem a um determinado grafo desenhado não corresponder a uma categoria, o programa informa apenas o primeiro motivo detectado. No CaTLeT, a informação referente ao grafo ser ou não ser categorial é disponibilizada em um botão com o desenho de um octágono, que encontra-se na barra de ferramentas (fica à direita da combo-box que contém as categorias ativas). Se esse botão encontra-se cinza, significa que ele está inativo e que, portanto, a estrutura representada continua sendo uma categoria. O botão passa a ficar ativo - com a cor vermelha - quando o grafo representado deixa de ser categorial. Nesse caso, se o usuário clicar no botão, o programa irá expor uma lista com todos os elementos que faltam no grafo desenhado para que este se torne uma categoria.

Por fim, cabe ressaltar um acréscimo funcional que distingue o CaTLeT das demais ferramentas estudadas - com exceção da Computational Category Theory. Esta é a capacidade de realizar o cálculo de produto com qualquer aridade inteira maior ou igual a zero.

2.7 Comparativo Entre as Ferramentas Estudadas

Nesta dissertação, julga-se que três são os critérios que devem ser perseguidos na elaboração de um simulador categorial: acessibilidade (facilitar ao máximo o acesso ao

software e às estruturas categoriais nele implementadas), relevância das estruturas implementadas e cobertura (dentro do escopo tido como relevante, implementar o maior número possível de conceitos categoriais). É sob a luz desses critérios que será feita a comparação entre os simuladores categoriais vistos.

2.7.1 Acessibilidade

Em termos de acessibilidade, fica claro que interface gráfica e acesso via web são dois pontos que colaboram bastante para diminuir as barreiras entre usuário e software. Entretanto, ao falar de interface gráfica está se falando do uso efetivo que seus recursos visuais oferecem. Caso contrário, podemos chegar à estranha situação de construir uma ferramenta com recursos gráficos menos intuitiva que uma ferramenta textual bem elaborada.

Sob alguns aspectos é o que acaba se observando com o DBC e com o CTDT. Um desses aspectos é ressaltado em [PFE2002] e diz respeito ao esquema de índices associados às categorias abertas utilizado pelo DBC. Esse esquema facilita as interações com o usuário, mas foi abandonado pelo CTDT.

Aliás, o arcabouço gráfico utilizado pelo CTDT é uma prova de que é possível criar-se um software em nesse tipo de ambiente sem, contudo, agregar facilidade de uso. Se não fosse a integração feita entre o CTDT e o VGJ, seria possível questionar se o CTDT evoluiu em termos de acessibilidade em relação ao seu antecessor, posto que a forma de interação com o módulo elaborado pelo grupo canadense é, na maior parte do tempo, tipicamente textual.

A visualização das estruturas categoriais de maneira visual, como já foi dito anteriormente, explora uma das vantagens do uso de Teoria das Categorias para a Ciência da Computação: a notação gráfica. A percepção de maneira visual das construções e conceitos categoriais facilita o seu aprendizado, tornando essa teoria abstrata um pouco mais concreta e tangível. Nesse quesito, destacam-se o SimCat e o CaTLeT, que desde a entrada de dados até a avaliação de propriedades é feito uso de cores e mensagens em tempo de construção para deixar o mais visual possível os conceitos implementados. As outras duas ferramentas que fazem uso de representação visual de categorias - o CTDT e especialmente o GDCT, ambos apoiados na integração com o VGJ - deixam bastante a desejar. O recurso de visualização tridimensional, característica exclusiva dessas ferramentas em relação às demais estudadas, pouco acrescenta para a compreensão dos conceitos ou para a limpeza do ambiente. E a entrada e edição de dados nessas ferramentas utilizam mais uma interação textual do que uma interação visual.

Aspectos tipicamente ligados à linguagem de programação Java, independência de plataforma e acessibilidade via web estabelecem menos barreiras para que o usuário acesso o simulador. Nesse quesito, o CTDT leva vantagem, pois além de atender os dois quesitos citados não requer a instalação de plug-in para ser acessado em browsers. Em seguida, podemos citar o SimCat e o CaTLeT, que requer a instalação do plug-in adequado, mas ainda assim satisfaz os itens citados. O GDCT vem logo a seguir, pois é independente de plataforma mas não pode ser acessado pela Web.

O menos acessível, sem dúvida, é o Computational Category Theory. Além de requerer instalação de ambiente ML para funcionar e conhecimentos de operação de ambiente funcional, precisa ser compilado toda vez que se abre o ambiente e não conta com qualquer tipo de facilidade para entrar ou sair com dados.

Um aspecto não explorado por nenhuma das ferramentas é o uso de recursos tridimensionais para visualizar funtores, já que isso permitiria visualizar as categorias inscritas em

planos sob um ângulo perspectivo e as setas constituintes do funtor desenhados entre esses planos.

2.7.2 Relevância das Estruturas Implementadas

A maior dificuldade de avaliar esse aspecto é o fato de ele estar bastante ligado aos objetivos do projeto, o que não está muito claro na descrição de alguns projetos. Além disso, como há projetos com objetivos diferentes, a comparação fica dificultada. Em virtude disso, algumas avaliações poderão soar um pouco genéricas.

Um dos ambiciosos objetivos mencionados pelos autores do Computational Category Theory era que esse aplicativo servisse como base para futuras experiências científicas no campo da mecanização de Teoria das Categorias. Dessa maneira, esse autor acredita que, apesar das dificuldades de acessibilidade ligadas ao aplicativo desenvolvido, a amplitude de estruturas e conceitos agregados pela ferramenta era o requisito essencial para que o objetivo proposto pudesse ser tido como alcançado. Portanto, a quantidade e variedade de conceitos categoriais implementados, além de surpreendente para a época em que foi desenvolvido - embora mais compreensível quando se observa a facilidade de implementar construções categoriais no paradigma funcional -, foi de extrema relevância. Sobretudo porque eles abrangem grande parte, senão a maioria, dos conceitos categoriais pesquisados nas universidades.

As ferramentas implementadas sob a orientação de Robert Rosebrugh não deixam claro qual o objetivo que perseguem com suas ferramentas - apenas mencionam, em material que descreve o DBC, que “os autores acreditam que um sistema que permita armazenamento, recuperação, consultas e computações em objetos categoriais serão valiosos para a pesquisa e a instrução” (*A Database of Categories (pdf)*, disponível em [FLE95]). Entretanto, este autor acredita que, no mínimo, a precisão conceitual na implementação dos conceitos categoriais é relevante. Nesse sentido, estranha-se algumas simplificações utilizadas pelo software na implementação de categorias. Também se considera relevante que haja a checagem de estrutura entrada, a fim de verificar se esta é uma categoria. Todavia, tal verificação requer um rigor conceitual nas estruturas implementadas que este autor sente falta ao trabalhar com os softwares - até em pequenos diálogos, como foi mencionado em outros tópicos.

Os softwares desenvolvidos no contexto do LFC, os quais estão integradas ao mesmo projeto, tem como objetivo o desenvolvimento de ferramentas capazes de facilitar ao estudante de Teoria das Categorias acesso facilitado aos conceitos introdutórios. Em [PFE2002] verifica-se que o autor considera serem conceitos introdutórios:

- validação da construção (se é ou não categoria);
- objetos especiais (inicial, terminal e zero);
- morfismos especiais (mono, epi e iso);
- categoria dual;
- produto de qualquer aridade inteira maior ou igual a zero;
- equalizador;
- produto fibrado;
- cone;

- limite;
- funtor.

Observa-se que o conceito de subcategoria não é citado, embora este seja um dos primeiros conceitos estudados em Teoria das Categorias. Apesar disso, SimCat possui tratamento para esse relevante conceito - o que não acontece com o CaTLeT. Da lista apresentada, o SimCat não implementa produtos de aridade diferente de 2 e funtor - o que pode ser relevado pelo fato de ter sido considerado como um laboratório experimental do CaTLeT. Já o CaTLeT implementa, com maior ou menor profundidade - como veremos a seguir, ao analisar sua cobertura -, todos os conceitos listados. A ampliação do CaTLeT permitiu a observação de categorias estruturadas, o que pode ser considerado um tópico introdutório.

2.7.3 Cobertura

Uma análise mais aprofundada a respeito da cobertura do Computational Category Theory é dificultada pela quantidade de conceitos categoriais tratados pela ferramenta. Aprofundar-se em todos esses conceitos e explorá-los implicaria representar todas as categorias e conceitos tratados por boa parte dos livros de Teoria das Categorias. Uma falha mais perceptível é que, ao implementar do conceito de subcategorias, deixou de tratar a subcategoria larga. Entretanto, o programa é capaz de tratar categorias como FinSet - dada como exemplo em [RYD88] - e realizar sobre esta as avaliações categoriais modeladas - embora, muitas vezes, isso praticamente signifique fazer o usuário “programar” em ML uma categoria.

Os softwares desenvolvidos pela equipe canadense peca bastante pela precisão conceitual. Se é relevante implementar coproduto, não menos importante é cobrir esse conceito corretamente. Além disso, uma ferramenta que se propõe a realizar a tratar do conceito de categorias deveria cobrir conceitos como identidade e composição dentro do contexto de morfismos, não permitindo sua omissão - ou, pelo menos, admitindo a opção de enxergá-los visualmente. Mais questionável é a forma como o aplicativo cobre o conceito de identidade como resultado de uma composição, no qual ela é representada como sendo o objeto a qual pertence. Este autor sente uma falta de coerência semântica mais clara nos conceitos que as ferramentas se dispõe a abordar.

Os aplicativos desenvolvidos no LFC poderiam cobrir mais profundamente alguns conceitos que implementam. No SimCat, por exemplo, o fato de não ser possível representar objetos e morfismos repetidos no diagrama implica uma limitação nos cálculos ligados ao diagrama. Esse problema não existe no CaTLeT. Entretanto, o fato de CaTLeT permitir averiguação de propriedades em categorias com relações como morfismos sugere que, com um esforço reduzido, seria possível cobrir outras estruturas advindas de restrições sobre relações - como funções parciais, por exemplo.

3 ESPECIFICAÇÃO DA FERRAMENTA DESEJADA

Para atender às metas estabelecidas, é necessário que a ferramenta projetada seja útil tanto para o ensino e aprendizado de conceitos intermediários de Teoria das Categorias quanto servir como uma "calculadora categorial" eficiente para que um pesquisador nessa área consiga utilizá-la como ferramenta de suporte.

Para que sirva a propósitos educacionais, o quesito acessibilidade - especialmente no que se refere à usabilidade - é fundamental. Tanto no ensino a distância quanto no laboratório da universidade, a ferramenta deve apresentar o menor número de barreiras técnicas possíveis, levando o aluno a focar sua atenção nos conceitos categoriais implementados.

Para que sirva a propósitos científicos de um pesquisador, os quesitos relevância e, sobretudo, cobertura são especialmente importantes. Um simulador categorial será capaz de ajudar em uma pesquisa científica se ele puder, de alguma maneira, ser utilizado para, no mínimo, evidenciar uma hipótese. É especialmente importante que este programa trabalhe com funtores e cálculos funtoriais, já que são estes que conferem à Teoria das Categorias seu poder de expressão. O aspecto acessibilidade, embora seja acessório neste contexto, diminui barreiras para o uso efetivo da ferramenta por pesquisadores das diversas áreas. Em especial, pesquisadores da área aplicada da Ciência da Computação que tenham interesse em utilizar os recursos de Teoria das Categorias em seus projetos encontrarão em uma ferramenta acessível um apoio para vencer eventuais barreiras de entendimento teórico sem, no entanto, esbarrar em dificuldades técnicas sem importância (aspecto similar ao do propósito educacional).

Observadas tais questões, e levando em consideração o estudo apresentado no capítulo anterior, este autor está convencido que uma ferramenta bastante satisfatória seria aquela que agregasse a acessibilidade do CaTLeT e a funcionalidade do Computational Category Theory. O projeto do CaTReS - ferramenta desenvolvida no contexto desta dissertação -, embora não tenha como meta integrar essas duas ferramentas, buscou agregar algumas qualidades presente nelas.

3.1 Aspectos Técnicos

A pesquisa sobre as ferramentas apresentadas no capítulo anterior convenceu este autor que o paradigma de programação funcional é o que mais naturalmente implementa os conceitos categoriais. Tendo como referencial teórico o cálculo λ tipado, as linguagens funcionais trabalham essencialmente com definição e aplicação (ou avaliação) de funções. Os conceitos de categoria, de funtor e dos cálculos e conceitos envolvendo essas duas construções são, em grande parte, definidos utilizando funções. Um exemplo disso é o conceito de categoria, que é definida como uma sêxtupla ordenada composta por duas

coleções - que podem ser vistas como funções constantes -, três funções aplicadas sobre estas coleções e uma função parcial aplicada sobre o produto cartesiano dessas coleções - que pode ser modelada como uma função que retorna, por exemplo, `não_componível` para os casos de indefinição.

Entretanto, as linguagens funcionais não são as que mais oferecem recursos que auxiliem na acessibilidade do software implementado. Tais linguagens costumam gerar aplicativos que necessitam de ambientes de operação especiais, que dependem de plataforma e que não são operáveis pela web, além de, eventualmente, necessitarem de compilação sempre que tal software for operado - como é o caso do Computational Category Theory. Além disso, para que seja elaborada uma interação minimamente amigável entre homem e máquina é necessário um esforço extra de programação que, muitas vezes, não é trivial em linguagens funcionais. Como os objetivos deste projeto não permitem prescindir da acessibilidade, o uso de uma linguagem funcional foi logo descartado.

A partir daí, foi definido que o programa seria desenvolvido como um applet Java - utilizando o compilador disponível no Java 2 Software Development Kit (SDK) Standard Edition versão 1.4.2_04, da Sun - em virtude das seguintes questões:

- Independência de Plataforma: a compilação de um código fonte Java gera um arquivo com extensão ".class", o qual não é diretamente executado pelo computador. Para executar o programa, é necessário que esteja instalada no computador a Máquina Virtual Java (JVM). A JVM é um programa implementado em diversas plataformas a partir do qual é possível executar softwares desenvolvidos em Java. Assim, com o mesmo conjunto de arquivos ".class" já compilados, é possível realizar a execução em qualquer plataforma onde a JVM estiver instalada, o que garante a independência de plataforma - o mesmo aplicativo compilado pode ser executado em diversos sistemas operacionais;

- Execução Através de da Web: um *applet* é um programa Java que executa em um navegador de *World Wide Web* (WWW) com suporte a Java. O navegador executa um applet quando um documento HTML contendo o applet é aberto no navegador [DEI2001]. Os primeiros browsers com suporte a Java embutiam JVM com suporte a Java versão 1.0. Versões posteriores - como a deste trabalho - requerem que a JVM esteja instalada na máquina. Assim, programar um software como um applet significa permitir que ele possa ser executado a partir de um browser, sem necessitar instalá-lo na estação de trabalho - mesmo porque, o usuário pode não ter privilégio no sistema operacional que lhe permita instalar programas;

- Linguagem Orientada a Objetos: apesar da maneira que se programa neste tipo de linguagem não possuir a naturalidade para codificar Teoria das Categorias que linguagens funcionais possuem, as vantagens da análise, projeto e programação orientadas a objeto, bem conhecidas no meio acadêmico e no mercado de tecnologia da informação, justificam sua escolha em detrimento de outros paradigmas não-funcionais. Dentre essas vantagens estão o reuso de software por herança e, em especial, o encapsulamento de operações;

- Integração com Ambiente Hyper-Automaton [MAC2000]: trata-se de um Sistema de Gerenciamento para Ensino a Distância (SGEAD) onde todo conteúdo é armazenado na forma de pequenos scripts HTML, os quais são fornecidos e combinados com o instrutor. Assim, implementar um simulador categorial como um applet Java permite que este seja integrado ao Hyper-Automaton, uma vez que a chamada

de um applet consiste simplesmente de uma marca HTML. Dessa maneira, pode-se armazenar no ambiente Hyper-Automaton unidade instrucionais para cursos de Teoria das Categorias que são apenas marcas parametrizadas para a execução do simulador. Tal integração serve aos propósitos educacionais da ferramenta;

- CaTLeT é um applet Java: os simuladores categoriais desenvolvidos no LFC, grupo ao qual este autor pertence, possuem virtudes de acessibilidade e de precisão conceitual que, em conjunto, não encontram paralelo nas outras ferramentas pesquisadas. Dessa maneira, decidiu-se utilizar o CaTLeT como um conveniente ponto de partida, a fim de aproveitar suas qualidades e não "reinventar a roda".

3.2 Funcionalidades

O *Category Theory Researching Software* (CaTReS), terceira ferramenta desenvolvida no LFC, foi especificado para conter os seguintes avanços em relação ao seu antecessor:

- Representar a estrutura dos objetos não somente através de sua cardinalidade, mas também através de um conjunto de elementos dados pelo usuário como entrada;
- Selecionados dois objetos estruturados (representando conjuntos finitos), a ferramenta deverá ser capaz de construir todas as relações possíveis de um objeto para o outro;
- A partir de uma categoria aberta, o CaTReS deverá ser capaz de, a partir de um comando do usuário, formar uma segunda categoria, a qual tenha como objetos os mesmos presentes na primeira e como morfismos todas as possíveis relações entre quaisquer dois objetos da categoria;
- Deverá ser possível apagar morfismos ou objetos de uma categoria aberta. No segundo caso, devem ser apagados também os morfismos cuja origem ou destino sejam um dos objetos removidos;
- Permitir que objetos e morfismos representem outras estruturas além daquelas já suportadas pelo CaTLeT - por exemplo, morfismos representando funções finitas e funções parciais finitas -, oferecendo ao usuário a possibilidade de escolher, quando este for criar uma nova categoria, qual estrutura os objetos e morfismos representarão;
- Como uma ferramenta de apoio à pesquisa, deve ser capaz de representar estruturas funtoriais, já que são estas que dão o poder de expressão para tratar de maneira uniforme modelos matemáticos distintos, cada qual representado através de uma categoria;
- Ser capaz de formar todos os funtores possíveis entre duas categorias selecionadas pelo usuário;
- Ser capaz de construir todos os funtores possíveis entre as categorias abertas;
- Possibilidade de verificar se as categorias e funtores abertos formam uma categoria de categorias;
- Verificar quais são os funtores identidade de cada categoria;

- Sendo selecionados dois funtores componíveis, ser possível criar um funtor composição ou indicar sua existência;
- Ter a possibilidade de apresentar ao usuário a lista de composição de funtores;
- Se as categorias e funtores abertos formarem uma categoria de categorias, então o software deverá ter a possibilidade de realizar o cálculo de dualidade, assim como a verificação de propriedades categoriais já existentes no software, só que agora trabalhando no escopo de categorias de categorias;
- Identificar se os funtores abertos são fidedignos ou plenos;
- Devido à relevância que transformação natural possui para a pesquisa em Teoria das Categorias, a ferramenta deverá ter suporte a transformações naturais, incluindo o cálculo das composições vertical e horizontal;
- Também em virtude de sua relevância na pesquisa, o tratamento de adjunções deverá ser contemplado pela ferramenta;
- Da mesma maneira, o tratamento de mônadas será inserido no CaTReS.

Uma das decisões de projeto tomadas foi a de não retomar a possibilidade de representar objetos e morfismos como estruturas atômicas, situação presente no CaTLeT antes da ampliação realizada em [VIE2003], onde tal funcionalidade foi abandonada.

As cinco primeiras funcionalidades dizem respeito à ampliação do suporte categorial que a ferramenta fornece, enquanto as outras doze referem-se à inserção de tratamento de funtores no aplicativo. Todavia, a quinta funcionalidade já pode ser vista como a primeira perna da estrutura funtorial disponível na ferramenta, uma vez que, no CaTReS, o funtor é visto graficamente como se fosse um morfismo comum ligando objetos - os quais, neste caso, são categorias já representadas pelo usuário. Portanto, duas das outras estruturas que o aplicativo suporta é objetos representando categorias e morfismos representando funtores entre categorias.

Há duas razões que explicam a ênfase em tratamento funtorial nas funcionalidades citadas:

- O CaTReS herdou do CaTLeT um tratamento categorial pronto, enquanto as operações ligadas a funtores tiveram que ser criadas praticamente do zero;
- Embora este trabalho pretenda que o objetivo do criador do CaTLeT - que o simulador sirva de suporte ao ensino e ao aprendizado - seja mantido e, inclusive, ampliado (oferecendo suporte a conceitos intermediários, e não apenas introdutórios), este software tem como prioridade a inserção de mecanismos úteis para o pesquisador em Teoria das Categorias - o que, sem dúvida, nos leva ao tratamento de funtores.

Embora seja verdade que, em termos funcionais, o software projetado não supera o Computational Category Theory, também é verdade que o CaTReS é, no conjunto das qualidades que agrega, um software bem mais completo e mais útil. Enquanto Computational Category Theory só pode ser utilizado por usuários experientes em linguagem funcional - mais especificamente, em ML -, o CaTReS possui a virtude de tratar de muitos dos conceitos do Computational Category Theory - e outros que não são nele tratados,

como, por exemplo, funtores fidedignos ou plenos - de maneira acessível, usando modernos recursos gráficos e de acesso pela web, investindo, assim, na qualidade da interação homem-computador. Tais vantagens, como já foi citado, não se mostram úteis apenas para o aprendizado de Teoria das Categorias, mas também para auxiliar pesquisadores da área aplicada que tenham interesse em utilizares conceitos categoriais em seus projetos. Este autor não encontrou outro simulador categorial que agregue tais virtudes.

3.3 Limitações

Há funcionalidades que, embora de reconhecida utilidade para a Teoria das Categorias - especialmente no campo da pesquisa -, não são tratados pelo CaTReS. Podemos citar, por exemplo:

- Permitir que objetos representem conjuntos infinitos e que morfismos também possam representar estruturas infinitas;
- Permitir que categorias possuam infinitos objetos e morfismos;
- Permitir que objetos representem estruturas como, por exemplo, monóides e grafos;
- Permitir verificação automática de propriedades categoriais e funtoriais;
- Já que objetos são vistos como conjuntos e que morfismos podem ser vistos relações, oferecer suporte para que a estrutura construída possa ser vista como um banco de dados relacional, permitindo, assim, que fossem agregadas à ferramenta conceitos oriundos do cálculo relacional, da álgebra relacional e do SQL.

Quando estamos falando em uma ferramenta que ofereça suporte à pesquisa, sempre é possível encontrar novas funcionalidades relevantes para inserir no programa. Foram citadas cinco funcionalidades, mas certamente existem muitas outras de grande utilidade para os objetivos traçados nesta dissertação. No contexto de uma dissertação de mestrado, é impossível implementar uma ferramenta que agregue todas as funcionalidades que potencialmente possam ser úteis para pesquisadores.

Portanto, torna-se necessário lembrar que esta ferramenta não se propõe a ser uma solução final para o pesquisador que tiver interesse em Teoria das Categorias, mas sim uma ferramenta de apoio que possa ser útil a determinadas pesquisas. Cabe lembrar que, até onde se sabe, implementar uma ferramenta com este porte e essas características constitui experiência inédita, situação que implica desafios e dificuldades.

Tabela 3.1: Comparativo entre o CaTReS e as ferramentas categoriais estudadas

	CCT	**DBC**	**CTDT**	**GDCT**	**SimCat**	**CaTLeT**	**CaTReS**
Interface Gráfica	-	-	+ / -	+	+	+	+
Independ Plataf	-	-	+	+	+	+	+
Acesso via Web	-	-	+	+	+	+	+
Amigabilidade	-	+ / -	+ / -	+	+	+	+
Multipl Categ	+	+	+	+	-	+	+
Valida Constr	-	-	-	-	+	+	+
Sup Corr Erro	-	-	-	-	+	+	+
Sup Obj-Conj	+	-	-	-	-	+ / -	+
Sup Obj-Cat	+	+	+	+	-	-	+
Sup Morf-Rel	+	-	-	-	-	+ / -	+
Sup Morf-Funç	+	-	-	-	-	-	+
Sup Mor-FParc	+	-	-	-	-	-	+
Sup Funtor	+	+	+	+	-	-	+
Sup Prop Rel	+	-	-	-	-	-	+
Sup Calc Funt	+	-	-	-	-	-	+

4 ALGORITMOS E CONCEITOS MATEMÁTICOS

Neste capítulo, serão apresentadas as linhas gerais dos algoritmos que foram utilizados para implementar as funcionalidades mencionadas, junto com a teoria matemática que está por traz desses algoritmos.

Cabe ressaltar que o foco refere-se à explanação das funcionalidades conceituais, e não das gráficas. Portanto, não será tratado, por exemplo, como é feito para tirar os desenhos gráficos de objetos e morfismos na funcionalidade ligada a permitir que estes sejam apagados. Tal decisão adveio da percepção de que mais importante do que explicar recursos gráficos de uma linguagem de programação específica é apresentar as estratégias algorítmicas utilizadas para simular em um computador os conceitos matemáticos utilizados.

Também não são apresentadas todas as definições correspondentes às estruturas matemáticas utilizadas na ferramenta, apenas aquelas que se mostrem relevantes para explicitar as características de um determinado algoritmo. Este autor sugere os livros [MEN2001] e [MEN2005] - ligados a Teoria das Categorias e Matemática Discreta, respectivamente - como fonte de consulta para os conceitos aqui tratados. As estruturas matemáticas utilizadas neste trabalho costumam estarem disponíveis na maioria dos livros de Teoria das Categorias e de Matemática Discreta.

4.1 Relações no CaTReS

Na Ciência da Computação, muitas construções são baseadas em relações ou em conceitos derivados - como funções parciais. Banco de dados relacional e rede de petri são dois exemplos de conceitos ligados à Ciência da Computação e que são definidas utilizando o conceito de relação.

A escolha realizada em [VIE2003] - a respeito de qual estrutura dar ao objetos e morfismos do CaTLeT - pode ser justificada tanto pela utilidade desse conceito na Ciência da Computação como pela capacidade de facilmente ser possível derivar outras estruturas apenas acrescentando restrições ao conceito de relação. Neste trabalho, valeu-se dessa facilidade para implementar novas estruturas no CaTReS.

Definição 4.1 - Relação
Sejam A e B conjuntos. Uma *Relação* (pequena e binária) R de A em B é um subconjunto de um produto cartesiano $A \times B$, ou seja:

$$R \subseteq A \times B$$

sendo que:

A é denominado *domínio*, *origem* ou *conjunto de partida* de R;

B é denominado *contra-domínio*, *codomínio* ou *conjunto de chegada* de R.

Para uma relação R ⊆ A×B - que também pode ser denotada por R:A→B -, se ⟨a,b⟩ ∈ R afirma-se que *a relaciona-se com b*.

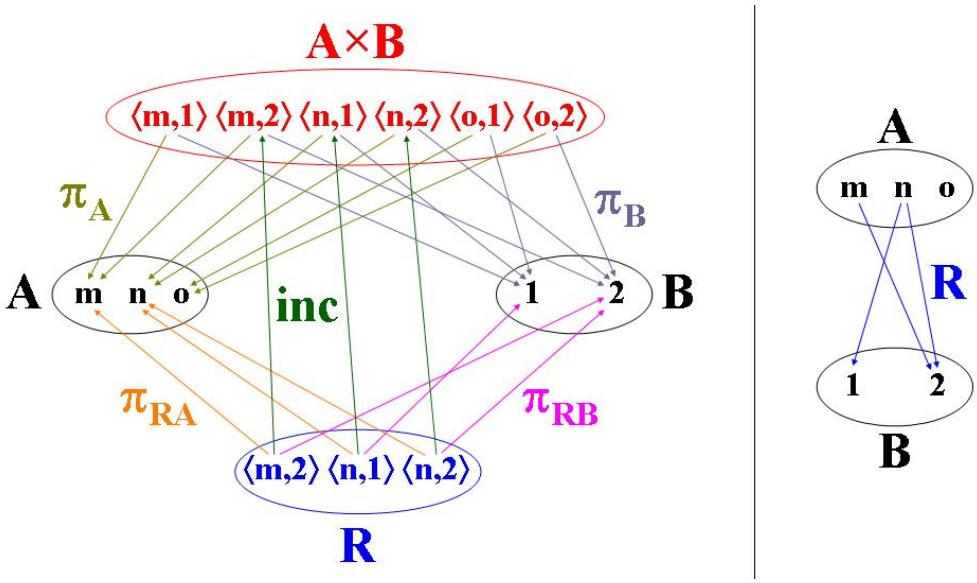


Figura 4.1: Exemplo de uma relação - R ⊆ A×B, ou R:A→B

Um outro conceito bastante importante para compreender este trabalho é o da categoria FinRel - a categoria com todos os conjuntos finitos como objetos e todas as relações binárias finitas como morfismos -, como segue:

Definição 4.2 - Categoria FinRel
A categoria *FinRel* é definida da seguinte maneira:

$$\text{FinRel} = \langle \text{Obj}_{FinRel}, \text{Morf}_{FinRel}, \partial_{0FinRel}, \partial_{1FinRel}, \iota_{FinRel}, \circ_{FinRel} \rangle$$

onde:

a) Obj_{FinRel} é o conjunto formado por todos os conjuntos finitos;

b) Morf_{FinRel} é o conjunto formado por todas as relações binárias (finitas) que tenham domínio e contradomínio em conjuntos finitos;

c) $\partial_{0FinRel}$,$\partial_{1FinRel}$:Morf_{FinRel}→Obj_{FinRel} são tais que, para qualquer relação R de Morf_{FinRel} com domínio em A e contradomínio em B (R:A→B), tem-se que $\partial_{0FinRel}$(R)=A e $\partial_{1FinRel}$=B;

d) $\circ_{FinRel}$:$(\text{Morf}_{FinRel})^2$→Morf_{FinRel} é a operação parcial de composição de relações de Morf_{FinRel}, de tal forma que, sendo R:A→B e S:B→C relações de Morf_{FinRel}, S$\circ_{FinRel}$R:A→C é tal que, para todo elemento a de A, b de B e c de C, ⟨a,b⟩ ∈ R e ⟨b,c⟩ ∈ S se e somente se ⟨a,c⟩ ∈ S$\circ_{FinRel}$R;

e) ι_{FinRel}:Obj$_{FinRel}$→Mor$_{FinRel}$ é a operação identidade de FinRel, segundo a qual cada conjunto finito A é associado à relação identidade id$_A$:A→A tal que para todo a∈A, id$_A$(a)={a}.

No CaTLeT versão estendida, toda categoria representada é uma subcategoria finita de FinRel. Isso acontece porque o usuário, ao criar um objeto, entra com uma cardinalidade que representa o número de elementos que o conjunto em questão possui. Ao criar o morfismo, o usuário entra com a relação entre os dois conjuntos criados. Para lidar com os elementos dos conjuntos na hora de criar a relação, o programa pressupõe que o nome de cada elemento é formado por uma letra - a qual é derivada do identificador do objeto - e por um indexador. Tal esquema pode ser visto na figura 2.10.

Tal forma de lidar com o problema mostra que, embora o CaTLeT represente subcategorias finitas de FinRel, nem toda subcategoria finita de FinRel é representável pelo software, posto que não é possível definir o nome dos elementos do conjunto. Entretanto, é possível dizer que toda subcategoria finita de FinRel possui uma isomorfa que é representável no CaTLeT - desconsiderando fatores ligados a limitação de memória.

Ao permitir representar a estrutura dos objetos não somente através de sua cardinalidade, mas também através de um conjunto de elementos dados pelo usuário como entrada, estamos oferecendo uma alternativa que mais colabora com a usabilidade do que, propriamente, aumenta o poder de expressão ferramenta. Agora, temos o poder de representar mais subcategorias finitas de FinRel do que tínhamos antes. Entretanto, continuam não sendo todas, e sim somente aquelas cujos conjuntos tenham elementos que são compostos por caracteres. É bom ressaltar que, quando dizemos os objetos de FinRel são todos os conjuntos finitos, estamos incluindo aqueles cujos elementos não representam uma palavra ou um texto (por exemplo, um conjunto que tenha como elementos grafos). Em termos categoriais, as novas estruturas que podem ser representadas já encontravam no CaTLeT um isomorfo.

Entretanto, cabe ressaltar que a usabilidade - que tem a ver com o critério acessibilidade - é um aspecto valorizado neste trabalho. Se antes o usuário só podia criar relações entre palavras formadas pela concatenação de letras com indexadores, agora é possível entrar, por exemplo, com nomes de cidade em um conjunto, nomes de países em outro, nomes de atividades profissionais em um terceiro e números representando idades em um quarto. Posteriormente, é possível criar relações entre esses conjuntos, construindo uma estrutura similar a de um banco de dados relacional.

Entretanto, implementar essa funcionalidade implica mais verificações de consistência do que representar conjuntos como objetos com cardinalidade implicava:

- Sabendo que um conjunto com elementos repetidos é igual ao mesmo conjunto sem os elementos repetidos (por exemplo, {1,1,2}={1,2}), não convém armazenar em memória repetição de elementos - além de, conceitualmente, ser inadequado tratar conjuntos com elementos repetidos. Assim, quando o usuário entra com elementos repetidos em um conjunto, as repetições são descartadas;

- Qualquer categoria é composta por duas coleções, uma que contém os objetos e outra que contém os morfismo. Em coleções, assim como em conjuntos, repetir elementos não o torna diferente. Assim sendo, não é conveniente, pelas mesmas razões alegadas no item anterior, permitir que conjuntos e relações já existentes tornem a serem representadas. Assim, caso o usuário tente entrar com um conjunto ou com uma relação já existente na categoria, as repetições também devem ser descartadas;

- Como o CaTReS manteve a funcionalidade de representar conjuntos através da cardinalidade, torna-se necessário um cuidado para tratar do conjunto vazio. A categoria FinRel possui somente um conjunto sem elementos. Não haveria sentido, portanto, permitir que fosse criado um conjunto de cardinalidade zero e outro sem elementos na mesma categoria. Esse é o único caso em que um conjunto oriundo da entrada de elementos e outro, oriundo da entrada de cardinalidade, são tratados como sendo iguais, sendo feito o descarte da repetição.

Embora conjuntos sejam estruturas matemáticas sem ordem (por exemplo, {1,2}={2,1}), o CaTReS os modela como listas, estabelecendo uma relação de ordem. Tal ordenamento é transparente para o usuário. O motivo de tal escolha é o de economizar memória ao armazenar relações.

Desde o CaTLeT relações são modeladas como um vetor de pares ordenados - embora, a rigor, uma relação também seja um conjunto (de pares ordenados) e, portanto, ao contrário dos vetores, não possua ordem. No CaTLeT, cada par ordenado do vetor armazena dois índices, o primeiro referindo-se ao elemento do domínio e o segundo, ao elemento do contradomínio. Nesse simulador, cada índice é armazenado como um tipo inteiro, o qual, em Java, possui um tamanho de 32 bits. Para armazenar cada par ordenado de uma relação, portanto, se gasta um total de 64 bits - sem contar o que é gasto para armazenar a estrutura de par ordenado.

Manter essa forma de armazenar morfismos limitaria os avanços possíveis no CaTReS, sobretudo o de representar todas as relações possíveis entre dois conjuntos, posto que sendo A um conjunto de cardinalidade #A e B um conjunto de cardinalidade #B, o número de relações possíveis com domínio em A e contradomínio em B é igual à cardinalidade de P(A$\times$B) - conjunto das partes do produto cartesiano A$\times$B -, que é dado por:

$$\#\mathrm{P}(\mathrm{A}\times\mathrm{B}) = 2^{\#A\times\#B}$$

Se tivermos, por exemplo, um domínio com 3 elementos e um contradomínio com 4, armazenar todas as relações existente equivale a armazenar $2^{3\times4}=2^{12}=4096$ relações. Se o nosso contradomínio tiver o acréscimo de apenas um elemento, passaremos a armazenar $2^{15}=32768$ relações (oito vezes mais). Como é possível constatar, trata-se de um problema de complexidade exponencial. Dessa forma, armazenar uma relação torna-se um problema crítico e, portanto, esta deve ser representada valendo-se da menor estrutura possível. A forma encontrada para minimizar sua estrutura foi ver a relação como uma tabela.

Tabela 4.1: Representação da relação exemplificada na figura 4.1

	2	**1**
o	0	0
n	1	1
m	1	0

A tabela 4.1 nos mostra um exemplo de como uma relação pode ser modelada através de uma tabela. A primeira coluna contém os elementos do domínio e a primeira linha contém os elementos do contradomínio. Quando um elemento x do domínio se relaciona com um elemento y do contradomínio, a célula (x,y) possui o valor 1 (um). Caso contrário, (x,y) terá o valor 0 (zero). Observe, portanto, que se trata de uma representação

binária: utilizando apenas um bit - mais a informação de posição (x,y) -, a informação de um par ordenado é armazenada.

Tabela 4.2: Representação equivalente à da tabela 4.1

⟨o,2⟩	⟨o,1⟩	⟨n,2⟩	⟨n,1⟩	⟨m,2⟩	⟨m,1⟩
0	0	1	1	1	0

Na tabela 4.2 temos uma outra forma de ver a relação da figura 4.1. Aqui, temos na primeira linha todos os elementos resultantes do produto cartesiano do domínio com o contradomínio. Na segunda linha temos, através do número 1, a indicação de quais destes elementos fazem parte da relação.

Essa representação nos mostra que uma relação pode ser vista como um número binário. Cada diferente combinação de zeros e uns nos fornece uma diferente relação. Começando com todos os bits zerados, o que nos indica uma relação vazia, e terminando com todos os bits em um, nos indicando que cada elemento do domínio se relaciona com todos os elementos do contradomínio. Dessa forma, um número de 6 bits pode armazenar até 6 pares ordenados.

Sabendo que podemos representar uma relação através de inteiros, a primeira questão relevante a se desvendar é como, para montar uma relação dada pelo usuário, acrescentar os pares ordenados nesse inteiro. Sabendo que cada par ordenado é representado por um bit, inserir um par ordenado no inteiro significa “ligar” esse bit (atribuindo-lhe 1). As pergunta que se fazem necessárias são:

- Quando o usuário entra com os pares ordenados da relação, como se descobre, para cada par ordenado, os índices do primeiro e do segundo componente do par nas listas que reproduzem os conjuntos domínio e contradomínio, respectivamente?

- Descobertos os índices, como se faz para criar um número inteiro que represente essa relação?

A primeira pergunta é de resposta mais simples: para cada par ordenado entrado pelo usuário, para encontrar o índice do primeiro componente do par é feita uma pesquisa em cada campo da lista que armazena o domínio da relação. Quando a seqüência de caracteres de um campo for igual ao entrado pelo usuário, então se pega aquele índice para armazenar a relação. O mesmo procedimento é feito com o segundo componente do par ordenado, mas na lista que armazena o contradomínio.

Para responder a segunda pergunta é necessário um pouco de conhecimento a respeito de números binários. Sabemos que o inteiro 0 representa a relação vazia, já que nenhum bit desse inteiro está “ligado”. Portanto, é a partir do número 0 que começamos a inserir pares ordenados, através de operações adequadas que “liguem” os bits apropriados. No CaTReS, a operação escolhida foi a operação lógica | (*or bit-a-bit*). O peso de cada casa decimal deve ser levado em consideração para obter o resultado apropriado da relação.

Assim sendo, sejam as seguintes estruturas (criadas no CaTReS):

a) A={a_0,a_1,a_2,...,a_{n-3},a_{n-2},a_{n-1}} um conjunto de n elementos, indexados de 0 até n-1;

b) B={b_0,b_1,b_2,...,b_{m-3},b_{m-2},b_{m-1}} um conjunto de m elementos, indexados de 0 até m-1.

Imagine que deseje-se criar no CaTReS uma relação R $\subseteq$ A$\times$B, sendo que o par $\langle a_i,b_j \rangle \in$ R. O usuário, então, clicará no objeto A (soltando o botão) e, posteriormente, clicará no objeto B. Então, aparecerá uma janela de diálogo, na qual o usuário entra textualmente com a relação requerida. Um exemplo dessa janela de diálogo do CaTReS pode ser vista na figura 4.2.

Figura 4.2: Janela de diálogo do CaTReS para a entrada de relações

Na figura, observamos que o usuário está entrando com os pares ordenados ⟨José Antônio, Porto Alegre⟩, ⟨José Antônio, São José dos Ausentes⟩ e ⟨José, São José dos Ausentes⟩, que poderia ser vista, por exemplo, como a relação entre nome de sócios de um clube e cidades onde já morou.

Voltando à relação R $\subseteq$ A$\times$B que o usuário deseja criar, após digitá-la e clicar em OK, o CaTReS converterá cada par ordenado de elementos em um par ordenado de índices, conforme foi mencionado ao responder a primeira pergunta. Assim, do par $\langle a_i,b_j \rangle$ geraremos ⟨i,j⟩. Com esses pares de índices será gerado um número inteiro que represente de maneira única a relação dada. Note-se que os inteiros de 0 até $2^{n\times m}$-1 representam todas as possíveis relações entre A e B. Qualquer relação entre A e B encontra um número inteiro que a represente neste intervalo, e não há número inteiro neste intervalo que não represente uma relação, assim como não há inteiro que represente duas relações distintas e tampouco há relação representada por dois inteiros distintos. Tal fato é facilmente perceptível fazendo variações em cima da tabela 4.2: zerando a tabela, temos a relação vazia, e a cada incremento do inteiro armazenado na tabela temos uma diferente relação, até o ponto de atingirmos a relação de todos os elementos do domínio com todos os do contradomínio.

Para gerar um inteiro a partir de uma relação, o CaTReS interpreta a tabela de aspecto matricial - como a da figura 4.1, sendo a primeira linha e coluna índices - como sendo um vetor binário - como na tabela 4.2, sendo a primeira linha indexador -, que na verdade é uma forma de ver um inteiro (um vetor de bits). Tal vetor é construído a linha de 1 à linha de índice 0 e, posteriormente, a linha de índice 2 ao resultado, e assim por diante - de maneira similar a como a tabela 4.1 é convertida na tabela 4.2 (sendo as células com elementos ou pares ordenados índices). Assim, uma matriz 5x5 é convertido em um vetor de 25 células, e o índice ⟨1,0⟩ na tabela passa a ser o índice 5 do vetor.

O índice ⟨i,j⟩ da tabela domínio×contradomínio é convertido, portanto, em um índice de vetor, que é dado por i×m+j. Aplicando essa regra para todos os pares da relação, teremos uma matriz de bits convertida em um vetor indexado de bits. Para converter esse vetor em um inteiro, entretanto, é necessário observar a cada casa binária possui o que costuma se chamar de *significância*, que é o valor com que essa casa binária acrescenta ao número quando está em 1. Assim, para acrescentarmos o índice i×m+j a um inteiro, é necessário gerar um número inteiro com a significância da posição da casa binária de mesmo índice. Esse inteiro é o $2^{i\times m+j}$, conforme nos mostra a tabela 4.3 - nela, por motivos de clareza, um produto $a\times b$ é representado simplesmente por ab.

Tabela 4.3: Peso das casas binárias de um número que armazene uma relação R:A→B

	$\mathbf{b}_{m-1}$	...	$\mathbf{b}_j$	...	$\mathbf{b}_2$	$\mathbf{b}_1$	$\mathbf{b}_0$
$\mathbf{a}_{n-1}$	2^{nm-1}	...	$2^{(n-1)m+j}$	...	$2^{(n-1)m+2}$	$2^{(n-1)m+1}$	$2^{(n-1)m}$
...	...	...	...	...	...	...	...
$\mathbf{a}_i$	$2^{(i+1)m-1}$	...	$\mathbf{2}^{im+j}$	...	2^{im+2}	2^{im+1}	2^{im}
...	...	...	...	...	...	...	...
$\mathbf{a}_2$	2^{3m-1}	...	2^{2m+j}	...	2^{2m+2}	2^{2m+1}	2^{2m}
$\mathbf{a}_1$	2^{2m-1}	...	2^{m+j}	...	2^{m+2}	2^{m+1}	2^m
$\mathbf{a}_0$	2^{m-1}	...	2^j	...	2^2	2^1	2^0

Portanto, para acrescentar ⟨i,j⟩ a um inteiro, o CaTReS realiza a operação | entre o inteiro gerado a partir de pares ordenados anteriormente inseridos na relação e o número $2^{i\times m+j}$. Por exemplo, suponha que i=2, j=1 e m=3. Assim, o segundo operando de | será $2^{2\times 3+1}=2^7$=128 em decimal - em binário, 10000000 (perceba que somente a casa decimal de índice 7 tem dígito 1). Se, por exemplo, a variável inteira que esteja armazenando outros pares ordenados da mesma relação de ⟨2,1⟩ armazene 371 (binário 101110011) imediatamente antes de inserirmos o par ⟨2,1⟩ no inteiro. Assim, ocorrerá o seguinte cálculo entre números binários:

$$\begin{array}{r} 101110011 \\ |\ \underline{010000000} \\ 111110011 \end{array}$$

Assim, o novo inteiro passa a armazenar o par ⟨2,1⟩ e, por conseqüência, o par ordenado correspondente ao relacionamento entre o elemento do domínio de índice 2 e o elemento do contradomínio de índice 1.

Quando o CaTReS precisa recuperar a informação dos pares ordenados da relação tendo o inteiro, o domínio e o contradomínio, é feito um processo de busca de dígitos binários 1 sobre o inteiro. Assim, se a relação 231 (binário 11100111) possui como domínio e contradomínio os conjuntos indexados A={a_0, a_1} e B={b_0, b_1, b_2, b_3}, respectivamente, recuperar a relação que este número representa corresponde a fazer os seguintes passos:

1 - Variável de índice da casa binária idxBin começa zerada;

2 - Armazena-se o inteiro em uma variável auxiliar Aux;

3 - Verifica se o resto da divisão de Aux por 2 (Aux % 2) resulta 1. Caso não resulte - o que indica que o bit menos significativo de Aux é 0 (não armazena informação de par ordenado) -, incrementa idxBin, realiza a divisão inteira de Aux por 2 (o que equivale a "mover" os bits de Aux para a direita, descartando o bit menos significativo e acrescentando 0 no mais significativo) e repete o passo 3;

4 - Aux % 2 resulta 1, o que significa que o bit menos significativo de Aux armazena informação de par ordenado. O índice i no domínio será dado pela divisão inteira de idxBin pelo tamanho do conjunto contradomínio. O índice j no contradomínio será dado através do resto da divisão de idxBin pelo tamanho do conjunto contradomínio;

5 - Busca os elementos correspondentes aos índices i e j no domínio e no contradomínio, respectivamente, montando o par ordenado $\langle a_i,b_j \rangle$;

6 - Acrescenta o par ordenado encontrado no conjunto que armazena a relação. Caso Aux ainda não tenha se tornado o número inteiro 0 incrementa idxBin, realiza a divisão inteira de Aux por 2 volta para o passo 3.

No nosso exemplo, na primeira passagem, no passo 3, Aux terá 231 e se verificará que 231%2 resulta 1. Então, no passo 4, se descobrirá que i=0/4=0 e j=0%4=0. No passo 5, verificaremos que os elementos de índice 0 no domínio e no contradomínio são, respectivamente, a_0 e b_0, o que nos dará o par ordenado da relação $\langle a_0,b_0 \rangle$, o qual será acrescentando no conjunto que armazena a relação (que estava vazio). Aux, que valia 231, passa a valer 231/2=115, e idxBin, que era zero, passa a valer 0+1=1. Volta-se ao passo 3 e repete-se o processo. Ao final de 8 iterações, teremos o conjunto relação sendo $\{\langle a_0,b_0 \rangle, \langle a_0,b_1 \rangle, \langle a_0,b_2 \rangle, \langle a_1,b_1 \rangle, \langle a_1,b_2 \rangle, \langle a_1,b_3 \rangle\}$.

Visto como o conceito de relações é implementada no CaTReS, torna-se fácil verificar como a funcionalidade de criar todas as relações de um conjunto para o outro foi implementada. Basta acrescentar todos os números de 0 até $2^{n \times m}$-1 à lista de relações da categoria que tenham o domínio e o contradomínio especificados. Implementar a funcionalidade de criar todas as possíveis relações entre todos os conjuntos da categoria significa aplicar a funcionalidade anteriormente explicada para todas as possibilidades de domínio e contradomínio na categoria.

Note que a relação vem sendo tratada até este momento como sendo um único número inteiro. Entretanto, isso é apenas uma simplificação conveniente adotado por este autor para explicar os algoritmos que envolvem relações. Na prática, armazenar relações em apenas um inteiro limitaria muito a ferramenta, posto que um inteiro longo do Java possui apenas 64 bits. Esse tipo suportaria armazenar no máximo qualquer uma das possíveis relações entre dois conjuntos de 8 elementos. Se trabalharmos com um domínio de 9 elementos e um contradomínio de 8 elementos, tal estrutura passa a ser insuficiente (seriam necessários 72 bits).

Em virtude disso, o CaTReS armazena relações não através de um único inteiro, mas através de uma lista de inteiros longos do Java. Apesar deste tipo de dados possuir 64 bits este autor decidiu descartar o bit mais significativo e utilizar apenas 63 para armazenar os inteiros, já que o 64^o bit refere-se à informação de sinal (se o número é um inteiro positivo ou negativo).

A tabela 4.4 nos mostra uma relação com um domínio com 12 elementos e um contradomínio com 10. Posto que temos pares ordenados posteriores à 63^a casa binária - casa de índice 62, que corresponde ao par ordenado $\langle a_6,b_2 \rangle$ -, não é possível representar essa

Tabela 4.4: Relação R:{a_0,a_1,a_2,a_3,a_4,a_5,a_6,a_7,a_8,a_9,a_{10},a_{11}}→{b_0,b_1,b_2,b_3,b_4,b_5,b_6,b_7,b_8,b_9}

	$\mathbf{b}_9$	$\mathbf{b}_8$	$\mathbf{b}_7$	$\mathbf{b}_6$	$\mathbf{b}_5$	$\mathbf{b}_4$	$\mathbf{b}_3$	$\mathbf{b}_2$	$\mathbf{b}_1$	$\mathbf{b}_0$
$\mathbf{a}_{11}$	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*
$\mathbf{a}_{10}$	*0*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*	*1*
$\mathbf{a}_9$	*1*	*1*	*1*	*1*	*1*	*1*	*0*	*1*	*0*	*0*
$\mathbf{a}_8$	*0*	*1*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*1*
$\mathbf{a}_7$	*0*	*1*	*0*	*1*	*1*	*0*	*0*	*0*	*0*	*1*
$\mathbf{a}_6$	*0*	*1*	*1*	*0*	*0*	*0*	*0*	0	1	1
$\mathbf{a}_5$	1	0	0	1	0	0	0	1	1	0
$\mathbf{a}_4$	1	0	0	1	0	0	0	0	0	1
$\mathbf{a}_3$	1	1	1	1	1	1	1	1	1	1
$\mathbf{a}_2$	0	0	0	1	1	1	1	1	1	0
$\mathbf{a}_1$	0	0	0	0	0	0	0	0	0	1
$\mathbf{a}_0$	1	1	1	1	0	0	1	0	0	0

relação em apenas um inteiro. Portanto, a partir da 64^a casa binária está sendo utilizando um novo inteiro. Ele é interpretado pelo CaTReS, para todos os efeitos apresentados neste capítulo, como sendo uma continuação do primeiro inteiro.

Com essa estrutura, foi possível aumentar significativamente o tamanho dos morfismos representáveis. Um exemplo interessante de ser dado é em relação à relação identidade. Para um conjunto de até 12199 elementos, o CaTReS consegue criar automaticamente a relação identidade.

4.1.1 Funções (Totais) e Funções Parciais no CaTReS

Como já foi dito, CaTLeT, ao ser ampliado, passou a ter como referencial teórico a categoria FinRel. Isso significa que, desprezando limites físicos de tempo e espaço (especialmente de memória), qualquer subcategoria finita de FinRel é isomorfa a pelo menos uma categoria que é passível de ser representada no CaTLeT.

O CaTReS avançou em relação ao CaTLeT e não trabalha com apenas um referencial teórico para as categorias criadas nele. Isso significa que, ao iniciar a elaboração de uma nova categoria, o usuário pode definir configurações que limitem o potencial da categoria em construção, de modo que existam subcategorias finitas de FinRel as quais não haja categoria isomorfa desenhável nesta categoria em construção.

As restrições se aplicam ao conceito de relações, e a partir daí surgem dois conceitos muitíssimo utilizados na Ciência da Computação e na Matemática: *função parcial* e *função* (total). Estas podem ser definidas como segue:

Definição 4.3 - Função Parcial
Uma *função parcial* é uma relação que possua a propriedade da funcionalidade - ou seja, se uma relação R é uma função parcial, então sempre é verdade que, se $\langle a,b\rangle \in R$ e $\langle a,c\rangle \in R$, então b=c.

Definição 4.4 - Função (total)
Uma *função total* - ou simplesmente *função* - é uma relação que seja uma função parcial e que possua a propriedade da totalidade - ou seja, se uma relação R é uma função, então sempre é verdade que R é função parcial e que para todo elemento a do domínio existe sempre um elemento b no contradomínio tal que $\langle a,b\rangle \in R$.

Após ter concluído as formatação que o tratamento de relações obteve no CaTReS foi necessário inserir apenas algumas restrições para que fosse possível trabalhar também com esses dois conceitos, de forma que adquirimos dois novos referenciais teóricos: as categorias *FinSet* e *FinPfn*, definidas como segue:

Definição 4.5 - Categoria FinPfn
A categoria *FinPfn* é definida como sendo a subcategoria larga e não-plena de FinRel tal que todos os morfismos de FinRel que são funções parciais também são morfismos de FinPfn e nenhum outro morfismo é morfismo de FinPfn.

Definição 4.6 - Categoria FinSet
A categoria *FinSet* é definida como sendo a subcategoria larga e não-plena de FinRel tal que todos os morfismos de FinRel que são funções também são morfismos de FinSet e nenhum outro morfismo é morfismo de FinSet.

No CaTReS, ao se cria uma nova categoria ("em branco") o software abre uma janela que permite a configuração de que tipo de morfismos a categoria terá. Lá, é possível definir que a categoria criada terá como morfismos relações, funções parciais ou funções. Ao definir, por exemplo, que a categoria será composta por funções, o usuário está restringindo o tipo de morfismo que ele pode criar (relações não-totais, por exemplo, não são permitidas). Assim, o referencial teórico da categoria a ser criada deixa de ser FinRel, e passa a ser FinSet - para qualquer categoria de FinSet, sempre existirá uma categoria isomorfa a ela que seja representável nessa nova categoria que começa a ser construída (desprezando fatores ligados a limitação temporal e espacial - sobretudo de memória).

Ao especificar que a categoria a ser construída possuirá apenas funções parciais como morfismos, por razão análoga à citada acima, esta categoria passa a ter como referencial teórico a categoria FinPfn.

Para evitar que o usuário entre com uma relação que não respeite as regras do tipo de morfismo escolhido pelo usuário, o CaTReS verifica a propriedade de funcionalidade - e totalidade, se for o caso - junto com outras checagens de consistência da entrada do usuário. No caso de funcionalidade, checa-se se o texto à esquerda de "`->`" (ver figura 4.2) não apareceu ainda nessa circunstância. No caso de totalidade, armazenam-se todos os elementos entrados à esquerda de cada "`->`" em uma lista, evitando repetições (já tendo verificado que são entradas válidas). Verifica-se, então, se a lista construída tem o mesmo tamanho da lista com o conjunto domínio - será entrada válida se possuir.

4.1.2 Propriedades de Relações

As relações costumam serem avaliadas essencialmente sob a luz de quatro propriedades: funcionalidade, injeção, totalidade e sobrejeção. O conceito de funcionalidade foi apresentado junto à definição 4.3 e o de totalidade, junto à definição 4.4. Abaixo seguem as definições das duas outras propriedades:

Definição 4.7 - Relação Injetora
Uma relação R:A→B é dita *injetora* se e somente se sempre for verdade que, para qualquer elemento b de B e para quaisquer elementos a_1 e a_2 de A, caso $\langle a_1,b\rangle \in R$ e $\langle a_2,b\rangle \in R$, então $a_1 = a_2$.

Definição 4.8 - Relação Sobrejetora
Uma relação R:A→B é dita *sobrejetora* se e somente se, para qualquer elemento b, existe um elemento a de A tal que $\langle a,b\rangle \in R$.

Também são bastante utilizadas a relação bijetora, a monorrelação, a epirrelação e a isorrelação. Entretanto, estas podem ser representadas pela combinação de duas ou mais propriedades já explanadas: relação injetora e sobrejetora é também bijetora; relação total e injetora é também monorrelação; relação funcional e sobrejetora é também epirrelação; relação total, injetora, funcional e sobrejetora é também isorrelação.

Na criação de uma categoria nova ("em branco"), o CaTReS permite fazer outros tipos de restrição sobre os morfismos relacionais que podem ser criados - e não apenas definindo que são funções ou funções parciais. Ao invés de especificar o tipo de morfismo, o usuário pode definir as propriedades que os morfismos criados - que serão relações - devem respeitar. Trata-se das 4 propriedades citadas neste item. O usuário pode escolher por nenhuma das propriedades - o que equivale a permitir qualquer tipo de relação - ou pode definir que uma ou mais dentre as 4 propriedades devem ser respeitadas.

Da mesma maneira que funções e funções parciais, a verificação de consistência das propriedades escolhidas é feita junto com outras checagens ligadas ao conteúdo textual entrado pelo usuário. A checagem da propriedade injetora é feita de maneira dual a verificação de funcionalidade, mencionada anteriormente - ao invés de ser à esquerda, verifica se o elemento entrado à direita de "->" não apareceu nessa circunstância. A de sobrejeção é dual à de totalidade: monta o conjunto de elementos (sem repetições) que apareceram à direita de "->" e compara seu tamanho com o do contradomínio, sendo válido se o tamanho for igual.

4.2 Funtores no CaTReS

Uma das principais aplicações da Teoria das Categorias é a de unificação de estruturas matemáticas. De fato, alguns dos mais importantes funtores são aqueles que descrevem a passagem de um tipo de estrutura matemática para outro. Nesse contexto, funtores e transformações naturais são de fundamental importância [MEN2001].

Funtor é definido como segue:

Definição 4.9 - Funtor
Sejam C=$\langle \text{Obj}_C, \text{Morf}_C, \partial_{0C}, \partial_{1C}, \iota_C, \circ_C \rangle$ e D=$\langle \text{Obj}_D, \text{Morf}_D, \partial_{0D}, \partial_{1D}, \iota_D, \circ_D \rangle$ categorias. Um *funtor covariante* ou simplemente *funtor* f:C→D é um par de funções:

$$\text{f} = \langle \text{f}_O, \text{f}_M \rangle$$

no qual f_O:$\text{Obj}_C \rightarrow \text{Obj}_D$ e f_M:$\text{Morf}_C \rightarrow \text{Morf}_D$ são tais que (a operação $\circ$ é a composição de funções):

- preserva-se origens: $\partial_{0D} \circ \text{f}_M = \text{f}_O \circ \partial_{0C}$
- preserva-se destinos: $\partial_{1D} \circ \text{f}_M = \text{f}_O \circ \partial_{1C}$
- preserva-se identidades: $\iota_D \circ \text{f}_O = \text{f}_M \circ \iota_C$
- preserva-se composições: para todos os morfismos f:A→B e g:B→C de Morf_C, tem-se que $\text{f}_M(\text{g} \circ_C \text{f}) = \text{f}_M(\text{g}) \circ_D \text{f}_M(\text{f})$

Em outras palavras, um funtor é um morfismo entre duas categorias que preserva a estrutura categorial. Uma forma conveniente de ver essa estrutura para o CaTReS é

como um morfismo entre dois objetos, sendo que os objetos representam categorias e o morfismo representam um par ordenado de funções (com determinadas propriedades).

Em termos gráficos, o CaTReS herdou do CaTLeT um ambiente visual de construção de categorias - o qual pode ser visto na figura 2.9 - o qual sofreu melhorias. A fim de aproveitar a boa usabilidade dos recursos desse ambiente, decidiu-se utilizar a mesma estrutura diagramática do CaTReS, a qual é utilizada para representar categorias, para representar também funtores. Dessa maneira, o aspecto visual de funtores interligando categorias é bastante parecido com aquele que pode ser visto na figura 2.9. Entretanto, há questões ligadas com a interação do usuário com o CaTReS que tiveram que ser adaptadas para permitir o tratamento adequado de funtores:

- Ao clicar no botão `Nova Categoria`, que no CaTLeT existia para começar a edição de categorias inicialmente vazias, no CaTReS passa a agregar a possibilidade de selecionar a criação de uma categoria de categorias. Se o usuário selecionar essa opção e clicar no botão `Ok`, será aberta uma nova janela de diálogo através da qual o usuário informará ao sistema quais dentre as categorias já abertas no CaTReS ele quer que estejam presentes nessa nova categoria como seus objetos (se a estrutura representada não for categoria, então não é disponibilizada como opção). Somente categorias cujos morfismos são relações, funções parciais ou funções são passíveis de serem objetos dessa categoria, o que exclui outras eventuais categorias abertas que trabalhem com funtores como morfismos - as quais não serão disponibilizadas como opção. Caso nenhuma categoria selecionável esteja aberta, a última janela de diálogo mencionada não aparecerá, sendo disponibilizada uma categoria inicialmente vazia;

- Quando uma categoria de categorias estiver ativa, o usuário pode realizar, em princípio, as mesmas interações que ele poderia realizar com outros tipos de categoria. Entretanto, algumas ações provocam efeitos específicos no CaTReS:

  – Criar um novo objeto nessa categoria equivale a criar uma nova categoria (de tipo diferente ao que ela representa). Assim, o diálogo de criação de uma nova categoria aparece, mas com a opção de criar uma categoria de categorias desabilitada. Assim, o usuário cria uma nova categoria "em branco" no CaTReS, a qual fica associada a um novo objeto da categoria que motivou sua criação;

  – Apagar um objeto nessa categoria equivale a fechá-lo, retirando-o do grupo de categorias abertas naquele momento. Portanto, ao fazê-lo, o evento disparado provocará resultado idêntico ao da opção `Fechar Categoria`, do menu `Aplicação`, aplicado sobre a categoria correspondente ao do objeto a ser excluído.

- Quando uma categoria desse tipo estiver ativa, as opções referente à criação de todas as possíveis relações - envolvendo dois objetos ou todos os objetos - ficarão desabilitadas, habilitando-se assim as respectivas opções similares, mas envolvendo a criação de todos os funtores.

Observe que há uma sincronia entre os objetos criados em uma categoria de categorias e as respectivas categorias que eles representam. Assim, reflete-se na estrutura interna do objeto de uma categoria de categorias qualquer tipo de modificação realizada na categoria associada a esse objeto.

Assim como as categorias elaboradas a partir de relações, funções parciais e funções possuíam cada qual o seu referencial teórico, esse tipo de categoria também possui o seu, e se trata da categoria das categorias finitas de relações finitas *FCatFRel*, que é definida como segue:

Definição 4.10 - Categoria FCatFRel
A categoria *FCatFRel* é definida da seguinte maneira:

FCatFRel = $\langle \text{Obj}_{FCatFRel}, \text{Morf}_{FCatFRel}, \partial_{0FCatFRel}, \partial_{1FCatFRel}, \iota_{FCatFRel}, \circ_{FCatFRel} \rangle$

onde:

a) $\text{Obj}_{FCatFRel}$ é o conjunto formado por todas as subcategorias finitas de FinRel;

b) $\text{Morf}_{FCatFRel}$ é o conjunto formado por todos os funtores com origem e destino em subcategorias finitas de FinRel;

c) $\partial_{0FCatFRel}$,$\partial_{1FCatFRel}$:$\text{Morf}_{FCatFRel} \rightarrow \text{Obj}_{FCatFRel}$ são tais que, para qualquer funtor f de $\text{Morf}_{FCatFRel}$ com origem em C e destino em D (f:C→D) tem-se que $\partial_{0FCatFRel}$(f)= C e $\partial_{1FCatFRel}$=D;

d) $\circ_{FCatFRel}$:$(\text{Morf}_{FCatFRel})^2 \rightarrow \text{Morf}_{FCatFRel}$ é a operação parcial de composição de funtores de $\text{Morf}_{FCatFRel}$, a qual é o subconjunto da operação parcial de composição de funtores tal que, para todo f$\circ_{FCatFRel}$g, f $\in$ $\text{Morf}_{FCatFRel}$ e g $\in$ $\text{Morf}_{FCatFRel}$;

e) ι_{FinRel}:$\text{Obj}_{FinRel} \rightarrow \text{Mor}_{FinRel}$ é a operação identidade de FCatFRel, segundo a qual cada subcategoria finita de FinRel C é associada ao funtor identidade id_C=$\langle \text{id}_{CO}$,$\text{id}_{CM} \rangle$:C→C, tal que:

   – Para todo objeto A da categoria C, tem-se que id_{CO}(A) = A;
   – Para todo morfismo R da categoria C, tem-se que id_{CM}(R) = R.

Embora uma categoria com referencial teórico FinPfn ou FinSet possa ser objeto de uma categoria de categorias no CaTReS, isso não invalida FCatFRel como referencial teórico dessas categorias de categorias. Os objetos de FCatFRel foram definidos como sendo subcategorias de FinRel pelo fato de ser possível representar em categorias de categorias do CaTReS objetos que tenham como referencial teórico tanto FinRel quanto FinPfn quanto FinSet e pelo fato de FinPfn e FinSet serem subcategorias de FinRel. Assim, mesmo que um objeto de uma categoria seja uma categoria que não tenha como referencial teórico FinRel, esta será necessariamente subcategoria finita de FinRel. Como, a rigor, todos os morfismos de uma categoria que seja objeto de outra categoria do CaTReS são necessariamente relações (já que funções e funções parciais são casos particulares de relações) então é possível dizer que o suporte a categoria de categorias do CaTReS possui como referencial teórico a categoria FCatFRel.

Seria possível implementar no CaTReS limitações em categorias das categorias, de modo a representar construções que tenham como referenciais teóricos FCatFPfn e FCatFSet (categorias similares a FCatFRel, mas que tratam funções parciais e funções, respectivamente, ao invés de relações). Entretanto, tal funcionalidade não foi prevista no CaTReS.

Dessa forma, qualquer subcategoria finita de FCatFRel é isomorfa a pelo menos uma categoria que é representável no CaTReS - desconsiderando limitações de tempo e espaço

(especialmente envolvendo memória). Assim, CaTReS adquire a condição de simulador categorial capaz de representar de maneira elegante funtores entre categorias estruturadas, característica inédita entre os simuladores categoriais conhecidos por este autor.

Projetar no CaTReS o suporte ao conceito de funtores dentro de um contexto de categoria de categorias, aproveitando os recursos e a filosofia de interação com o usuário já implementados, trouxe as seguintes vantagens:

- Maior simplicidade na interação do usuário com a ferramenta, já que ele lida com funtores de maneira parecida com que ele lida com outros tipos de morfismos;
- Menor trabalho na inserção do conceito no aplicativo, uma vez que grande parte do tratamento necessário já estava implementado;
- Maior possibilidade de aplicar funcionalidades anteriormente implementadas no contexto de funtores (como verificação de monomorfismo), mesmo que, eventualmente, seja necessário realizar adaptações.

Conceitualmente falando, funtor foi implementado inserindo alterações na classe Morfismo. Por, essencialmente, um funtor ser formado de um par de funções - sendo estas essencialmente relações -, foi aproveitado todo o tratamento de relações que já tinha sido desenvolvido e utilizado o que já tinha sido implementado a respeito de funções (relações com restrições).

Entretanto, como foi visto anteriormente, relações foram implementadas considerando que trabalham com listas indexadas que armazenem os conjuntos domínio e contradomínio. Assim, é fundamental para o conceito de relações implementado no CaTReS que os elementos do domínio e do contradomínio esteja, de alguma maneira, ordenados e indexados. Entretanto, os conjuntos de objetos e morfismos foram implementados no CaTLeT como estruturas sem ordem. A manutenção dessa estrutura inviabilizaria o reuso do que foi implementado a respeito de relações, e faria com que as vantagens de economia de espaço advinda da forma como relações são implementadas no CaTReS não fossem aproveitadas.

Seria possível aproveitar o fato de que objetos e morfismos são identificados por um índice automaticamente gerado pelo CaTReS. Contudo, para ter uma maneira única de tratar relações na ferramenta e para não realizar uma alteração que poderia implicar dificuldades em uma futura alteração na forma com que objetos e morfismos são identificados, este autor decidiu alterar a estrutura de dados que armazenava as informações referentes a conjunto de objetos e conjunto de morfismos, passando a armazená-las em vetores (indexados), tal qual é feito para armazenar elementos nos conjuntos. Algumas adaptações tiveram que ser feitas no código para suportar a nova estrutura de maneira transparente para o usuário.

Assim, funtores são implementados no CaTReS como um morfismo com duas funções - uma que mapeia objetos e outra que mapeia morfismos -, sendo realizadas as verificações de preservação da estrutura categorial no momento da entrada textual das funções pelos usuários (tal qual ocorre quando o usuário entra textualmente, por exemplo, com uma função).

Assim, temos o suporte a mais um tipo de estrutura nos objetos e nos morfismos - uma das funcionalidades previstas para o CaTReS - e temos a capacidade de representar estruturas funtoriais - outra funcionalidade prevista para o CaTReS.

Para implementar as funcionalidades ligadas a implementar todos os funtores possíveis entre categorias foram aproveitados os métodos que calculam todas as relações entre

conjuntos, mas aplicando as restrições adequadas de modo a excluir relações que não são funções e que não preservam a estrutura categorial.

A verificação do funtor identidade de uma categoria essencialmente se dá aproveitando a estrutura anteriormente implementada no CaTReS, de forma a verificar se as duas funções do funtor são identidades. De maneira análoga, a composição de funtores é feita através da composição das funções que mapeiam objetos e que mapeiam morfismos. Verificações de propriedades categoriais, em grande parte, são herdadas do que já existe, assim como o cálculo de dualidade.

Foi projetado no CaTReS a verificação de propriedades típicas de funtor. É possível encontrar os funtores fidedignos (ou fiéis) e os funtores plenos (ou cheios) de uma categoria de categorias. Tal verificação foi facilitada pelo fato do CaTReS trabalhar com morfismos agrupados em coleções de morfismos paralelos (morfismos com mesma origem e destino fazem parte da mesma coleção).

Assim, para verificar se um funtor é fidedigno o CaTReS pega cada coleção de morfismos paralelos da categoria origem e verifica se cada um deles está indo para um diferente morfismo na categoria destino. Para verificar se um funtor f=$\langle f_O, f_M \rangle$ é pleno o CaTReS avalia se, para cada possível par de objetos A e B da origem e para cada morfismo h:f_O(A)→$f_O(B)$ do destino se existe um morfismo g na categoria origem que tenha como imagem (a luz do funtor) o objeto h da categoria destino (f_M(g) = h).

Outra funcionalidade implementada é o da verificação das transformações naturais existentes a partir de uma categoria de categorias. Assim, o CaTReS avalia cada par de funtores paralelos F e G e retorna para o usuário os morfismos da categoria destino que compõe a transformação natural τ:F→G (se esta existir) - cada morfismo retornado é indexado com o respectivo objeto da categoria origem. Para encontrar a transformação natural correspondente a cada par de funtores paralelos F e G, para cada par de objetos A e B e para cada morfismo f:A→B da categoria origem é feito o seguinte procedimento:

1 - Aplica-se F e G a A, B e f:A→B, encontrando-se, assim, F(A), F(B), F(f):F(A)→F(B), G(A), G(B) e G(f):G(A)→G(B) na categoria destino dos funtores;

2 - Procura-se por morfismos τ_A:F(A)→G(A) e τ_B:F(B)→G(B) na categoria destino dos funtores tais que G(f) ∘ τ_A = τ_B ∘ F(f);

3 - Se não encontra esses morfismos, então não há transformação natural de F para G;

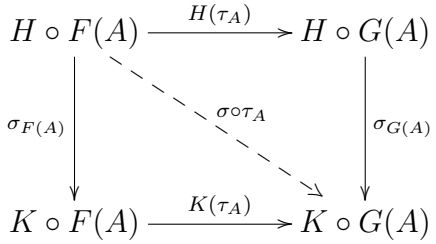
4 - Se encontra o morfismo, retorna-os, indexando τ_A como A e τ_B como B.

Se foram encontrados morfismos para todos os pares de objetos A e B e para todos os morfismos f:A→B da categoria origem, então é retornado para o usuário a coleção de todos esses morfismos, cada qual com o seu respectivo índice (objeto da origem que o identifica). Essa será a transformação natural τ:F→G. Esse procedimento é repetido para todos os pares de funtores paralelos existentes na categoria de categorias.

Foi escolhido verificar as transformações naturais dessa maneira para, assim, também já informar ao usuário as transformações naturais que representem composição vertical de duas outras. Tal informação é facilmente implementada, pois na categoria destino dos funtores existe a informação de quais morfismos são composições de quais.

O conceito de composição horizontal não está disponível em [MEN2001] e, portanto, será exposto:

Definição 4.11 - Composição Horizontal de Transformações Naturais
Sejam C, D e E categorias, F,G: C→D e H,K: D→E funtores e τ: F→G e σ: H→K transformações naturais, sendo τ_A elemento de τ indexado por um objeto A de C e $\sigma_{F(A)}$ e $\sigma_{G(A)}$ elementos de σ indexados por objetos F(A) e G(A) de D, respectivamente. A *composição horizontal de transformações naturais* $\sigma\circ\tau$: H○F → K○G é a coleção de morfismos de E indexados por objetos de C que formam uma transformação natural, de modo que, para todo morfismo $\sigma\circ\tau_A$ em $\sigma\circ\tau$ indexado por um objeto A da categoria C temos $\sigma\circ\tau_A = \sigma_{G(A)} \circ H(\tau_A) = K(\tau_A) \circ \sigma_{F(A)}$.



O diagrama acima nos mostra como é feita a composição horizontal de transformações naturais, sendo que todos os objetos e morfismos desse diagrama estão na categoria E, citada na definição. Maiores detalhes a respeito de composição horizontal podem ser encontrados em [ASP91].

No CaTReS, o usuário solicita que todas as composições horizontais das transformações naturais da categoria de categorias sejam informadas. Para realizar tal verificação, o CaTReS seleciona todas as combinações possíveis de 3 objetos (que são categorias) A, B e C e de morfismos (que são funtores) f,g:A→B e h,k:B→C. A partir deles, são calculadas as transformações naturais τ: f→g e σ: h→k, procedimento feito conforme explicado anteriormente. Para cada elemento τ_X de τ indexado por um objeto X de A, aplica-se o funtor h, encontrando os morfismos h(τ_X) de C. Verifica-se em h(τ_X) é componível com o elemento da transformação natural σ indexado por f(X). Se sim, compõe e inclui no conjunto resultado. Feito isso para todo τ_X de τ, encontrou-se uma das composições horizontais de transformações naturais. Muda-se a combinação A, B, C, f,g:A→B e h,k:B→C e começa-se novamente. Ao final, o CaTReS informa todas as composições horizontais existentes na categoria de categorias.

Em uma categoria de categorias, o CaTReS é capaz de verificar a existência de adjunções, retornando ao usuário todas as adjunções existentes. Para fazê-lo, ele varre todos os pares de funtores f:A→B e g:B→A. Para cada par selecionado, realiza-se o seguinte procedimento:

1 - Verifica a existência de uma transformação natural η do funtor identidade de A para o funtor g○f. Caso não haja, então não há adjunção envolvendo f e g;

2 - Se há, então pega todos os pares de objetos X de A e Y de B com todos os morfismos k:X→g(Y) de A. A seqüência de passos que segue vale para cada par de objetos X e Y e para cada morfismo k:X→g(Y);

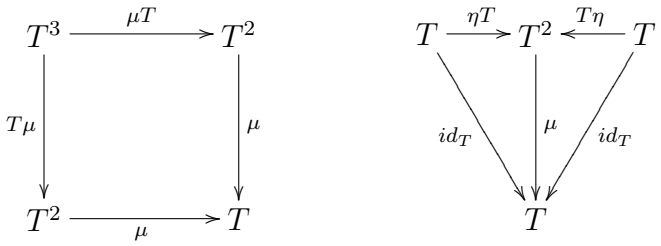
3 - Pega-se todos os possíveis morfismos l:f(X)→B e, para cada morfismo desses, verifica se g(l)○η_X = k (sendo η_X elemento de η indexado pelo objeto X). Se isso for verdade, então $\langle$f,g,$\eta\rangle$ é adjunção

No final, todas as adjunções são informadas para o usuário.

O programa também tem implementada a capacidade de mostrar as mônadas presentes na categorias de categorias. Como o conceito de mônada não está disponível em [MEN2001], ele é explicitado abaixo:

Definição 4.12 - Mônada

Uma *mônada* sobre uma categoria C é uma tripla $\langle$T, μ, $\eta\rangle$, onde T: C$\rightarrow$C é um funtor, μ:T$^2\rightarrow$T e η:id$_C\rightarrow$T são transformações naturais e os seguintes diagramas comutam:



Mais detalhes a respeito de mônadas podem ser vistos em [MOG91].

Para realizar tal procedimento, o CaTReS verifica, para cada objeto A (categoria) e cada endofuntor T:A$\rightarrow$A se existem transformações naturais μ:T$^2\rightarrow$T e η:id$_A\rightarrow$T. Se existirem, o programa testa se a tripla $\langle$T,μ,$\eta\rangle$ satisfaz as propriedades de comutatividade descritas nos diagramas acima. Caso satisfaça, acrescenta a tripla à lista de mônadas existentes na categoria, retornando ao final todas as mônadas existentes para o usuário.

5 PROJETO DO SOFTWARE

Neste capítulo é apresentado o projeto de software que permitiu, a partir do CaTLeT versão estendida (que incluía o tratamento de relações sobre objetos com cardinalidade representando conjuntos), a construção do CaTReS.

5.1 Paradigma Orientado a Objetos

O conceito de programação orientada a objetos não é novo. No final da década de 60, a linguagem Simula67, desenvolvida na Noruega, introduzia conceitos hoje encontrados nas linguagens orientadas a objetos. Em meados de 1970, o Centro de Pesquisas da Xerox desenvolveu a linguagem Smalltalk, a primeira totalmente orientada a objetos. No início da década de 80, a AT&T lançaria a linguagem de programação C++, uma evolução da linguagem C em direção à orientação a objetos.

Atualmente, a grande maioria das linguagens incorpora características de orientação a objetos, como Java e Object Pascal. Além das linguagens de programação, é possível encontrar esse conceito em banco de dados, como o Oracle8 e, principalmente, Jasmine da CA.

A programação orientada a objetos tem como principais objetivos reduzir a complexidade no desenvolvimento de software e aumentar sua produtividade. A análise, projeto e programação orientada a objetos são as respostas para o aumento da complexidade dos ambientes computacionais que se caracterizam por sistemas heterogêneos, distribuídos em redes, em camadas e baseados em interfaces gráficas.

As barreiras existentes para o paradigma orientado a objetos foram vencidas ao longo da década de 90 e hoje é uma realidade nas empresas de pequeno, médio e grande porte cuja atividade envolva desenvolvimento de software. Características como encapsulamento de operações, herança de classes (que tem a ver com reuso de classes implementadas) e polimorfismo justificam a capacidade com a qual esse paradigma penetrou no mundo corporativo da informática.

A *análise e o projeto orientado a objetos* encontra hoje em inúmeras ferramentas computacionais um suporte para trabalho. A notação UML (*Unified Modeling Language* vem emergindo como um padrão para modelagem de projetos orientados a objetos, muito pelo fato de os projetos nela descritos serem capazes de representar as vantagens do paradigma. Quase todas as ferramentas CASE (Compoter Aided Software Engineering) que permitem a realização de análise e projeto orientados a objetos suportam a notação UML e suas extensões.

Neste trabalho, a análise e projeto orientado a objetos é uma extensão da análise e projeto já realizados em [PFE2002], posto que a base adotada pelo CaTReS é o CaTLeT, um software já desenvolvido e acabado. Assim, tal qual o autor do CaTLeT, este autor tam-

bém baseou nos diagramas de pacotes e nos diagramas de classe, oriundos da linguagem UML, na análise e no projeto das extensões realizadas no CaTReS.

Para melhor compreender o trabalho desenvolvido ao longo deste capítulo, este autor sugere o livro [LAR2000] como referência para a notação UML.

5.2 Qualidade de Software no CaTReS

O CaTLeT, segundo [PFE2002], almejou alcançar características de qualidade pautadas por critérios definidos por McCall, os quais envolvem integridade, manutenibilidade, flexibilidade, testabilidade, interoperabilidade, usabilidade, eficiência, correteza, confiabilidade, portabilidade e reusabilidade - sendo nestes últimos 6 critérios aqueles nos quais o CaTLeT mais buscou se destacar. Estes são critérios gerais de qualidade para qualquer tipo de software, os quais não tratam, por exemplo, de critérios específicos referentes ao propósito de uma aplicação.

O CaTReS, como sucessor do CaTLeT, buscou manter o padrão de qualidade idealizado por Pfeiff - mesmo porque, mostrou-se uma conseqüência natural, pela forma como o CaTLeT tinha sido projetado e implementado. Contudo, este autor estabeleceu critérios de qualidade ligados também ao propósito do software. Tais critérios são decorrência da percepção deste autor a respeito de quais qualidades são desejáveis em um simulador categorial, e já foram mencionados no item 1.2 desta dissertação. Não são critérios contraditórios em relação aos apresentados em [PFE2002], e é possível observar uma intersecção no que diz respeito ao aspecto acessibilidade - que, como já foi visto, encontra suporte em usabilidade e portabilidade.

Observando os critérios de qualidade expostos em 1.2, podemos realizar uma análise quanto à qualidade do CaTReS, como é feito a seguir.

a) Acessibilidade: O CaTReS manteve as virtudes em termos de acessibilidade do CaTLeT, inclusive nas novas funcionalidades implementadas. O conceito de funtor, por exemplo, foi projetado dentro do contexto de um morfismo de uma categoria de categorias. Preserva-se, assim, uma coerência de método de interação que permite ao usuário do CaTReS lidar com morfismos em geral e com funtores de maneira bastante semelhante, questão importante no quesito acessibilidade;

b) Relevância das Estruturas Implementadas: É objetivo primeiro do CaTReS ser um software útil tanto para o aprendizado de conceitos intermediários de Teoria das Categoria quanto para a pesquisa dentro da área. Para isso, foi projetado um software que atendesse a necessidade de lidar com morfismos e objetos estruturados quanto que suportasse conceitos e cálculos funtoriais. Para os propósitos deste trabalho, tais estruturas são bastante relevantes. Outras que também seriam relevantes para cooperar com os propósitos estabelecidos não foram desenvolvidas especialmente em virtude da limitação de tempo que impõe o mestrado. Além disso, quando estamos falando de ferramenta de apoio à pesquisa, é impensável projetar uma ferramenta com escopo fechado, fora do qual estejam somente estruturas de baixa relevância para a pesquisa;

c) Cobertura: Enquanto o aspecto relevância é ligado a itens que convêm serem acrescentados a um simulador, o item cobertura tem a ver com a profundidade que se dá suporte a esses itens. Ao projetar relações (item relevante) no CaTReS, julgou-se

adequado incluir tratamentos para propriedades relacionais - totalidade, funcionalidade, injeção e sobrejeção (conceitos cobertos) -, além de incluir estruturas para funções e funções parciais, que tanto podem ser vistos como itens relevantes como também como conceitos cobertos no item relevante mais geral, que é relações. O mesmo ocorre com funtores (item relevante) a partir do qual foram desenvolvidos tratamentos ligados a transformações naturais, adjunções e mônadas, que tanto podem ser visto como itens individualmente relevantes para o trabalho quanto podem ser vistos como conceitos cobertos dentro do tema categoria.

Como pode ser observado no item que trata de cobertura, nem sempre é claro distinguir o que é um item relevante ou o que é um conceito coberto dentro de um item relevante. Poderíamos dizer que projetar transformações naturais foi um item relevante, enquanto implementar composições vertical e horizontal de transformações naturais são conceitos cobertos dentro do item. Tais idéias dizem respeito ao que são os conceitos principais representados nas ferramentas e o que são conceitos acessórios (ou secundários) que reforçam o suporte que a ferramenta oferece ao conceito principal.

5.3 Projeto UML

A estrutura do diagrama de pacotes foi herdada do CaTLeT, não havendo necessidade de alterá-la para inserir as funcionalidades projetadas para o CaTReS. Este diagrama é composto de cinco pacotes:

- br.ufrgs.inf.catres.categoria - contém as classes que definem uma categoria;
- br.ufrgs.inf.catres.catresui - contém as classes que definem as interfaces com o usuário;
- br.ufrgs.inf.catres.erro - contém classes utilizadas no apoio à verificação se um diagrama constitui categoria. Foram desenvolvidas já com o intuito de incorporar à aplicação o suporte à correção automática dos erros apontados;
- br.ufrgs.inf.catres.operacao - neste pacote estão agrupadas classes de apoio aos algoritmos de cálculo;
- br.ufrgs.inf.catres.propriedades - composto por apenas uma classe, Propriedades, destinada a prover à aplicação uso facilitado das configurações de propriedades.

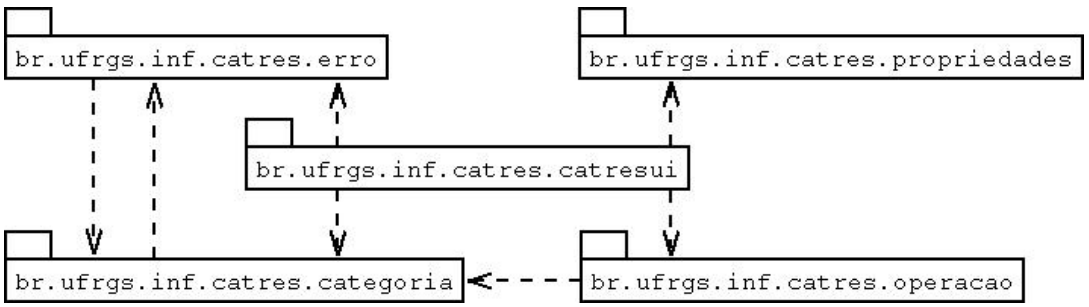


Figura 5.1: Diagrama de pacotes do CaTReS

Cada um desses pacotes corresponde a um diagrama de classes do projeto UML do simulador categorial. Abaixo, segue uma avaliação mais detalhada a respeito dos pacotes, avaliando o que foi alterado do CaTLeT para o CaTReS.

5.3.1 Pacote br.ufrgs.inf.catres.propriedades

Trata-se de um pacote de uma única classe - a classe abstrata Propriedades. Isso se deve ao fato de que essa classe não possuir semelhança semântica com nenhuma dentre as demais classes do CaTReS.

Esta classe foi criada no CaTLeT a fim de dar suporte ao projeto de internacionalização da aplicação. Por ser abstrata, não pode dar origem a instâncias, e sus dois únicos métodos são declarados métodos de classe. Sua função é prover à aplicação uma forma simples de acessar os arquivos textos que armazenem o conteúdo textual da ferramenta, de acordo com a configuração ligada à língua com a qual a ferramenta trabalha em uma dada execução. Tal configuração é armazenada em um arquivo denominado "CaTLeT_execution_properties.clt" e, em caso de este não poder ser lido, será pega a configuração de um outro arquivo, denominado "CaTLeT_default_properties.clt". Nestes arquivos está configurada uma língua e um país e, a partir dessas configurações, o programa captura em outros arquivos as mensagens textuais que ele utilizará para interagir com o usuário. Por exemplo, se em "CaTLeT_execution_properties.clt" estiver configurado a língua `pt` e o país `BR`, então teremos na área de menu do CaTReS o menu `Aplicação`. Entretanto, se a língua for `en` e o país for `US`, então o nome do menu anteriormente citado passará a ser `Application`. Tais nomes para o mesmo menu correspondem ao conjunto de arquivos que vem com o CaTReS e podem ser alterados.

O CaTReS deu continuidade ao projeto de internacionalização, visto que ele corrobora o princípio da acessibilidade. Não observou-se necessidade de realizar alteração na classe deste pacote, fazendo com que este autor se limitasse a utilizar os métodos dessa classe e adaptasse e criasse os arquivos textuais adequados para permitir o diálogo do usuário com a máquina.

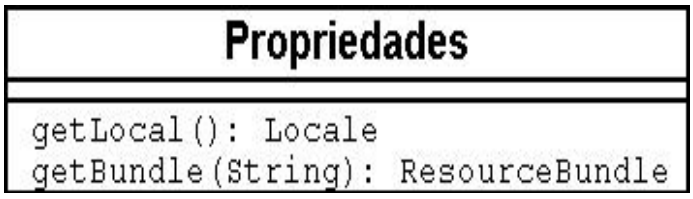


Figura 5.2: Diagrama de Classes do Pacote br.ufrgs.inf.catres.propriedades

5.3.2 Pacote br.ufrgs.inf.catres.erro

Caso o grafo representado no CaTReS seja não categorial, é neste pacote que será armazenada tal informação, armazenando qual propriedade categorial o grafo não satisfaz e quais componentes categoriais estão ferindo tal propriedade.

A forma com que o CaTReS interage com o usuário permite essencialmente a existência de duas razões para que um grafo em construção deixe de ser uma categoria: não haver um endomorfismo identidade definido para um dos objetos e não haver um morfismo composição definido para um par de morfismos componíveis.

No CaTLeT, uma terceira possibilidade era considerada: a de haver problemas de associatividade entre as composições definidas. Entretanto, no CaTReS, somente dois tipos

essenciais de morfismos são representáveis: as relações binárias e os funtores (cada uma em uma categoria diferente). Não é possível haver relações R, S e T tal que (T∘S)∘R≠ T∘(S∘R), posto que a composição de relações é uma operação associativa. Também é possível verificar que, mesmo que restrinjamos uma categoria de relações, exigindo que todas as relações nela representadas obedeçam a uma ou mais propriedades dentre funcionalidade, injeção, totalidade e sobrejeção - como é possível fazer no CaTReS - não haverá problema, pois é possível provar que uma composição S∘R:A→C de quaisquer duas relações R:A→B e S:B→C que satisfaçam um mesmo subconjunto dessas quatro propriedades, satisfará as mesmas propriedades satisfeitas por R e por S. De qualquer maneira, tal questão não tem a ver com associatividade. O fato de composição de relações ser associativa é suficiente para dizer que uma categoria com relações como morfismos não pode gerar composições não-associativas.

O mesmo acontece em categorias com funtores. Como sabemos, um funtor é um par ordenado de funções $\langle f_O,f_M\rangle$ e uma composição de funtores $\langle f_O,f_M\rangle \circ_{fnt} \langle g_O,f_M\rangle$ resulta em $\langle f_O \circ g_O,\ f_M \circ g_M\rangle$ (sendo ∘ a operação de composição de funções). Como a operação de composição de funções é associativa, então não é possível existirem funtores $\langle f_O,f_M\rangle$:C→D, $\langle g_O,g_M\rangle$:D→E e $\langle h_O,h_M\rangle$:E→F tal que $(\langle h_O,h_M\rangle \circ_{fnt} \langle g_O,g_M\rangle) \circ_{fnt} \langle f_O,f_M\rangle \neq \langle h_O,h_M\rangle \circ_{fnt} (\langle g_O,g_M\rangle \circ_{fnt} \langle f_O,f_M\rangle)$, pois:

$$\langle (h_O \circ g_O) \circ f_O,\ (h_M \circ g_M) \circ h_M\rangle = \langle h_O \circ (g_O \circ f_O),\ h_M \circ (g_M \circ h_M)\rangle$$

Dentro dessas circunstâncias, não há a possibilidade de um usuário criar no CaTReS composições não-associativas. Portanto, não há necessidade de existir uma classe que armazene tal tipo de erro.

Cabe ressaltar que não é função das classes deste pacote avaliar se um determinado grafo corresponde a uma categoria. Ele apenas guarda tal informação.

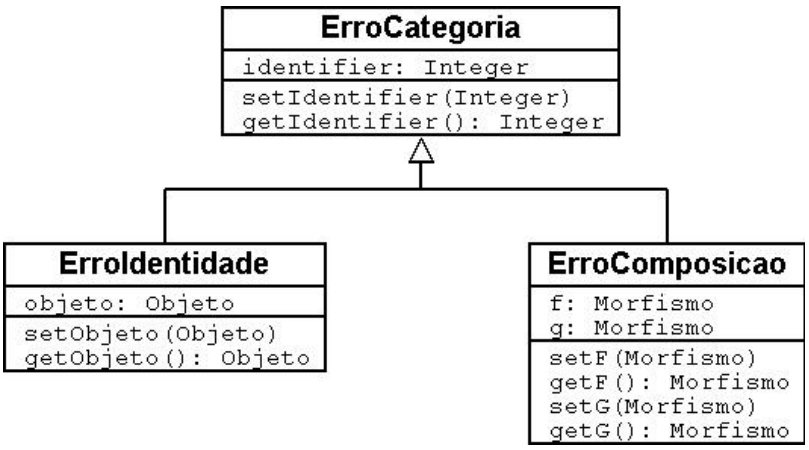


Figura 5.3: Diagrama de Classes do Pacote br.ufrgs.inf.catres.erro

5.3.3 Pacote br.ufrgs.inf.catres.operacao

Trata-se de um pacote cujas classes são estruturas auxiliares utilizadas para cálculos categoriais do CaTReS.

Pacote herdado do CaTLeT, não verificou-se necessidade de alteração neste pacote, posto que nenhuma das novas funcionalidades implementadas podem ser vistas como cálculos categoriais. Embora as estruturas armazenadas nos objetos e nos morfismos

tenham sofrido alterações, isso não implicou necessidade de alterar tais pacotes, posto que as mudanças estão encapsuladas nas classes adequadas, as quais são utilizadas como atributos nas classes destes pacotes.

A classe pré-produto foi criada para servir de repositório para armazenar resultados intermediários decorrentes da execução do primeiro passo do algoritmo para cálculo do produto categorial. A classe produto, entretanto, constituiu tanto uma estrutura auxiliar ao cálculo do produto categorial quanto o repositório de armazenamento do resultado final da execução deste cálculo.

Utiliza-se a classe Cone nos cálculos de Cone e Limite. A resposta fornecida pela execução dos métodos responsáveis pelos dois cálculos é fornecida como uma instância desta classe. O cálculo do equalizador encontra na classe Equalizador uma estrutura auxiliar.

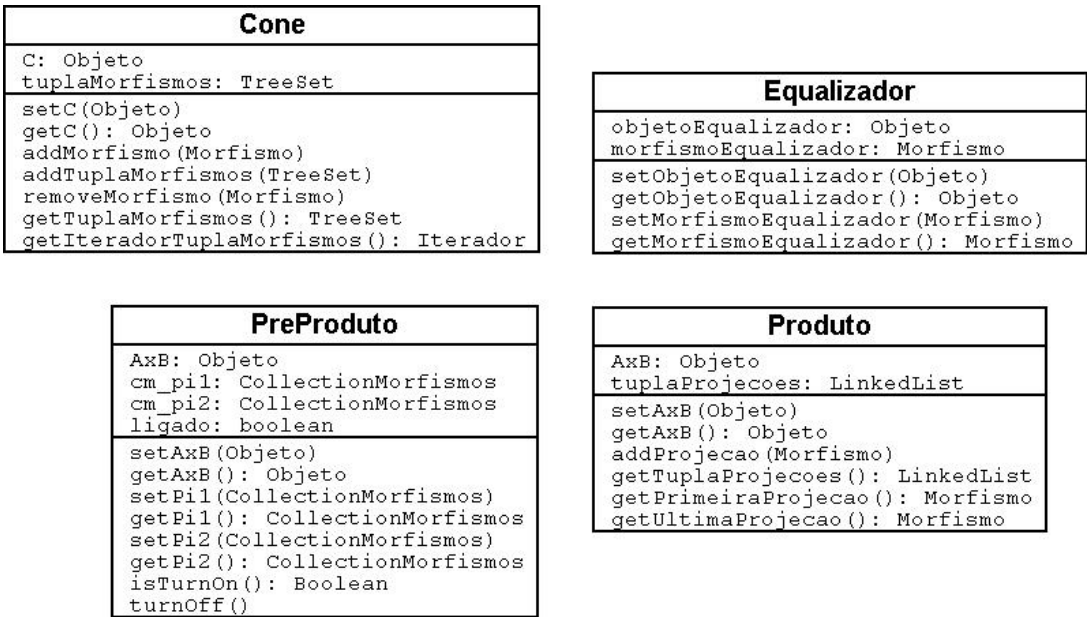


Figura 5.4: Diagrama de Classes do Pacote br.ufrgs.inf.catres.operacao

5.3.4 Pacote br.ufrgs.inf.catres.categoria

Este pacote de classes constitui o núcleo da aplicação, na qual estão as classes que definem as principais estruturas categoriais. Este e o pacote br.ufrgs.inf.catres.catresui foram os que mais sofreram alterações no processo de elaboração do CaTReS.

A classe Categoria, como uma classe elaborada para modelar, com auxílio de outras classes, uma categoria - e permitir instanciá-las -, encontra em dois atributos sua principal base conceitual: uma lista de objetos e uma lista de morfismos. No CaTLeT, tais estruturas eram modeladas como conjuntos, o que, em termos conceituais, é mais adequado. Entretanto, a implementação de funtores no CaTReS requereu o mapeamento de objetos em objetos e de morfismos em morfismos através de duas funções. Como vimos no capítulo anterior, o CaTReS trabalha com relações - e função é um tipo de relação - através de um inteiro, o qual armazena pares ordenados de índices, os quais se referem ao índice do elemento do domínio e ao índice do elemento do contradomínio. Nesse contexto, mostrou-se necessário ter objetos e morfismos indexados na estrutura categorial, e armazená-lo em conjuntos passou a ser inconveniente. Procedeu-se, assim, a migração da estrutura anterior para listas indexadas.

As listas de objetos foram projetadas para armazenar instâncias da classe Objeto, enquanto as listas de morfismos armazenam instâncias da classe CollectionMorfismos. Por sua vez, CollectionMorfismos é uma classe que armazena uma lista de morfismos paralelos - portanto, com mesma origem e destino. Tal escolha foi feita para simplificar alguns algoritmos.

Assim surgiu a CollectionMorfismos, uma classe originalmente projetada para ser uma subclasse de java.util.TreeSet - uma classe Java que encapsula conjuntos e operações sobre conjuntos. Entretanto, armazenar morfismos de uma categoria em um conjunto, como já vimos, não é o mais adequado, já que a forma como relações são implementadas no CaTReS requerem que os objetos possuam uma ordem. Assim, foi necessário remodelar CollectionMorfismos, e torná-lo uma subclasse de java.util.ArrayList - uma das classe que armazena um lista. Assim, CollectionMorfismos armazena informações referentes a origem, destino e todos os morfismos com a origem e o destino definidos - cada morfismo sendo uma instância da classe Morfismo.

A classe Categoria precisou receber novos métodos para suportar as novas funcionalidades do CaTReS. Por exemplo, o cálculo de todas as possíveis relações entre dois conjuntos dados pelo usuário é feito por um método da classe Categoria, o qual não existia no CaTLeT. Também não existia anteriormente o método responsável por verificar se um determinado conjunto que se deseja acrescentar como novo objeto de uma determinada categoria já existe. Verificações como estas tiveram que ser acrescentadas nessa classe, além da alteração do código de outros métodos já existentes.

Significativas alterações tiveram que ser feitas na classe Objeto, que passou a ser capaz de instanciar conjuntos com elementos por extensão (e não apenas através de cardinalidade, como era no CaTLeT) e também passou a ser capaz de armazenar categorias. Dessa maneira, a classe Categoria e a classe Objeto passam a ter uma relação n para n - e não mais 1 para n, como era no CaTLeT. Quando o objeto se trata de um conjunto, este é modelado dentro da estrutura da própria classe. Uma classe Objeto instanciada obtém a informação a respeito do que ela representa através de um atributo da classe Categoria, a qual informa se este é um objeto de uma categoria de relações - ou alguma derivada - ou de uma categoria de categorias.

A capacidade de o CaTReS representar diferentes tipos de morfismos estruturados foi o motivo de que fossem realizadas muitas mudanças na classe Morfismo. Através de um atributo da classe Categoria que o CaTReS sabe se um determinado morfismo é uma relação ou um funtor. Pelo fato da informação referente à relação ser essencialmente um atributo do tipo lista de inteiro e a informação referente a funtor serem dois atributos do tipo lista de inteiros (similar a duas relações), este autor não achou conveniente criar classes para encapsular o tratamento de relações e de funtores, concentrando todo o tratamento no contexto da classe Morfismo. Embora separar relações e funtores em classes seja aparentemente uma solução mais elegante, a prática mostra que muitos métodos ligados a relações também são utilizados para tratar funtores, o que torna a eficácia de tal separação questionável. Por exemplo, há dois atributos lista de inteiros na classe Morfismo a fim de armazenar funtores. Quando estamos trabalhando com relações, um deles é aproveitado e o outro é simplesmente ignorado.

Desde o CaTLeT a composição é representada através da classe ParMorfismos, a qual, quando instanciada, guarda a informação de quais morfismos da categoria estão sendo compostos nela. Contudo, o tratamento de composição - seja composição de relações, seja de funtores - é feito por um método da classe Morfismo.

Este autor julgou adequado acrescentar, por tratarem-se de estruturas novas sendo

representadas no CaTReS e pela complexidade que elas carregam consigo, três novas classes neste pacote: as classe TransfNatural, Monada e Adjuncao, modelando e encapsulando as estruturas ligadas a transformação natural, mônada e adjunção, respectivamente.

Como uma transformação natural é, dentro do contexto de uma categoria de categorias, uma coleção de morfismos de um objeto (categoria) determinada por dois funtores, a classe TransfNatural relaciona-se apenas com a classe morfismos. Como este aplicativo suporta tanto composição vertical quanto horizontal de transformações naturais, foi criado mais uma classe, a ParTransfNaturais, nos mesmos moldes de ParMorfismos, que armazena a estrutura responsável pela composição de transformações naturais, sejam elas horizontais ou verticais.

A classe Monada se relaciona com TransfNatural e com Morfismo, e armazena as estruturas relevantes de uma mônada. A classe Adjuncao, assim como Monada, também é determinada por funtores e transformações naturais e, portanto, também se relaciona com TransfNatural e com Morfismo. A figura 5.5 mostra o diagrama de classes deste pacote.

5.3.5 Pacote br.ufrgs.inf.catres.catresui

Trata-se do pacote onde estão as telas desenvolvidas para a interação com o usuário. As principais classes deste pacote são a DiagramBuilder - o container da aplicação - e DiagramBuilderPanel - a área destinada à edição de grafos. É a partir destas que toda a ação do CaTReS é coordenada.

Uma série de novas telas de diálogo tiveram que ser inseridas, em virtude de novos diálogos oriundos das novas funcionalidades implementadas no CaTReS. Tiveram que ser implementadas desde telas nas quais o usuário entra com os objetos origem e destino para que sejam criadas todas as relações entre esses objetos até diálogos para entrada das categorias que compõe uma categoria de categorias.

Algumas telas tiveram que sofrer adaptações. A tela utilizada na criação de categorias, por exemplo, passou a requerer do usuário informações referentes ao tipo de categoria criada (se os morfismos desta categoria são relações, funções parciais ou funções, se as relações são injeções, funcionais, sobrejeções ou totais, ou se os morfismos são funtores e os objetos são categorias).

Algumas das principais classes do pacote aparecem em 5.6.

5.4 Considerações sobre o Projeto

Os diagramas UML desenvolvidos neste capítulo guardam uma semelhança com os apresentados em [PFE2002]. Tal situação é absolutamente normal, posto que este projeto é uma continuação daquele desenvolvido na dissertação de Pfeiff.

Se observarmos o projeto na íntegra, teremos uma percepção que houve um acréscimo mais significativo em termos de modelagem de conceitos suportados pelo CaTReS do que propriamente dos recursos gráficos utilizados.

Tal situação deve-se principalmente a uma opção explícita de projeto. Acreditou-se desde o princípio que não haveria necessidade de ampliar significativamente os recursos gráficos disponíveis, posto que o CaTLeT já possui os recursos necessários para a implementação das funcionalidades desejadas.

Embora conceitos funtoriais sejam de extrema importância para atender aos objetivos - servindo aos propósitos de pesquisa descritos nos objetivos -, é relevante observar que a forma como foi implementado relações no CaTReS interfere em todos os conceitos

aqui trabalhados - sobretudo na definição de funtores, que permitiu utilizar os recursos relacionais implementados para derivar o conceito de funtores de maneira simples.

Como a forma como foi projetada e implementada relações interfere em muitas das classes do pacote Categorias - o núcleo do projeto -, otimizar essa maneira de tratar relações implica eficiência. Ao final deste projeto, observou-se que seria possível acrescentar melhorias que contribuiriam significativamente para a eficiência e até para a ampliação do poder de representação prática do software.

Um exemplo é a criação de todas as relações entre dois conjuntos. Da forma como está implementado no CaTReS, estão sendo armazenados uma lista de inteiros para cada relação. Como, na prática, essa lista de inteiros pode ser vista como um único inteiro - que, eventualmente, utiliza mais bits do que qualquer tipo inteiro representável em Java -, cada lista armazenado os inteiros de 0 até 2^{mn}-1, sendo m e n o tamanho dos conjuntos origem e destino, respectivamente. Uma maneira simples de economizar espaço de armazenamento seria não armazenar os inteiros que estão em seqüência, armazenando apenas o primeiro e o último inteiro da seqüência, com alguma representação de que, entre eles, há uma seqüência de inteiros.

Assim, ao calcularmos todas as relações de um conjunto de m elementos para outro de n elementos, ao invés de armazenarmos 2^{mn} listas de inteiros, armazenaríamos apenas duas listas de inteiro - uma guardando o inteiro 0 e a outra guardando o inteiro 2^{mn}.

Tal idéia não chegou a ser implementada, e está na seção de trabalhos futuros como sugestão. Cabe sempre lembrar que essa ferramenta possui limitações ligadas aos 3 aspectos que ela se propõe a atender - acessibilidade, relevância das estruturas implementadas e cobertura - em virtude das limitações de tempo impostas por um mestrado e pela inovação que a proposta em si representa, o que implica encontrar problemas ao longo do desenvolvimento do projeto. Uma das contribuições desta dissertação é ter uma versão bem acabada de um projeto de simulador categorial que represente tanto conceitos categoriais quanto conceitos funtoriais. Tal contribuição permite que, posteriormente, sejam realizados trabalhos mais avançados em cima deste.

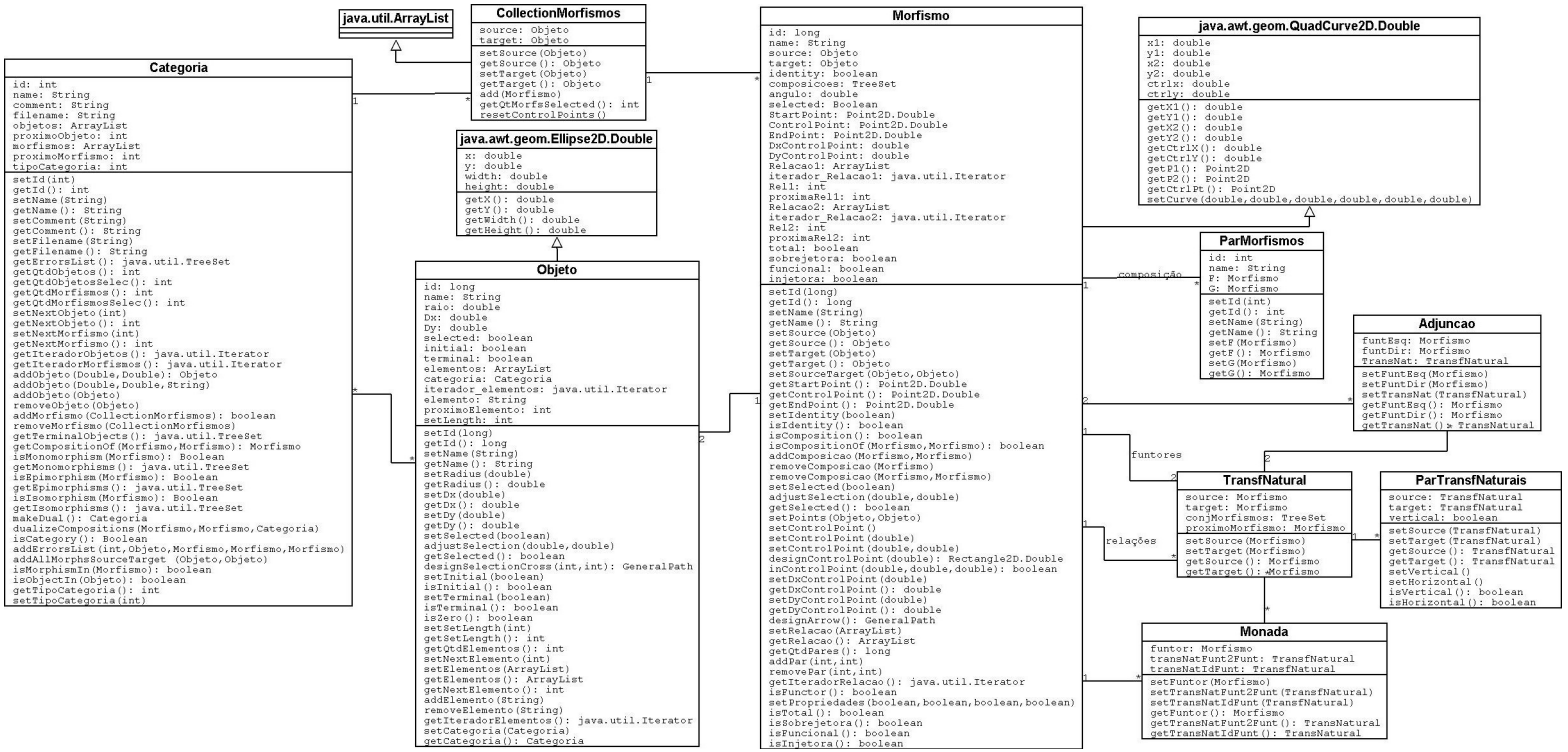


Figura 5.5: Diagrama de Classes do Pacote br.ufrgs.inf.catres.categoria

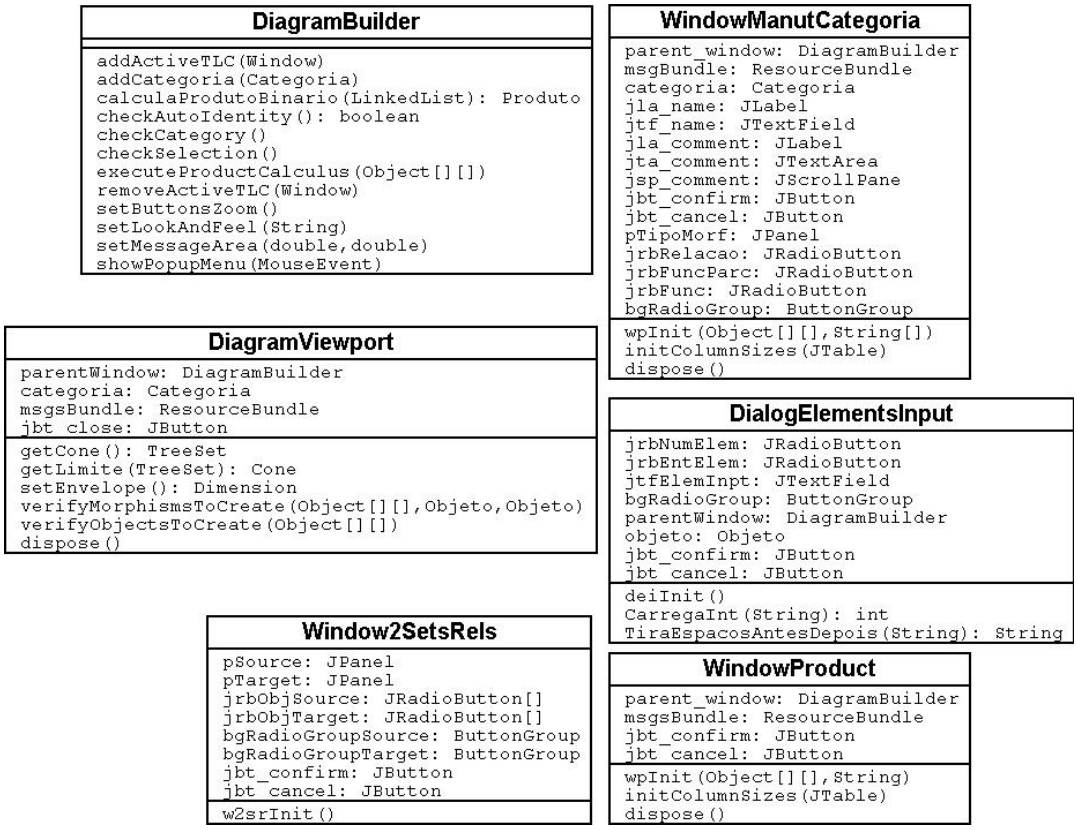


Figura 5.6: Diagrama de Classes parcial do Pacote br.ufrgs.inf.catres.catresui

6 INTEGRAÇÃO COM AMBIENTE HYPER-AUTOMATON

Neste capítulo é feita uma apresentação do ambiente de Sistema de Gerenciamento para Ensino a Distância (SGEAD) chamado Hyper-Automaton, o qual permite, através de hipertextos, que sejam elaborados cursos na web e, a partir do qual, é possível vislumbrar uma integração do CaTReS e, assim, aproveitar seus recursos para o ensino a distância.

6.1 Ambiente Hyper-Automaton

O ambiente Hyper-Automaton foi desenvolvido no contexto da dissertação de mestrado de Júlio Henrique Araújo Pereira Machado [MAC2000]. O objetivo central de seu trabalho era estudar a aplicação de autômatos finitos com saída - Máquina de Mealy e Máquina de Moore - como um modelo estrutural para a organização de hiperdocumentos instrucionais, em especial de cursos na web.

Uma premissa da dissertação de Júlio é que cursos instrucionais da web podem ser vistos como autômatos. Vendo dessa maneira, podemos dizer que as tuplas dos autômatos de cursos apresentam uma correspondência às estruturas de hiperdocumentos na web. O alfabeto de símbolos de entrada é um conjunto de nomes que identificam os links de navegação do hipertexto do curso. O estado inicial corresponde a ponto de entrada inicial de um curso, comumente chamado de *homepage*. Note que a noção de palavra permanece, pois as saídas (de um alfabeto) estão relacionadas a unidades de informação (conteúdo do curso) constituías por páginas web e as palavras de saída, presentes na função programa e função saída, são páginas concatenadas como uma única página web no navegador do usuário. As transições, definidas na função programa, funcionam como ligações lógicas entre os conteúdos dos cursos e definem possíveis links HTML a serem selecionados pelos alunos durante a navegação pelo hipertexto.

Optou-se ainda por uma solução que não é a padrão, pois se verificou que, com freqüência, em cursos pequenos, cada estado tem uma saída diferente, ou seja, cada página de conteúdo do curso corresponde a exatamente um estado. Esta solução foi criada para facilitar a especificação da função programa do autômato pelo professor e, conseqüentemente, a utilização de uma interface mais simples e direta.

Uma conseqüência direta da modelagem utilizada é que o conteúdo instrucional deve ser particionado por várias páginas de hipertexto, geralmente pequenas, que no todo formam um tópico a ser apresentado em um curso.

Em projetos de software para a Internet, a web aparece como uma estrutura básica que pode ser utilizada para a criação e apoio a ambientes de aprendizagem. A estrutura *world wide web* (www) é um núcleo de protocolos de transporte, interfaces e armazenamento de dados sobre o qual são definidas camadas de ferramentas para complementar uma estrutura já existente, visando torná-la mais apta para o suporte às necessidades de aplicações

mais específicas.

Como um complemento à estrutura Internet, o sistema consiste em um ambiente de apoio interativo ao professor, com recursos que facilitam uma representação gráfica do curso, assim como uma interface amigável para os serviços disponibilizados (disponibilização e manutenção de cursos).

Foi adotada uma arquitetura cliente-servidor para o sistema na. O servidor é formado por um conjunto de programas CGI - desenvolvidos na linguagem Perl - alocados em um servidor HTTP e oferece serviços de controle sobre as estruturas dos autômatos e acesso à hiperbase. Um cliente pode ser qualquer navegador com capacidade de execução de applets Java. As páginas HTML e o material hipermídia em geral estão localizados no servidor HTTP em questão. O sistema não utiliza nenhum editor em HTML integrado, podendo o professor utilizar o programa com o qual está mais familiarizado.

O autor de um curso terá acesso a uma interface gráfica com suporte à notação de grafos para a criação e manipulação dos autômatos, utilizando-se as facilidades multiplataforma da linguagem Java. Além de facilitar a visualização do autômato, pretende-se que a interface em Java implemente operações sobre caminhos do curso, com remoção e inserção de dados no autômato, bem como composição de autômatos com ou sem identificação de partes. Tais operações, construídas formalmente sobre o fecho reflexivo transitivo do autômato em nível de transições, serão coerentemente definidas por construções categoriais, a fim de manter consistente os autômatos. O uso de categorias deve-se ao fato do alto poder de expressividade de suas construções, que permitem definir operações de alto nível.

Os módulos CGI e Perl são responsáveis pela edição dos autômatos e controle da navegação através dos cursos. Estes scripts são encarregados da manutenção da base de dados dos autômatos (definição do alfabeto de entrada, cadastramento dos documentos HTML, definição das transições entre os estados) e o acesso à hiperbase, realizando a concatenação de vários documentos HTML em uma única página no navegador do usuário.

6.2 CaTReS no Ambiente Hyper-Automaton

O CaTReS foi elaborado como uma applet Java para, entre outras razões, atingir o propósito de torná-lo acessível através do ambiente web. Como foi citado, esse trabalho também influi em um contexto de ensino a distência, já que simuladores computacionais são ferramentas que facilitam um acesso visual ao conceito formal. Assim, utilizar o CaTReS no contexto de um curso de categorias através da web é uma alternativa bastante interessante.

No Hyper-Automaton, os cursos são modelados como autômatos finitos com saída, de modo que o conteúdo do curso é armazenado em pequenos scripts HTML, os quais são fornecidos e combinados pelo estrutor. Como o CaTReS foi elaborado como uma applet Java, e esta pode ser executado através de um navegador através de um documento HTML que possua a marca adequada de applet Java - desde que a JVM esteja adequadamente instalada na máquina -, pode-se armazenar no ambiente Hyper-Automaton unidades instrucionais para cursos de Teoria das Categorias que são apenas tags parametrizadas para execução do CaTReS.

Como o CaTReS representa conceitos que passam pelas idéias iniciais de Teoria das Categorias e vai até conceitos intermediários, ligados à propriedades funtoriais, trata-se de uma possibilidade interessante elaborar um curso onde se utilize o CaTReS como uma ferramenta de apoio para absorção de conceitos, a medida que estes vão sendo apresenta-

dos no curso.

Tal idéia encontra base em uma idéia desenvolvida em [MAC2000], que trata de utilizar operações categoriais no Hyper-Automaton parta realizar composição de cursos. Tal situação é apresentada na figura 6.1.

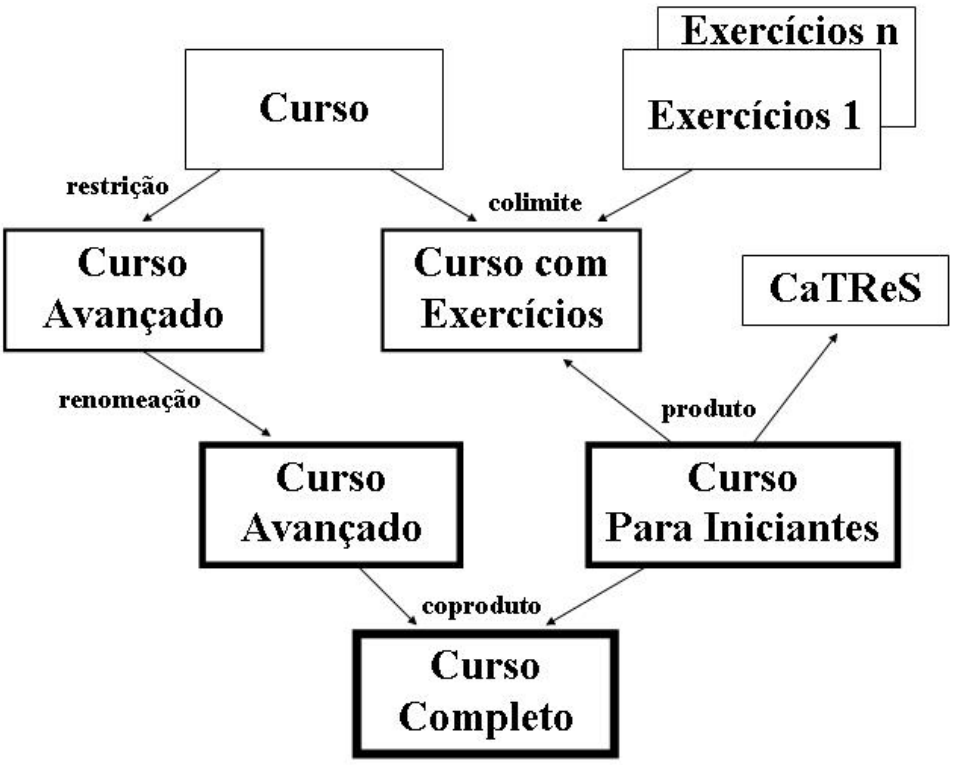


Figura 6.1: Construção de cursos compostos

O exemplo mostra como operações categoriais podem ser usadas como um esquema de composição de cursos no Hyper-Automaton de maneira a construir novos cursos a partir de um pequeno conjunto de cursos existentes. Supondo que tenhamos elaborado somente um curso (padrão), uma série de exercícios e tenhamos a disposição o CaTReS, representadas na figura pelas caixa de tracejado mais fraco, cujo conteúdo faz parte do autômato original do curso para uma dada hiperbase. As caixas de tracejado mais forte contêm os cursos formados através de operações categoriais, como é mostrado nas setas etiquetadas. A caixa `Curso` representa, por exemplo, a versão hipertexto do livro "Teoria das Categorias para Ciência da Computação" [MEN2001], contendo textos e figuras ilustrativas; a caixa `CaTReS` contém um conjunto de páginas HTML projetadas para executar o CaTReS, a fim de simular categorias e conceitos afins, por exemplo; as caixas `Exercícios 1` até `Exercícios n` representam uma base de dados de questões; a caixa `Curso Completo` é o curso resultante desejado.

Neste contexto, suponha que o professor deseja melhorar o seu curso padrão e decida incluir uma base de dados de questões que ele já tenha montado, a fim de que estudantes possam acessar os exercícios adequados para cada capítulo estudado. O objeto resultante da operação categorial de colimite vai resultar neste curso.

Em um passo seguinte, o professor quer construir um curso contendo links anexados ao conteúdo das páginas apontando para o CaTReS, o qual auxilia o estudante a apren-

der os conceitos introduzidos nas páginas seguintes. O objeto resultante da operação categorial produto entre `Curso com Exercícios` e `CaTReS` permite que estudantes acessem a applet Java CaTReS em todas as páginas.

De maneira a atender pedidos de alunos mais experientes em Teoria das Categorias, o professor poderia querer construir um curso com um caminho alternativo "com atalhos" (evitando passos que, sob a ótica de um aluno mais experimente, sejam demasiadamente simples), gerando uma versão avançada dos conteúdos do livro, sem o CaTReS e sem exercícios. Um `Curso Avançado` é o objeto resultante de operações categoriais de restrição e de renomeação, de maneira a retirar transições etiquetadas indesejáveis e reetiquetar transições para o novo propósito. Assim, o `Curso Completo` é o objeto resultante de uma operação categorial de coproduto entre a versão ampliada do curso (`Curso para Iniciantes`) e a versão simplificada do curso (`curso avançado`).

Tais construções envolvendo cursos, exercícios e simuladores computacionais foram retiradas de [MAC2000]. Maiores informações a respeito do sistema Hyper-Automaton podem ser obtidas nessa dissertação.

7 MANUAL DO USUÁRIO

Neste capítulo se dedicará a descrever como o usuário deve interagir com o CaTReS para executar as funcionalidades nele implementadas. A única presunção aqui assumida é a de que o leitor conheça Teoria das Categorias. Não é exigido conhecimento prévio de CaTLeT, e tampouco de qualquer outro simulador categorial. Para uma melhor compreensão, buscou-se utilizar numerosas imagens da ferramenta, a fim de melhor expor o que é descrito.

Este manual inicialmente identifica item a item os componentes de interface da tela principal da aplicação. Após é descrito o processo de criação de uma Categoria - pressupondo que o software esteja configurado conforme o padrão - e explica as verificações que são feitas durante o processo.

7.1 Elementos de Interface

A interface do CaTReS permite a construção e manipulação gráfica de grafos, sua validação categorial, a realização de cálculos sobre a estrutura construída e a criação automática de morfismos, dependendo da preferência selecionada ou da opção de menu selecionado.

A figura 7.1 mostra a tela principal do CaTReS e numera seus componentes. Já a tabela 7.1 enumera um a um os componentes de interface do CaTReS, os quais são descritos nas subseções a seguir.

7.1.1 Área de Menu

Reúne opções para acionamento de grande parte das ações disponíveis ao usuário. Apresenta-se subdivida em cinco itens:

- **Aplicação:** agrupa predominantemente as operações de entrada e saída (carga, salvamento e fechamento de categorias);
- **Preferências:** possui item para apresentar as composições existentes na categoria e outro item que permite personalizar configurações do software;
- **Categoria:** permite a realização de cálculos categoriais e a inserção na categoria de todos os morfismos possíveis entre dois objetos, observando a estrutura representada nos objetos;
- **Funtor:** oferece algumas opções de criação e listagem de funtores herdadas do CaTLeT. A presença deste menu no CaTReS tem como finalidade apenas preservar uma ligação histórica com o seu antecessor. Entretanto, recomenda-se que tal menu

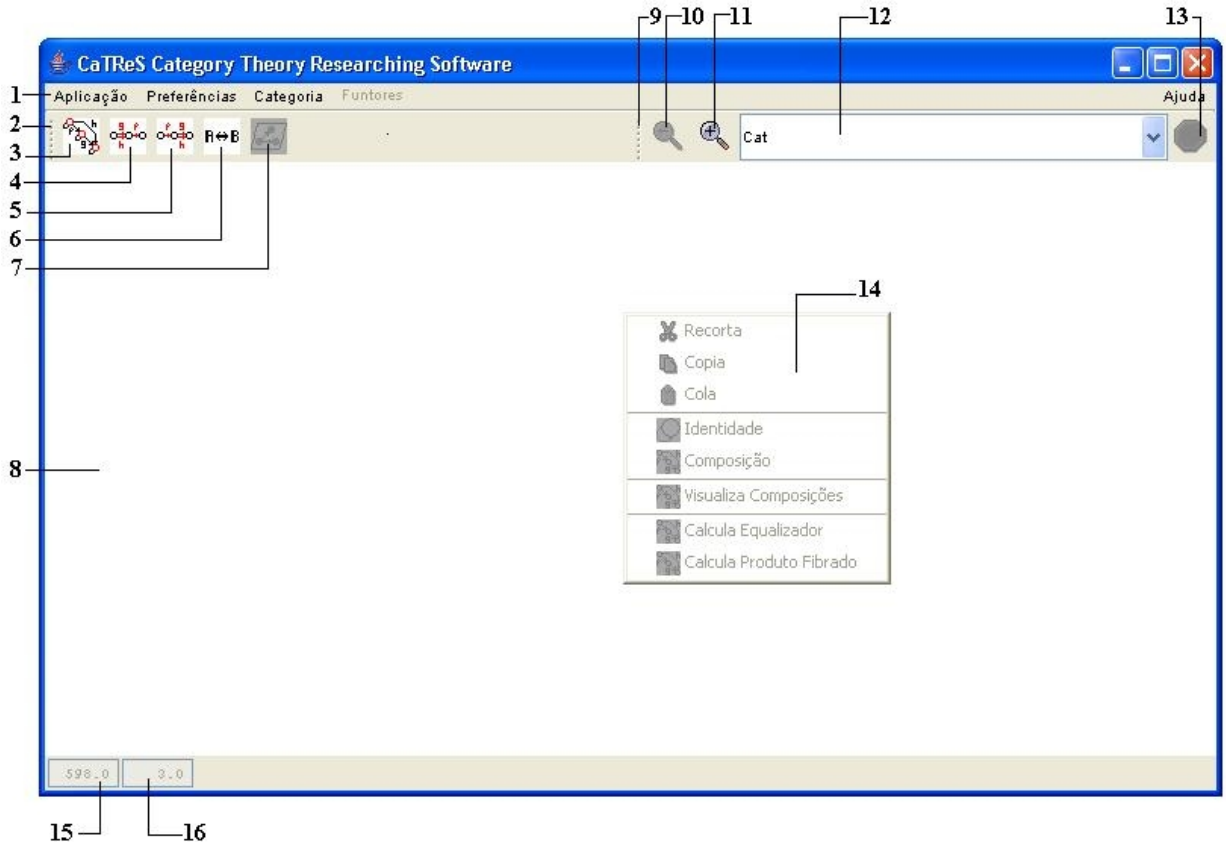


Figura 7.1: Tela Principal do CaTReS

não seja utilizado, pois o CaTReS possui meios mais completos e elegantes de lidar com funtores;

- **Ajuda:** área que reúne acesso à documentação de apoio ao usuário.

7.1.2 Barra de Ferramentas de Cálculos Categoriais

São botões com funcionalidades complementares ao menu Categoria, permitindo a verificação de quais dentre os morfismos da categoria são monomorfismos (item 4 da figura 7.1), epimorfismos (item 5 da figura 7.1) e isomorfismos (item 6 da figura 7.1).

Adicionalmente, a criação de novas categorias é feita através de um dos botões da barra (item 3 da figura 7.1). A edição e visualização de diagramas (item 7 da figura 7.1) permite a realização dos cálculos categoriais que envolvem diagramas (como limite).

7.1.3 Área de Trabalho

É o painel onde os grafos são desenhados.

7.1.4 Barra de Ferramentas de Opções Utilitárias

Agrupa os botões de zoom (itens 10 e 11 da figura 7.1), a combobox utilizada para visualização das categorias carregadas em memória pela aplicação - bem como da atualmente ativa - e um botão cuja finalidade é indicar se o grafo do diagrama ativo atende ou não à definição de categoria. Estando este botão habilitado, o diagrama ativo não constitui

Tabela 7.1: Enumeração dos componentes de interface do CaTReS

Identificador	Descrição
1	Área de Menu
2	Barra de Ferramentas de Cálculos Categoriais
3	Botão “Nova Categoria”
4	Botão “Monomorfismo”
5	Botão “Epimorfismo”
6	Botão “Isomorfismo”
7	Botão “Edição e Visualização de Diagrama”
8	Área de Trabalho (painel onde são desenhadas as categorias)
9	Barra de Ferramentas de Opções Utilitárias
10	Botão “Mais Zoom”
11	Botão “Menos Zoom”
12	Combo-Box das categorias ativas, para seleção da categoria corrente
13	Botão de mensagens de erro. Só habilitado se grafo corrente não representa categoria
14	Menu de Contexto
15	Coordenada Vertical
16	Coordenada Horizontal

uma categoria, e basta pressioná-lo para verificar detalhadamente cada falha do diagrama perante a definição categorial.

7.1.5 Menu de Contexto

O menu de contexto é acionado conforme a plataforma. Em ambiente Windows, usualmente o acionamento deste ocorre pelo uso do botão direito do mouse.

Este menu contém quatro grupos de opções. A presença de todas as opções colocadas neste menu justifica-se pela dependência destas opções ao conjunto de objetos e morfismos selecionados pelo usuário da categoria corrente.

O primeiro grupo comporta-se da seguinte forma: as opções “Recorta” e “Copia” são habilitadas se qualquer objeto ou morfismo estiver selecionado. A opção “Cola” habilita se houver algum objeto recortado ou copiado na área de transferência;

O segundo grupo, integrado pelas opções “Identidade” e “Composição”, habilita independentemente de o grafo representar ou não uma categoria, uma vez que são funções auxiliares à construção desta. Atendem, entretanto, a restrições específicas de cada função:

- Identidade: o único componente selecionado deve ser um endomorfismo;
- Composição: os únicos componentes selecionados devem ser três morfismos. Adicionalmente é verificado se algum deles pode constituir a composição de dois outros, em todas as combinações possíveis.

O terceiro grupo é formado somente pela opção “Visualiza Composições”. Aqui, tal qual a opção presente no menu “Preferências”, é possível visualizar as composições existentes na categoria representada.

O quarto e último grupo, composto pelas opções de cálculo “Equalizador” e “Produto Fibrado” habilitam-se segundo as seguintes regras:

- Equalizador: os únicos componentes selecionados da categoria são dois morfismos paralelos (mesmos objetos origem e destino);
- Produto Fibrado: os únicos componentes selecionados da categoria são dois morfismos com mesmo objeto destino.

7.2 Processo de Criação de uma Categoria

A criação de uma categoria começa pelo botão "Nova Categoria" (item 3 da figura 7.1), o qual, ao ser pressionado, apresentará a janela de diálogo representada na figura 7.2.

Figura 7.2: Janela de Diálogo que Cria uma Nova Categoria

A partir desta janela, o usuário define propriedades da categoria a ser criada. Enquanto algumas propriedades são apenas informativas ou relevantes para acessar a categoria - ou seja, não impactando no modo como o programa lida com a categoria -, outras referem-se a características que impactam na maneira como o software trabalha com a categoria.

No primeiro caso estão as informações referentes ao nome da categoria e comentários. No segundo, encontram-se a definição do tipo de morfismo permitido na categoria e restrições sobre o tipo escolhido, a partir da escolha de propriedades que tal morfismo tem que respeitar. No caso do CaTReS, os tipos de morfismos representáveis são derivações de relação (função, função parcial e relação) e funtores. Somente as derivações de relações podem ser restringidas através da seleção de algumas das propriedades das relações. Ao selecionarmos a opção Funtor, as opções que definem as propriedades de relações ficam desabilitadas.

Convém resaltar que, ao escolhermos uma derivação de relação que não seja a própria, a propriedade referente a ela fica selecionada e desabilitada. Na figura 7.2, por exemplo, aparece selecionado o morfismo Função (Total). Em virtude disso, as propriedades Funcional e Injetora automaticamente aparecem selecionadas e desabilitadas no quadro. Entretanto, no CaTReS, o contrário não é verdadeiro (a seleção das propriedades Funcional

e Total não implica a automática seleção do tipo de morfismo Função).

Dentro do escopo de relações, as propriedades relacionais implementadas não são mutuamente exclusivas. Por exemplo, uma monorrelação será necessariamente uma relação injetora e total. Essa consistência é feita pelo CaTReS - ou seja, a seleção da propriedade monorrelação ativa automaticamente as propriedades injetora e total. Nesse caso, o contrário é verdadeiro (ao selecionarmos injetora e total, monorrelação é selecionada). O mesmo vale quando uma opção é desmarcada: ao desmarcarmos injetora ou total, monorrelação é desmarcada, e se desmarcamos monorrelação, injetora e total são desmarcadas - a menos que trate-se de um tipo de morfismo onde não seja possível desmarcar a propriedade (por exemplo, se o tipo de morfismo é uma função, ao desmarcarmos a monorrelação, a propriedade total continuará marcada).

Observando o que foi descrito nos dois últimos parágrafos, verificamos que o CaTReS realiza uma consistência em suas propriedades, mas que isso não afeta o tipo de morfismo selecionado. Contudo, o tipo de morfismo rege as propriedades de relação, impedindo que um determinado morfismo deixe de possuir uma propriedade intrínseco a si.

No caso de selecionarmos um dos morfismos derivados de relação e clicarmos em Ok, aparece o nome da categoria criada na combo-box de categorias ativas, tornando esta uma categoria ativa e, no momento, a categoria corrente. A criação de objetos ocorre através do clique na área de trabalho, o qual provoca a criação da janela de diálogo da figura 7.3.

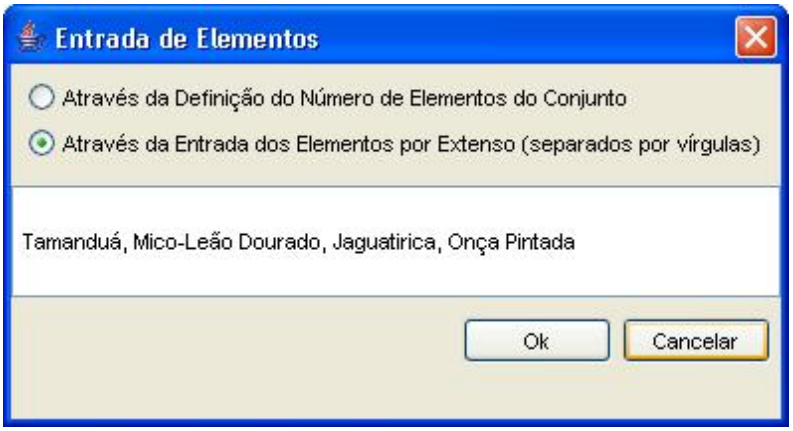


Figura 7.3: Janela de diálogo para a entrada de elementos do conjunto

Como estamos criando uma categoria baseada em morfismos derivados de relações, os objetos criados são conjuntos. Estes podem ser definidos de duas maneiras:

- Através da cardinalidade do conjunto;
- Através da enumeração dos elementos.

O tratamento de cardinalidade do conjunto é uma herança do CaTLeT, e pode ser utilizado em virtude do fato de que, dentro dos contextos categoriais em que tais construções se inspiram, dois conjuntos com idêntico número de elementos, não importa quais elementos sejam estes, são isomorfos - o que, no contexto de Teoria das Categoriais, significa que eles são essencialmente iguais.

Assim sendo, se a opção “Através da Definição do Número de Elementos do Conjunto” estiver selecionada, basta entrar com o número de elementos do conjunto na área

de edição e pressionar Ok. Por exemplo, caso tenha sido entrado o número 5, o usuário estará criando um conjunto de 5 elementos, sem especificar quais são estes elementos.

Contudo, no caso da opção “Através da Entrada dos Elementos por Extenso (separados por vírgulas)” estar selecionada (caso da figura 7.3), o usuário deverá entrar com os elementos do conjunto a ser criado. No exemplo da figura 7.3, se for pressionado Ok, estará sendo criado um conjunto de 4 elementos referentes, por exemplo, a animais silvestres.

Cada objeto da categoria pode ser criado por qualquer um dos dois meios, não havendo a restrição de todos os objetos da categoria terem que ser estruturados pelo mesmo método (cardinalidade ou entrada de elementos).

Na figura 7.4 temos a categoria C, já criada, com dois objetos. Na janela ativa podemos ver os elementos de um destes objetos.

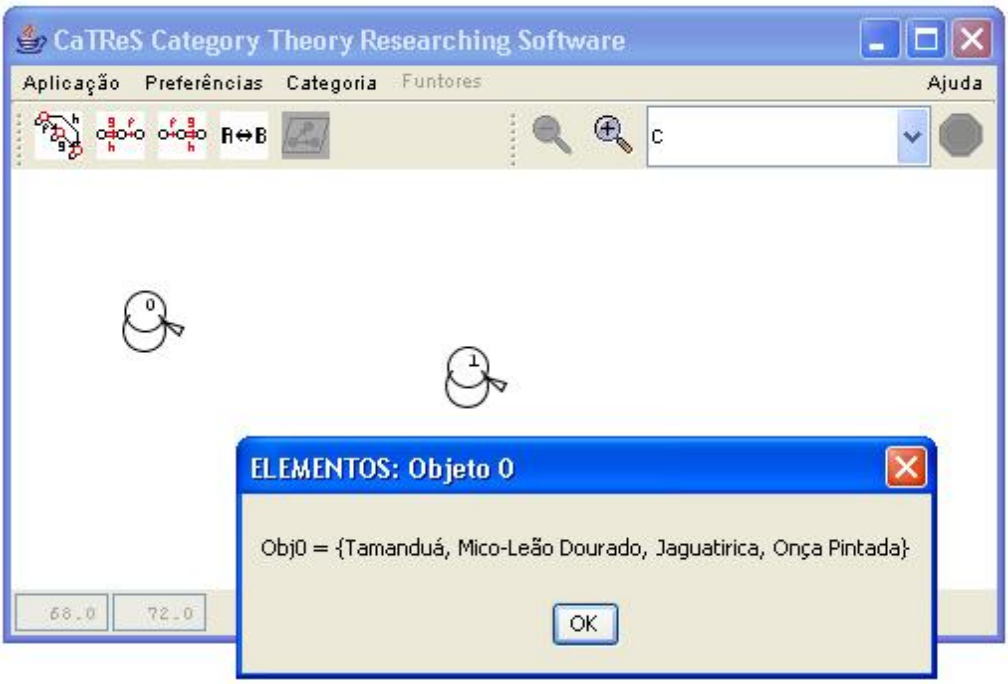


Figura 7.4: Categoria C, com dois objetos e os elementos de um deles sendo mostrado

A janela que aparece ativa é apresentada quando o usuário clica sobre um objeto existente na categoria. Além de mostrar os elementos deste objeto, tal clique dispara o processo inicial da criação de um morfismo, como é mostrado na figura 7.5

Na figura, observa-se uma linha a qual está sendo arrastada. Ela acompanha o movimento do mouse. Quando o usuário clicar em um objeto - por exemplo, no objeto 1 - uma janela de diálogo para que seja realizada a entrada do morfismo será apresentada. Tal janela é mostrada na figura 7.6.

Na janela são mostrados os elementos dos objetos origem e destino. No caso do exemplo da figura, temos como origem um objeto criado por entrada extensiva dos elementos e como destino um objeto provavelmente - mas não necessariamente - criado pela entrada da cardinalidade do objeto. Supondo que de fato o objeto destino tenha sido definido pela cardinalidade, embora este não tenha elementos definidos, na criação de um morfismos é exibido um nome padrão para cada um de seus elementos - a partir de uma letra e um indexador -, a fim de facilitar a criação do morfismo relação.

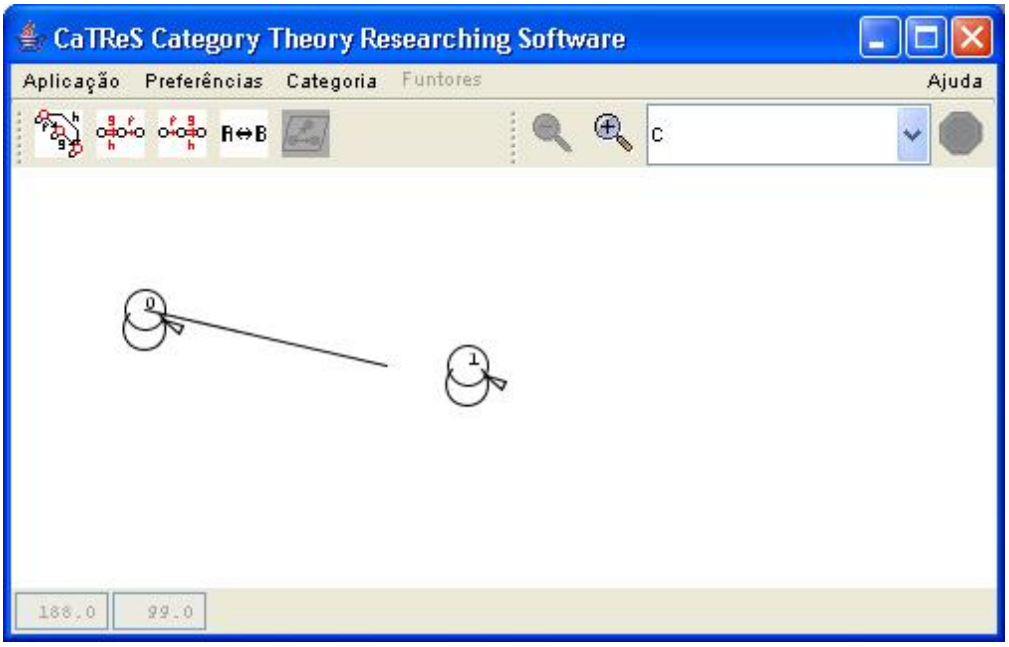


Figura 7.5: Início do processo de criação de um morfismo

Ainda na mesma figura, se for clicado o botão Ok, possivelmente será criada uma relação definida conforme o conteúdo entrado pelo usuário na caixa branca de edição. Contudo, se a entrada não for bem formada (por exemplo, entrar-se com “Tamanduá - B7”) ou se a relação entrada não representar o tipo de morfismo definido na criação da categoria ou se a relação não satisfizer alguma das propriedades selecionadas na criação da categoria, o programa retornará uma mensagem de erro como, por exemplo, a que aparece na figura 7.7.

No CaTReS, por padrão, a criação de um objeto provoca a criação automática do morfismo identidade, e a criação de dois morfismos componíveis gera automaticamente a criação do morfismo composição. Na figura 7.8, montada artificialmente (não é possível exibir uma tela assim no CaTReS), é possível visualizar-se o conteúdo dos morfismos 2 e 4, criados manualmente pelo usuário, e o conteúdo do morfismo 5, criado automaticamente pelo sistema.

Até o momento, estamos trabalhando com uma categoria criada com base em morfismos derivados de relações. Mas, e se a categoria criada tem como base funtores (ver figura 7.2)? Nesse caso, os objetos criados (assim como na situação anterior, basta clicar na área de trabalho) dão origem a novas categorias, como podemos observar na figura 7.9.

Na figura, tornamos a ver a janela de criação de uma categoria - a mesma que é disponibilizada ao pressionarmos o botão “Nova Categoria” -, mas desta vez com o botão Funtor desabilitado. Isso ocorre porque o CaTReS não suporta a criação de objetos que representem categorias de categorias.

7.3 CaTReS Passo a Passo

Nesta seção cada elemento de interação do o usuário presente no CaTReS será apresentado. Assim, este autor espera estar tornando o manual do usuário completo o suficiente para sanar as dúvidas de qualquer natureza referentes ao uso dos recursos do

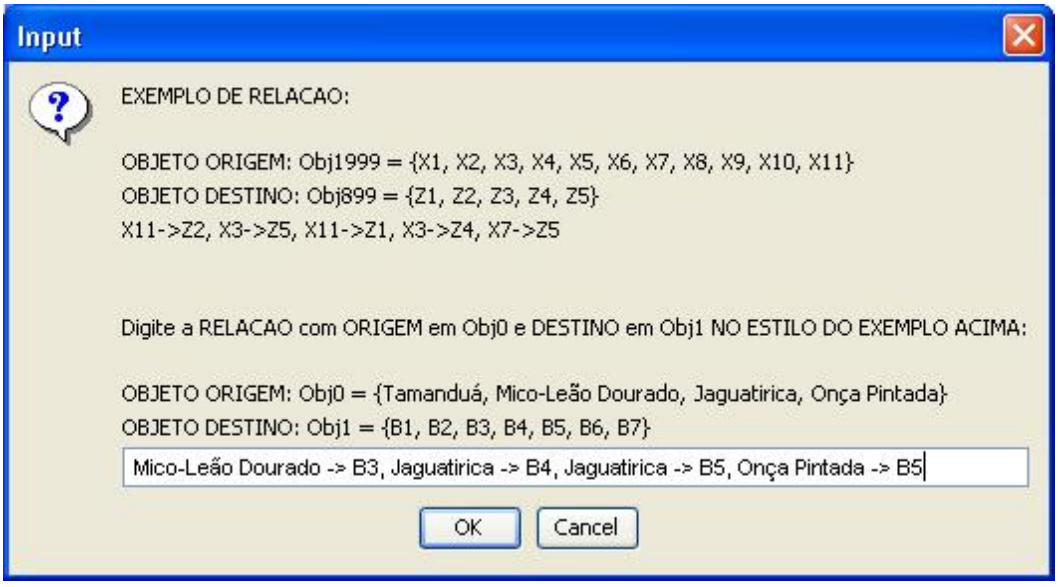


Figura 7.6: Janela de Diálogo para a Entrada de um Morfismo

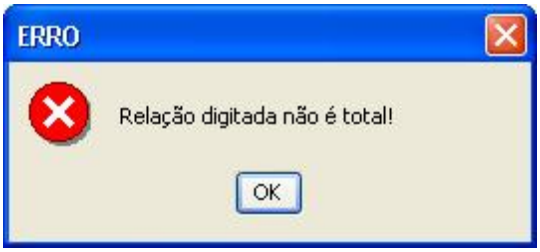


Figura 7.7: Exemplo Mensagem de erro apresentada ao tentar criar-se um morfismo

aplicativo.

Elementos que levem ao uso de funcionalidades que requeiram uma descrição mais detalhada a terão. Contudo, elementos cujas funcionalidades sejam mais facilmente compreensíveis terão uma explicação mais sucinta.

7.3.1 Área de Menu

O passo a passo deste manual começará pela descrição detalhada das opções disponíveis na Área de Menu - item 1 da figura 7.1. A seguir são apresentadas as opções de cada menu.

7.3.1.1 Aplicação

A figura 7.10 mostra as 5 opções disponíveis no menu Aplicação, as quais são brevemente descritas a seguir.

- Abrir Categoria: Permite que uma categoria armazenada em formato cat - formato próprio do CaTReS, herdado do CaTLeT - seja aberta (figura 7.11);

- Salvar Categoria: Salva o estado da categoria corrente referente ao momento em que a opção foi selecionada. Caso a categoria corrente não tenha sido anteriormente

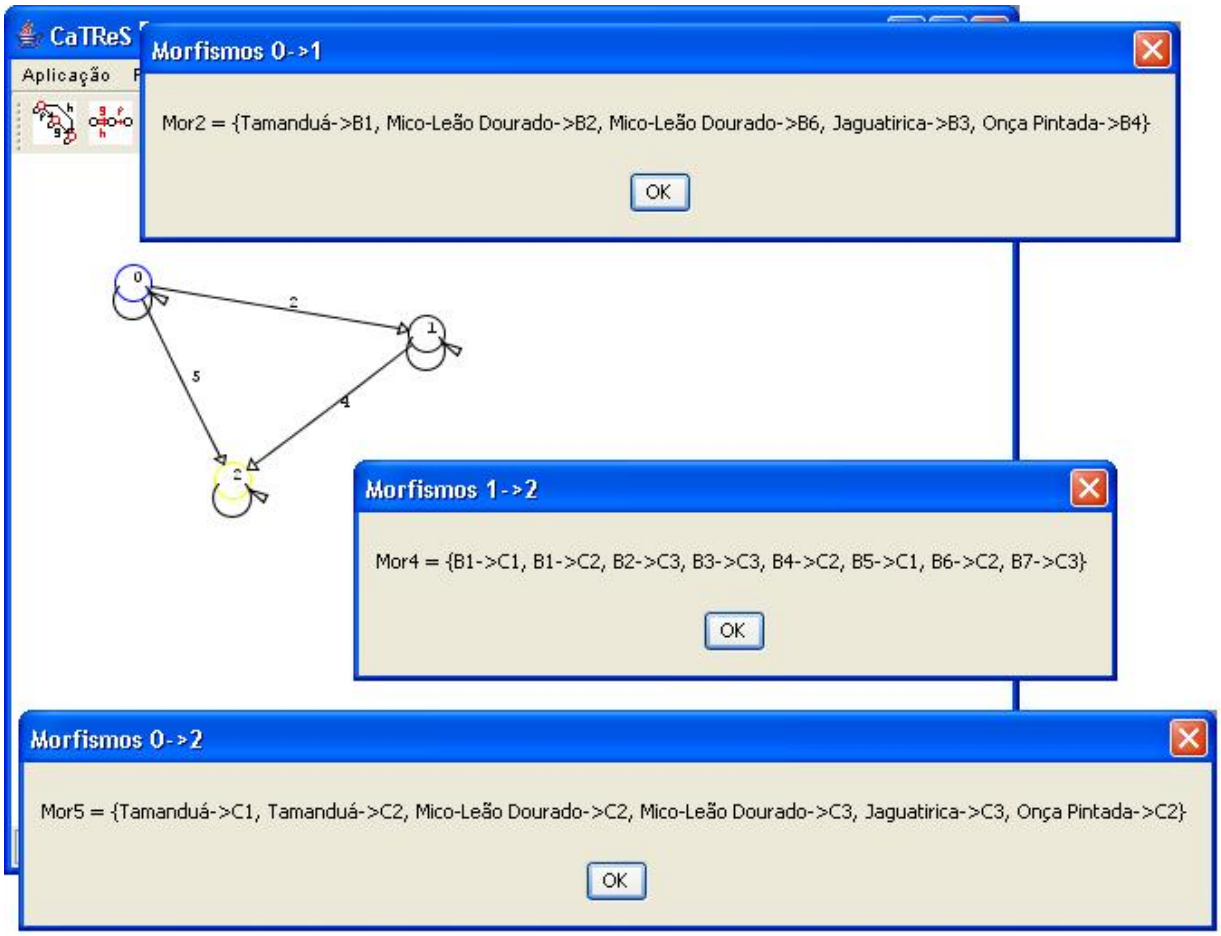


Figura 7.8: Figura montada, onde os morfismos 2 e 4 foram criados manualmente e o 5, automaticamente

salva, uma janela é aberta a fim de que o usuário defina um nome para o arquivo que armazenará o conteúdo da categoria (figura 7.12). Caso contrário, o estado atual da categoria sobrescreverá o anterior;

- Salvar Categoria Como: Tal qual a opção anterior anterior, salva o estado da categoria corrente no momento. Contudo, a seleção desta opção sempre provocará a abertura da janela da figura 7.12, salvando o estado momentâneo da categoria em um arquivo novo;

- Fechar Categoria: Fecha a categoria corrente sem salvar seu estado momentâneo;
- Sair: Fecha o programa CaTReS sem salvar as categorias abertas.

7.3.1.2 Preferências

O menu Preferências oferece ao usuário duas opções de seleção, conforme ilustra a figura 7.13, as quais são descritas a seguir:

- Relatório de Composições: Lista para o usuário os morfismos que são composição de dois outros, seguido de seus componentes. A lista não inclui composições que

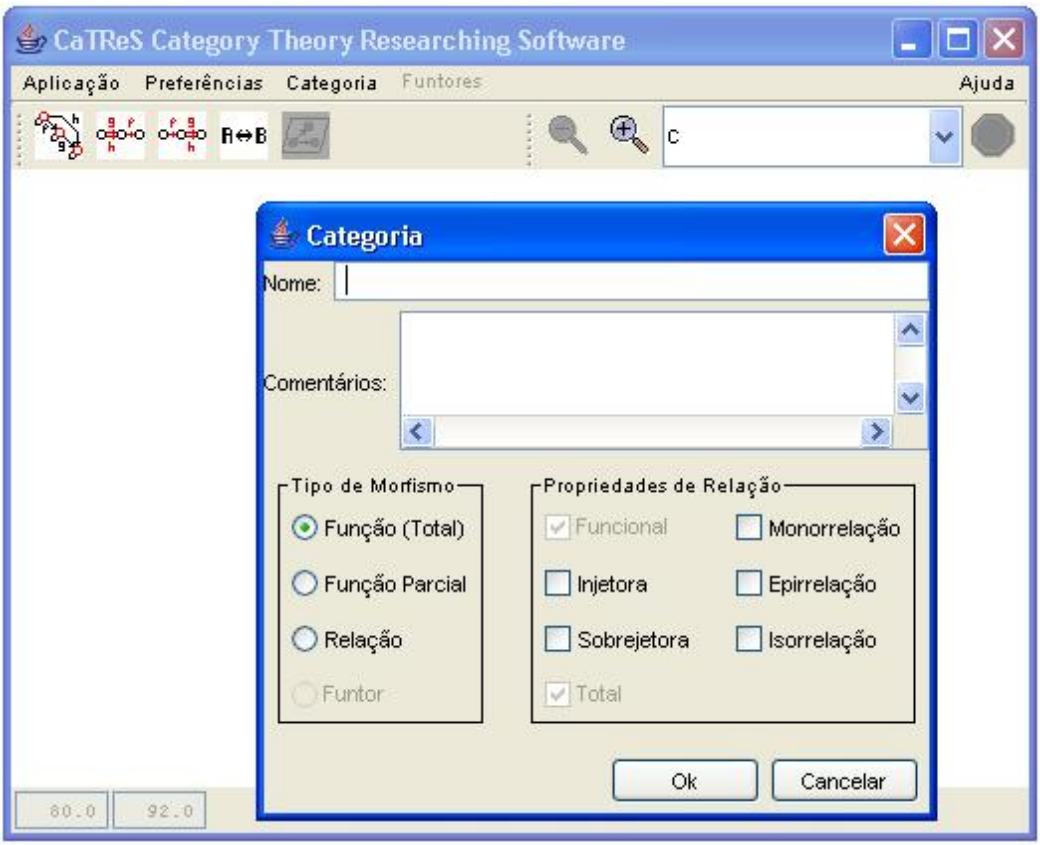


Figura 7.9: Criação de um objeto-categoria em uma categoria baseada em funtores

tenham morfismos identidade como um de seus componentes. A figura 7.14 mostra uma categoria desenhada no CaTReS e a janela Relatorio de Composições explicitando as composições [6] e [8] e seus componentes, [5] e [4], [7] e [4], respectivamente.

- Preferências: Permite ajustar algumas configurações do aplicativo. Ao selecionarmos essa opção de menu, a janela exibida na figura 7.15 - cabe ressaltar que, na figura, a imagem da tela está duplicada, a fim de mostrar todas as opções de configuração - aparece. Nela, encontramos as abas Geral e Cores. Na figura 7.15, a Geral é a tela que aparece à esquerda e a Cores, à direita. As seleções da aba Geral dizem respeito tanto a aspectos de funcionalidade quanto a aspectos de aparência. Já a aba Cores refere-se a aparência, embora explicite propriedades categoriais como objeto inicial:

  – Aba Geral: Os botões de rádio agrupados em *Look & Feel* dizem respeito à aparência dos componentes do CaTReS, sendo 3 as possibilidades de variação. A caixa de checagem Auto Identidade, se marcada, faz com que o software crie um morfismo identidade para cada objeto criado. A seleção caixa de checagem Auto Composição faz com que, no momento da criação de um novo morfismo, o CaTReS verifique quais outros morfismos são componíveis em relação ao novo criado, e crie, posteriormente, as composições adequadas automaticamente. A seleção de Visualizar Identificadores faz com

Figura 7.10: Menu Aplicações

que visualizemos, por exemplo, os números que identificam objetos e morfismos na figura 7.14. A seleção da última caixa de checagem da aba faz com que, na existência de um determinado número de morfismos paralelos, o CaTReS represente graficamente todos os morfismos através de uma única seta - a intenção é evitar a poluição visual.

– Aba Cores: os diversos botões permitem configurar as cores de fundo, dos objetos comuns (sem propriedades especiais), dos objetos iniciais, dos objetos finais, dos objetos zero, dos morfismos comuns (sem propriedades especiais), dos morfismos identidade, dos morfismos composição, dos morfismos que sejam componentes de uma composição (nesses dois últimos casos, excluem-se os casos onde o morfismo identidade é componente) e os morfismos agrupados.

7.3.1.3 Aplicação

A figura 7.16 mostra as opções disponíveis no menu Categoria. Trata-se de um menu que disponibiliza ao usuário cálculos categoriais que não envolvam diagramas e alguns cálculos acessórios sobre categorias.

No submenu Cálculos Categoriais encontram-se duas opções: Criar Dual e Produto. A primeira cria uma outra categoria, a qual é dual à categoria corrente. A segunda permite o cálculo de aridade qualquer entre os objetos existentes.

As outras duas opções de menu referem-se a cálculos acessórios universais. A opção Todas Relações Conj Orig Dest define todas as possíveis relações - com as devidas restrições configuradas na criação da categoria - existentes entre dois objetos, dados como entrada pelo usuário.

Já a opção Todas Relações cria todos os morfismos possíveis entre todos os objetos-conjuntos existentes na categoria.

7.3.1.4 Funtores

Menu existente como herança do CaTLeT. As funcionalidades de criação de funtores são melhores desempenhadas no CaTReS através da criação de uma categoria de categorias.

7.3.2 Barra de Ferramentas de Cálculos Categoriais

Na figura 7.1, item 2, temos a barra de ferramentas de cálculos categoriais, a partir da qual criamos novas categorias, verificamos propriedades de morfismo - mono, epi e iso - e realizamos cálculos categoriais diagramáticos.

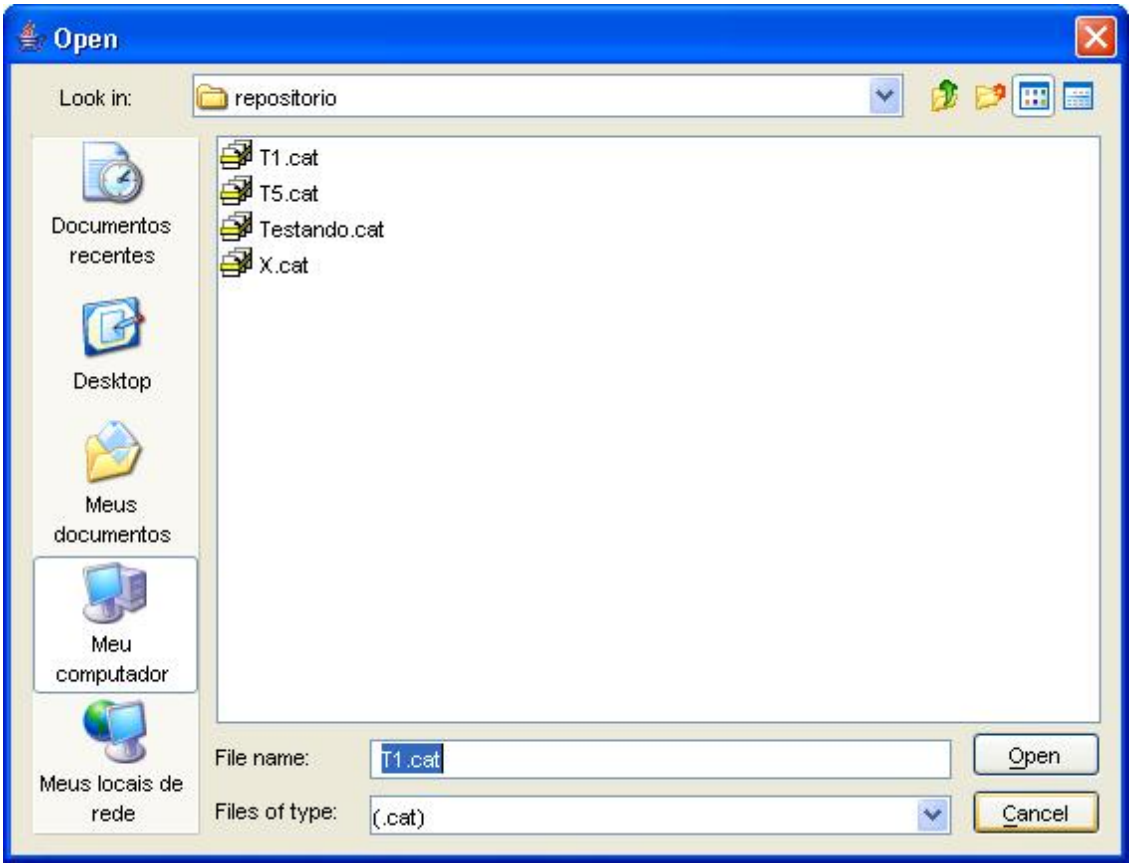


Figura 7.11: Janela aberta ao selecionar a opção *Abrir Categoria*

O clique no botão do item 3 da figura 7.1 abre a janela que é ilustrada na figura 7.2. Nela, são definidas propriedades a respeito da categoria a ser criada. No caso, definem-se o nome da categoria, algum comentário que o usuário deseje acrescentar, o tipo de morfismo que essa categoria irá possuir (função, função parcial, relação ou funtor) e, caso os morfismos sejam derivados de relação, que propriedades relacionais tais morfismos devem seguir.

Como já foi explicado antes, quando selecionamos função, função parcial ou relação, os conjuntos criados são conjuntos, que são definidos por cardinalidade ou por enumeração de elementos. Caso selecionemos funtor, então os objetos serão categorias com morfismos derivados de relação.

O botão representado pelo item 4 da figura 7.1 tem como papel informar ao usuário quais dos morfismos da categoria são monomorfismos. O item 5 da mesma figura, por sua vez, trata-se do botão que informa os epimorfismos. Já o botão do item 6 refere-se aos isomorfismos.

O botão representado pelo item 7, que aparece desabilitado na figura 7.1, permite a representação de diagramas no CaTReS. Assim, cálculos como limite podem ser realizados.

Na figura 7.17 vemos um diagrama criado sobre a categoria X. Para criarmos um diagrama, basta selecionarmos os morfismos e os objetos da categoria os quais desejamos que façam parte do diagrama em questão. Após, basta pressionar o botão representado pelo item 7 da figura 7.1. No caso da figura 7.17, foram selecionados os morfismos 4 e 6 e, após, clicou-se no botão. Como selecionou-se apenas morfismos, o CaTReS considera os objetos origem e destino correspondentes como selecionados também.

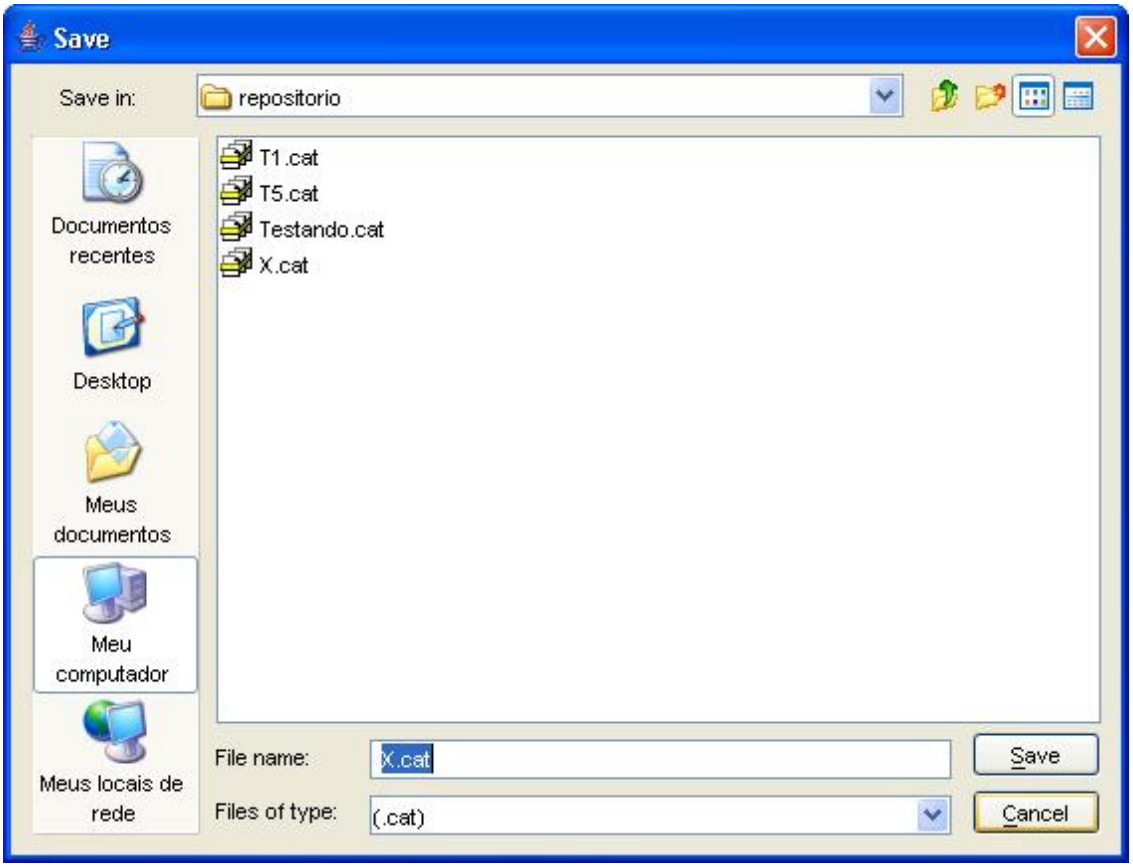


Figura 7.12: Janela referente ao salvamento de uma categoria em um arquivo novo

Figura 7.13: Menu Preferências

Tendo o diagrama e a categoria, podemos realizar os cálculos derivados de limite. No exemplo dado, por exemplo, podemos utilizar o diagrama em questão para calcular o equalizador dos morfismos 4 e 6. O resultado desse cálculo vem através da mensagem explicitada pela figura 7.18.

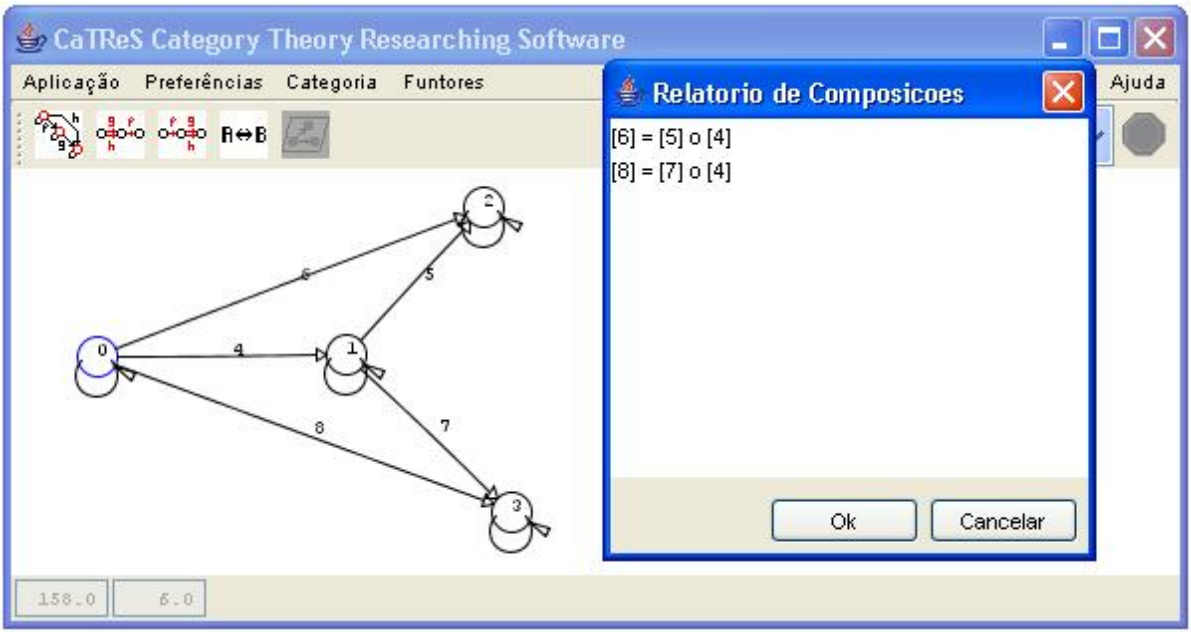


Figura 7.14: Relatório de Composições

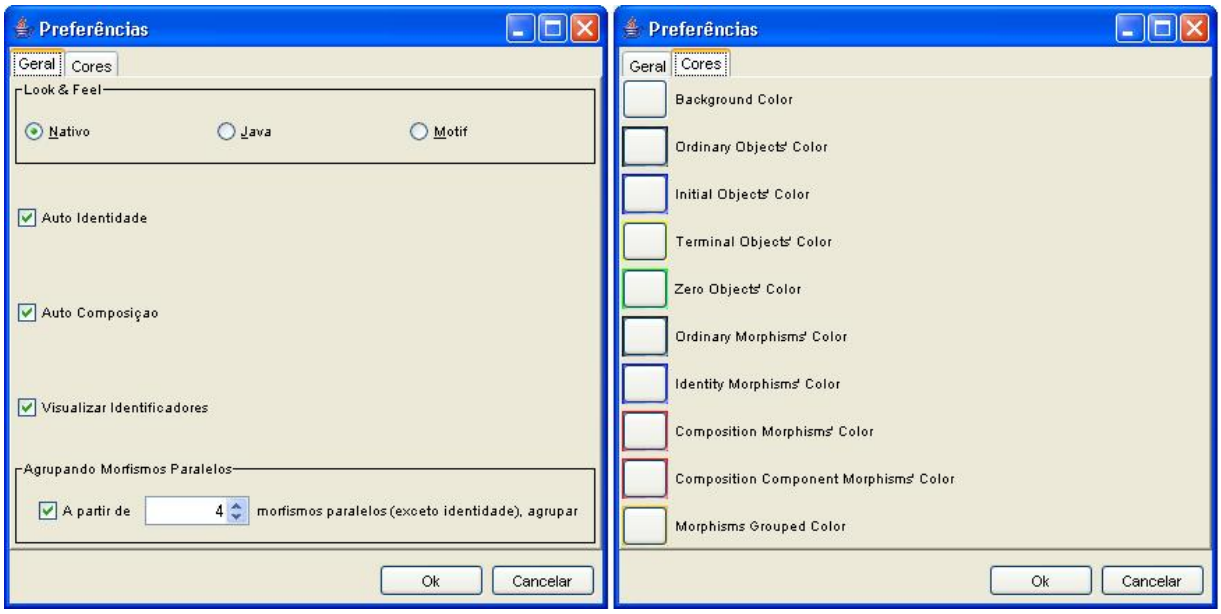


Figura 7.15: Janela Preferências - Tela Utilizada para Realizar Configurações no CaTReS

Figura 7.16: Menu Categoria

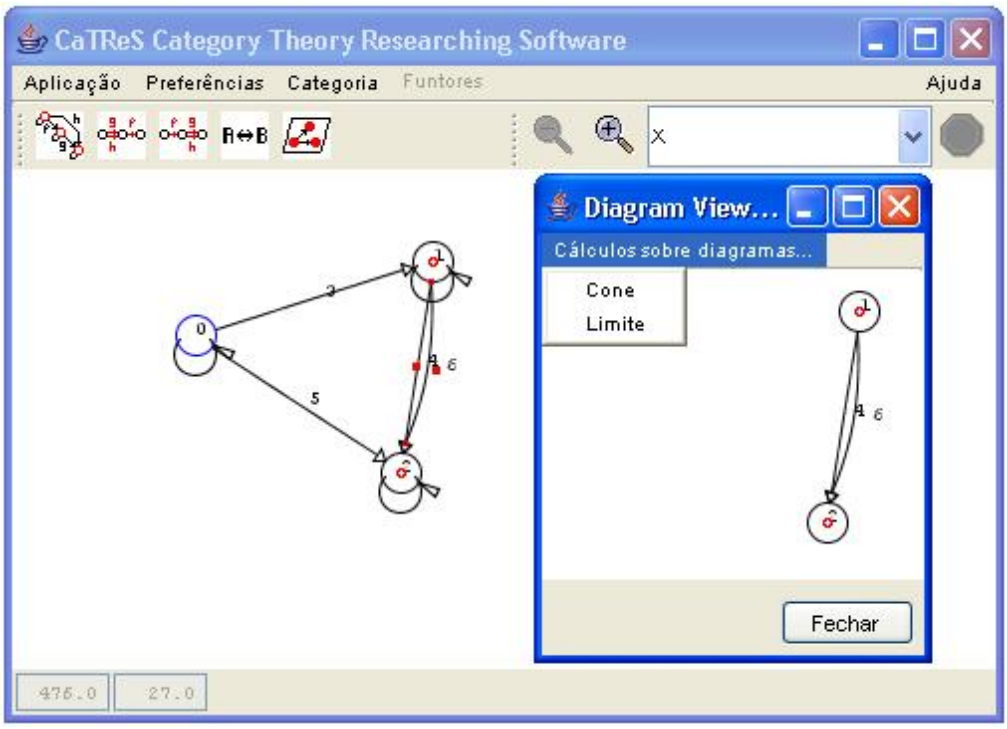


Figura 7.17: Diagrama e Cálculos Diagramáticos

Figura 7.18: Cálculo do Equalizador Representado pelo Diagrama da figura 7.17

8 CONCLUSÃO

Teoria das Categorias, embora venha sendo objeto de pesquisa com maior ênfase em aspectos teóricos, possui um potencial de aplicação que ainda não foi explorado. Esta dissertação apresentou simuladores categoriais como uma alternativa para tornar os conceitos categoriais mais acessíveis para a comunidade científica, facilitando seu uso no campo de pesquisa aplicado.

Observou-se que ainda encontra-se em estágio bastante prematuro o desenvolvimento de simuladores categoriais. Conceitos de extrema utilidade para o trabalho e para a pesquisa em Teoria das Categorias ou não estão adequadamente representados nas ferramentas ou, quando estão, possuem dificuldade de acesso bastante grande, exigindo do usuário o prévio conhecimento de conceitos que não são do interesse direto de Teoria das Categorias.

No contexto apresentado, esta dissertação apresentou três critérios que, segundo a percepção deste autor, compreendem importantes elementos para que um simulador categorial tenha a robustez necessária para aproximar do usuário interessado em conceitos categoriais do aprendizado desejado. Trata-se de boa acessibilidade, alta relevância das estruturas implementadas e alta cobertura. Observou-se também que, enquanto acessibilidade é um critério que, a priori, é independente do contexto de aplicação para o qual se deseja utilizar Teoria das Categorias, os critérios relevância das estruturas implementadas e cobertura devem ser avaliados de acordo com a aplicação que se deseja dar ao simulador categorial. Um critério que não foi incluído - especialmente pelo fato deste autor considerá-lo evidente - diz respeito à precisão conceitual dos conceitos categoriais implementados no simulador. Para a surpresa deste autor, alguns dos simuladores categoriais estudados no contexo dessa dissertação possuem baixa precisão formal, induzindo um eventual aprendiz de Teoria das Categorias a uma percepção errada de alguns conceitos.

Estes critérios - acessibilidade, relevância das estruturas implementadas e alta cobertura - foram utilizados para aferir a qualidade de um simulador categorial, uma vez que:

- um simulador com alta acessibilidade e com estruturas relevantes mas com baixa cobertura possui como deficiência a baixa capacidade que o usuário possui para explorar mais a fundo os relevantes conceitos categoriais representados. Um exemplo do que está sendo dito pode ser observado no CaTLeT (antes da ampliação realizada em [VIE2003]), o qual trata-se de um software bastante acessível, com estruturas adequadas para o seu propósito - apoio ao ensino e aprendizado de conceitos introdutórios de Teoria das Categorias -, mas que não permite ao usuário visualizar outras possibilidades dentro dos conceitos implementados (exemplo: uma categoria podem ser objetos e morfismos atômicos, mas também podem ser conjuntos e relações, conjuntos e funções parciais, categorias e funtores);

- um simulador com alta acessibilidade, com estruturas pouco relevantes implementadas e com alta cobertura é um software de baixa eficiência para o suporte ao propósito para o qual ele foi criado. É conveniente lembrar que, quando se fala de estruturas relevantes, estamos falando de relevância dentro de um propósito para o qual a ferramenta pretende atender ou auxiliar. Um simulador que implementa estruturas pouco relevantes implica ineficácia dentro dos objetivos traçados pelo simulador;

- um simulador com baixa acessibilidade, com estruturas relevantes e com alta cobertura é um aplicativo com grande potencial de atingir os objetivos para os quais ele foi criado, mas com obstáculos na interação homem-computador que, dependendo do grau dos obstáculos, podem restringir muito seu uso prático ou, no mínimo, demandar uma carga de pesquisa e estudo que o torna menos útil do que ele poderia ser. Tal situação pode ser percebida no Computational Category Theory, um simulador categorial bastante poderoso, mas com dificuldades tão grandes de operação que, muitas vezes, requer do usuário um esforço similar ao de um programador em linguagem funcional.

Observada tal situação, foi projetado e implementado o CaTReS, um simulador categorial que consolida os conceitos já existentes no CaTLeT, ampliando sua cobertura, e inova, representando conceitos funtoriais de grande importância para a Ciência da Computação, já que são construções funtoriais que permitem representar idéias complexas com poucos símbolos. Dessa forma, buscou-se contribuir com o estado da arte apresentando uma ferramenta que aglutinasse acessibilidade, estruturas relevantes para o aprendizado e para a pesquisa e cobertura dentro desses conceitos.

Assim, foram apresentados os conceitos matemáticos utilizados nesse projeto e apresentou-se, assim, um guia que pode ser útil em outros projetos nessa área. A forma simples como este autor aproveitou-se dos recursos gráficos já desenvolvidos para ampliar significativamente os conceitos representados mostra que é possível, em Teoria das Categorias, projetar uma série de conceitos complexos para conceitos mais simples, levando o usuário de um simulador a lidar com relações e com funtores dentro de uma linha lógica bastante parecida.

Foi apresentado o projeto UML do CaTReS, mostrando de que maneira este evoluiu em relação ao CaTLeT, seu antecessor, e enfatizando aspectos que dizem respeito às características específicas do CaTReS - como, por exemplo, não haver necessidade de existir uma classe que trate problemas de associatividade no CaTReS. Avaliou-se os aspectos herdados do projeto do CaTLeT e, sobretudo, de que maneira houve necessidade de adaptações para que o CaTReS atingisse seus propósitos.

Por fim, como uma ferramenta que também atende a propósitos educacionais e que tem potencial para auxiliar em projetos ligados a ensino a distância de Teoria das Categorias, foi avaliada a integração do CaTReS com um SGEAD denominado HyperAutomaton, o qual faz uso de autômatos finitos com saída e páginas HTML para elaborar um curso a distância. Avaliando-se a capacidade de implementar um curso que promova a execução do CaTReS através de um navegador - posto que, sendo uma applet Java, pode ser executado através de uma marca HTML, desde que a JVM esteja instalada na máquina em questão - foi estudada a criação de cursos de Teoria das Categorias que possam fazer uso do CaTReS como ferramenta de apoio ao curso a distância.

O CaTReS não é uma ferramenta que aglutina em si todo o propósito de auxílio e apoio à pesquisa. Tal meta, conforme já foi dito, é de alcance impossível, pois Teoria

das Categorias pode ser vista e utilizada em combinação com uma série de estruturas matemáticas que tornam as possibilidades praticamente infinitas. Assim, não é difícil encontrar aperfeiçoamentos possíveis em sua acessibilidade, estruturas relevantes que pudessem ser implementadas para o apoio a pesquisas ou outras estruturas que pudessem estar sendo cobertas dentro dos conceitos categoriais. Contudo, trata-se de uma experiência inovadora, a qual, conforme os estudos apresentados nesta dissertação, não encontra paralelo nem em termos de resultados alcançados e tampouco em termos de propósitos para as quais tal ferramenta pode ser útil. A forma como os conceitos categoriais são implementados e projetados nessa ferramenta - incluindo as virtudes do paradigma orientado a objetos - permite a ampliação dos recuros dessa ferramenta com um mínimo impacto nos conceitos já implementados.

A seguir, é feita uma avaliação mais criteriosa a respeito dos aspectos levantados como importantes para um simulador categorial, levantando os avanços e as limitações dos resultados alcançados.

8.1 Acessibilidade do CaTReS

As avaliações referente a acessibilidade de simuladores categoriais vem sendo subdividida de maneira informal ao longo desta dissertação em três subaspectos: usabilidade; independência de plataforma; capacidade de acesso pela web. Tal situação ocorre em virtude de estes serem, na essência, os aspectos que facilitam ou dificultam o acesso de um usuário a um software ou a seus recursos. A avalição se acessibilidade do CaTReS também segue esse critério de avaliação.

Em termos de usabilidade, o CaTReS praticamente herda os recursos disponíveis no CaTLeT, não tendo sido foco desse projeto ampliar esses recursos. Foi tomada essa decisão em virtude do desejo de priorizar a ampliação das estruturas categoriais representáveis no CaTReS, uma vez que este autor julgou que os recursos gráficos desenvolvidos no CaTLeT satisfatórios para os propósitos estabelecidos no projeto. Entretanto, este autor reconhece que avanços no aspecto gráfico permitiriam ao usuário uma melhor visualização das construções funtoriais. Um ambiente de visualização em três dimensões, por exemplo, permitiria ao usuário visualizar as estruturas das categorias origem e destino e como estas estão sendo ligadas através de funtores.

O CaTReS também herda as características do CaTLeT que o tornam independente de plataforma. Qualquer sistema operacional que possua a JVM adequada instalada permitirá o uso do CaTReS. Nesse sentido, vale lembrar que o CaTReS não é passível de ser executado somente através de navegadores. Ele pode ser executado localmente na máquina, permitindo seu uso mesmo em situações onde o acesso a Internet não se encontra disponível.

O fato de ser uma applet Java permite que ele seja executado através de páginas web, através de uma marca HTML. Tal circunstância também foi herdada do CaTLeT. Contudo, convém pontuar que o CTDT, ferramenta desenvolvida pelo grupo canadense, está um passo a frente do CaTReS nesse sentido. Enquanto o CaTReS requer que o sistema operacional no qual o navegador que irá executá-lo está instalado possua uma JVM instalada, o CTDT não requer isso, em virtude da versão do Java no qual ele foi implementado. Convém ressaltar, contudo, que escolher se manter na versão 1.0 do Java para possuir tal vantagem comparativa significaria abrir mão de recursos disponíveis no Java em versões posteriores à 1.0. Foi avaliado neste projeto que o esforço que iria requerer retroceder os recursos implementados no CaTLeT para a versão 1.0 do Java - e a perda desses recursos

- era um preço caro demais a ser pago para agregar a virtude em questão no CaTReS.

8.2 Relevância das Estruturas Implementadas no CaTReS

O trabalho desenvolvido nesta dissertação tinha essencialmente dois propósitos: ampliar os objetivos do CaTLeT, tornando o CaTReS uma ferramenta de apoio ao aprendizado de conceitos intermediários de Teoria das categorias; tornar o CaTReS uma ferramenta útil no apoio a pesquisas que envolvam conceitos categoriais.

A implementação de conceitos funtoriais e de objetos e morfismos estruturados como conjuntos e relações, respectivamente, permitiu que tais objetivos fossem atendidos. Agora, um aprendiz em Teoria das Categorias que venha a lidar com objetos e morfismos do CaTReS terá uma percepção mais clara de que estes podem possuir uma estrutura interna - eventualmente complexa - e irá lidar com isso. O contato com conceitos funtoriais permitirá a esse aluno observar construções categoriais sendo mapeadas para outras construções categoriais. Tal situação, por exemplo, dentro do contexto de um curso de Teoria das Categorias, pode ser útil para demonstrar sua potencial utilidade como teoria que unifica estruturas matemáticas.

Da mesma maneira, o CaTReS possui recursos funtoriais que permitam ser utilizados como auxílio para a averiguação de premissas, ou mesmo para o suporte de experiências aplicadas com Teoria das Categorias. Sobretudo, o trabalho desempenhado na mecanização de Teoria das Categorias permite que o pesquisador analize como foi feito para representar estruturas no código fonte do CaTReS. Assim, não se vislumbra apenas a utilidade que o CaTReS possa representar como ferramenta de apoio à pesquisa, mas os procedimentos utilizados na sua construção podem render frutos em pesquisas para, por exemplo, representar determinadas estruturas categoriais como tipos. Também é notória a importância do conceito de relações para a Ciência da Computação, abrindo espaço para a representação de categorias, por exemplo, representando bancos de dados e redes de petri. A partir da implementação dos recursos relacionais, torna-se reduzido o esforço para implementar tal suporte, o que serviria para realizar pesquisas aplicadas em Banco de Dados utilizando Teoria das Categorias.

8.3 Cobertura do CaTReS

Ao apoiar-se em um trabalho acabado, onde o suporte à relações já era uma realidade, decidiu-se ampliar a cobertura ao conceito, de modo a torná-lo capaz de representar estruturas de grande importância para a Matemática e para a Ciência da Computação.

Ao derivar o conceito de funções parciais, por exemplo, o simulador tornou-se capaz de representar um conceito que é bastante utilizado na Ciência da Computação - por exemplo, no campo de pesquisa ligado a semântica formal.

A derivação do conceito de funções permitiu que, no próprio trabalho, este fosse usado como suporte ao conceito de funtores. Entretanto, inúmeros conceitos da matemática e formalismos da Ciência da Computação se apóiam em funções, o que torna tais estruturas auxiliares de grande interesse para os propósitos deste trabalho.

O suporte fornecido a relações funcionais, injetivas, totais e sobrejetivas permite um estudo mais adequado a respeito desses conceitos - inclusive na própria Teoria das Categorias, onde tais conceitos ainda não encontraram uma definição categorial.

8.4 Validação do CaTReS

O CaTReS foi validado junto a Mestrandos que estavam em início de aprendizado de Teoria das Categorias, nos meses de Março e Abril de 2006. Foi trabalhado com eles uma série de testes envolvendo os diversos simuladores categoriais citados nesta dissertação, onde o foco foi avaliar o CaTReS em comparação aos simuladores já existentes. Inicialmente, a avaliação envolvia 3 mestrandos, mas um deles afastou-se, restando 2 mestrandos para avaliarem a ferramenta. Trata-se de Bruno Carreio da Silva e de Jerônimo Backes.

O Bruno avaliou a ferramenta como fácil de operar, destacando o potencial de uso para o aprendizado - segundo ele, mais do que para a pesquisa. Segundo o Bruno, a ferramenta não requer familiaridade com conceitos de Ciência da Computação diversos daqueles que são objeto da Teoria das Categorias. Contudo, para ele, a visualização dos conceitos caegoriais poderia ser mais intuitiva - e ele viu esse defeito em todos os simuladores categoriais. Considerou também os conceitos categoriais abordados pela ferramenta pertinentes, assim como as estruturas matemáticas modeladas - conjuntos, relações e funtores -, são pertinentes. Ele detectou bugs no CaTReS e nos demais simuladores categoriais. Avaliou positivamente a ferramenta, mas destacou como ponto negativo da ferramenta o exceço de janelas que aparecem ao usuário, as quais, segundo ele, seriam não-amigáveis. Sugeriu maior investimento em interface gráfica.

O Jerônimo, tal qual o Bruno, avaliou de maneira positiva a ferramenta. Reconheceu intuitividade na ferramenta ao confirmar que ela não requeria familiaridade com informática ou com conceitos mais técnicos de Ciência da Computação - fora os que envolvem Teoria das Categorias. Contudo, a visualização dos conceitos categoriais, segundo o Jerônimo, poderia ser mais intuitiva - tanto no CaTReS quanto nas demais ferramentas, exceto o SimCat (a qual ele reconheceu como reproduzindo de maneira intuitiva conceitos categoriais). Também vê possibilidade de uso do CaTReS para o aprendizado e pesquisa, mas mais para o aprendizado do que para a pesquisa. Para o Jerônimo, os conceitos categoriais utilizados e as estruturas matemáticas representadas são bastante pertinentes, e atribui um bom conceito geral a ferramenta. O usuário ressaltou, contudo, uma série de reparos quanto a forma como o usuário interage com as ferramentas, sobretudo em relação a eventuais imperfeições funcionais.

8.5 Publicações

As pesquisas, estudos e resultados produzidos como decorrência desta dissertação de mestrado geraram duas publicações na *Tenth International Conference on Computer Aided Systems Theory* (Eurocast), realizada de 7 a 11 de fevereiro de 2005 na Espanha, em Las Palmas de Gran Canaria. Segundo informação obtida junto ao sitio da SBC em [QUA2001], referente a oitava edição do evento, o Eurocast é evento internacional avaliado pela Qualis como sendo de nível A.

A primeira publicação é um resumo estendido, o qual trata de simulação computacional de construções categoriais. Nele, são apresentados os resultados obtidos por Piaget junto à crianças a respeito do raciocínio categorial - assunto tratado neste trabalho. A partir desse ponto, é feita uma proposta sobre a utilização de simuladores categoriais como forma de tornar Teoria das Categorias menos abstrata e mais intuitiva - situação que está presente da gênese humana mas que é perdida ao longo do processo de aprendizagem.

O resumo estendido foi selecionado para ser apresentado no evento, no contexto do

workshop *Systems Theory and Simulation: formal approaches and applications*. Aproximadamente um mês após a apresentação, foi solicitado pela organização do Eurocast um artigo completo sobre o assunto.

Tal solicitação gerou a segunda publicação, a qual está em fase de preparação para ser incluída no volume do Eurocast 2005 no Lecture Notes in Computer Science. O artigo completo trata das questões de raciocínio categorial - abordadas resumidamente na outra publicação -, de como simuladores categoriais podem auxiliar a vencer barreiras, sobre os trabalhos desenvolvidos na área de simulação categorial e uma modelagem de estruturas categoriais para um simulador é apresentada.

Os resultados expostos nessas publicações estão diluídos ao longo deste trabalho. O resumo e o artigo completo estão disponíveis nos apêndices desta dissertação.

8.6 Trabalhos Futuros

Se é verdade que esse trabalho não se coloca como uma solução final para o suporte computacional a Teoria das Categorias para pesquisa, também é verdade que ele possui a virtude de abrir, a partir dele, uma série de possibilidades de trabalhos. Aqui, são mencionadas apenas algumas:

- Em Teoria das Categorias é bastante comum trabalhar-se com categorias infinitas, especialmente para o fim de representar estruturas matemáticas. Assim, seria interessante implementar o suporte a categorias infinitas (capazes de armazenar conjuntos de objetos e morfismos infinitos);
- Na Matemática e na Ciência da Computação, conceitos como relações, funções parciais e funções muitas vezes são utilizados em conjuntos infinitos, representando morfismos infinitos. Seria interessante tal suporte ser empregado em um simulador categorial (implementar conjuntos infinitos e relações infinitas entre conjuntos);
- O CaTReS é limitado a representar um número finito de categorias e funtores. Seria interessante ser capaz de representar infinitas categorias e funtores (por exemplo, todas as subcategorias finitas de FinRel);
- Como já existe o suporte a relações, seria interessante implementar trabalhos na área de banco de dados, empregando suporte a esse tipo de estrutura, ao cálculo relacional, à álgebra relacional e ao SQL;
- A ampliação dos conceitos categoriais representáveis, como exponenciação, categoria cartesiana fechada, categoria das setas e topos;
- A forma como o CaTReS implementa categorias de categorias permite ambicionar-se a implementação de categorias de categorias de categorias, e assim por diante, através de algumas mudanças na estrutura de dados e da lógica do programa (trata-se de um passo que pode ser dado com naturalidade no futuro);
- Representar outras estruturas nos objetos - como grafos, monóides, funtores - e nos morfismos - como homomorfismo de grafos, homomorfismo entre monóides, transformações naturais entre funtores;
- Avançar no ambiente gráfico do CaTReS, implementando recurso gráficos tridimensionais para a visualização de categorias e funtores entre categorias.

- Utilizar as estruturas categoriais representadas no CaTReS e modelar tipos categoriais que possam ser utilizados em linguagens de programação.

Os passos que requerem representações infinitas implicam tratamento de fórmulas e outros meios algébricos de representar estruturas, além de requerer avaliação postergada. Outros avanços são dados como naturais a partir do trabalho desenvolvido - como o desenvolvimento de mais níveis de categorias dentro de categorias. Entretanto, grande parte dos trabalhos que podem ser desenvolvidos a partir do CaTReS podem aproveitar estruturas já implementadas nas ampliações, o que permite não só uma coerência de operação como também evitar retrabalho.

REFERÊNCIAS

[ASP91] ASPERTI, A.; LONGO, G. **Categories, Types, and Structures**: an introduction to category theory for the working computer scientist. Cambridge: MIT Press, 1991.

[BAR95] BARR, M.; WELLS, C. **Category Theory for Computing Science**. 2nd ed. London: Prentice-Hall, 1995.

[BAR2002] BARR, M.; WELLS, C. **Toposes, Triples and Theories**. 2002. Disponível em: <http://www.cwru.edu/artsci/math/wells/pub/pdf/tttpdf.zip>. Acesso em: 17 maio 2005.

[BIR88] BIRD, R. S. **Introduction to Functional Programming**. New York: Prentice Hall, 1998.

[BRA2002] BRADBURY, J. et al. **Graphical database for category theory**. Canada: Mount Allison University, 1998-2002. Disponível em: <http://mathcs.mta.ca/research/rosebrugh/gdct/index.html>. Acesso em: 15 jun. 2005.

[BRA2005] BRASIL. Ministério da Educação. **Diretrizes curriculares de cursos da área de computação e informática**. Disponível em: <http://www.mec.gov.br/sesu/ftp/curdiretriz/computacao/co_diretriz.rtf>. Acesso em: 14 jul. 2005.

[CAR99] CARNEIRO, C. R. J. B. **Autômato não seqüencial identificado como suporte para classes em Náutilus**. 1999. 116p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[DAV92] DAVIE, A. J. T. **An Introduction to Functional Programming Systems using Haskell**. Cambridge: Cambridge University Press, 1992.

[DEI2001] DEITEL, H. M.; DEITEL, P. J. **Java**: como programar. 3.ed. Porto Alegre: Bookman, 2001.

[EIL45] EILEMBERG, S.; MACLANE, S. General theory of natural equivalences. **Transactions of the American Mathematical Society**, [S.l.], v.58, p.231-294, 1945.

[FLE95] FLEMING, M.; GUNTHER, R.; ROSEBRUGH, R. **A database of categories**. Canada: Mount Allison University, 1995. Disponível em: <http://www.mta.ca/∼rrosebru/dbc>. Acesso em: 25 maio 2005.

[IMS98] IMS Project. 1998. Disponível em: <http://www.imsproject.org>. Acesso em: 20 jun. 2005.

[JON2002] JONES, S. P. (Ed.). **Haskell 98 Language and Libraries**: the revised report. Cambridge: Cambridge University Press, 2002. Disponível em: <http://www.haskell.org/onlinereport>. Acesso em: 13 jul. 2005.

[LAR2000] LARMAN, C. **Utilizando UML e Padrões**: uma introdução à análise e ao projeto orientados a objetos. Porto Alegre: Bookman, 2000.

[MAC2000] MACHADO, J. H. A. P. **Hyper-automaton**: hipertextos e cursos na web usando autômatos finitos com saída. 2000. 148p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[MAC71] MAC LANE, S. **Categories for the Working Mathematician**. New York: Springer Verlag, 1971.

[MAU93] MAUNAY, M. **Introduction to Functional Programming**. Lisboa: Universidade de Lisboa, 1993.

[MCC2005] MCCREARY, C. **Visualizing Graphs with Java**. Auburn: Auburn University. Disponível em: <http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html>. Acesso em: 20 jul. 2005.

[MEC2001] MENDES, S. C. **Aplicações do paradigma funcional no ensino de matemática discreta**. 2001. 134p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[MEN97] MENEZES, P. F. B. **Reificação de objetos concorrentes**. 1997. 148f. Tese (Doutorado) - Instituto Superior Técnico, Universidade Técnica de Lisboa, Lisboa.

[MEN99] MENEZES, P. F. B. A categorial framework for concurrent, anticipatory systems. In: INTERNATIONAL CONFERENCE ON COMPUTING ANTICIPATORY SYSTEMS, 2., 1998, Liège, BE. **Computing anticipatory systems**. Woodburry: American Institute of Physics, 1999. p.185-199.

[MEN2001] MENEZES, P. F. B.; HAEUSLER, E. H. **Teoria das Categorias para Ciência da Computação**. Porto Alegre: Instituto de Informática da UFRGS, 2001.

[MEN2005] MENEZES, P. F. B. **Matemática Discreta para Computação e Informática**. 2.ed. Porto Alegre: Satra Luzzatto, 2005.

[MES90] MESEGUER, J.; MONTANARI, U. Petri nets are monoids. **Information and Computation**, Orlando, v.88, n.2, p.105-155, Oct.1990.

[MOG91] MOGGI, E.; Notions of computation and monads. **Information and Computation**, [S.l.], v.93, n.1, p.55-92, 1991.

[MOS2005] MOSCOW ML home page. Disponível em: <http://www.dina.kvl.dk/∼sestoft/mosml.html>. Acesso em: 12 abr. 2005.

[MOU2001] MOURÃO, T. **Simcat**: Um simulador gráfico de teoria das categorias para a internet. 50f. Projeto de Diplomação (Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[OWS97] OWSTON, R. D. The world wide web: A technology to enhance teaching and learning? **Educational Researcher**, Washington DC, v.26, n.2. p.27-33, Mar.1997.

[PFE2002] PFEIFF, F. V. **CaTLeT**: Ferramenta computacional de apoio ao ensino/aprendizado de teoria das categorias. 2002. 90p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

[PIA73] PIAGET, J. **Psicologia e Epistemologia**: por uma teoria do conhecimento. Rio de Janeiro: Forense, 1973.

[PIA92] PIAGET, J. **Morphisms and Categories**: comparing and transforming. Hillsdale, NJ: Lawrence Erlbaum ssociates, 1992.

[QUA2001] QUALIS.2001. Disponível em: <http://www.sbc.org.br/sbc/educacao/qualis/qualis2001/Anais_2001.xls>. Acesso em: 19 jul. 2005.

[ROS98] ROSEBRUGH, R.; BRADBURY, J. **Category Theory Database Tools**. Canada: Mount Allison University, 1998. Disponível em: <http://www.mta.ca/∼rrosebru/mathcs/javasource/index.htm>. Acesso em: 15 jun. 2005.

[RYD88] RYDEHEARD, D. E.; BURSTALL, R. M. **Computational Category Theory**. [S.l.]: Prentice Hall, 1988. Disponível em: <http://www.dina.kvl.dk/∼sestoft/mosml.html>. Acesso em: 17 jun. 2005.

[SCO2000] SCOTT, M. L. **Programming Language Pragmatics**. San Francisco: Morgan Kaufmann Publishers, 2000.

[SEB2000] SEBESTA, R. W. **Conceitos de Linguagens de Programação**. 4.ed. Porto Alegre: Bookman, 2000.

[VIE2003] VIEIRA, R. B. **Relações na ferramenta catlet**. 35p. Projeto de Diplomação (Ciência da Computação) - Instituto de Informática, UFRGS, Porto Alegre.

APÊNDICE A EUROCAST 2005: RESUMO ESTENDIDO

Segue o resumo estendido publicado no contexto do Eurocast 2005, posteriormente apresentado workshop *System Theory and Simulation: formal approaches and applications*.

a

b

APÊNDICE B EUROCAST 2005: ARTIGO COMPLETO

Segue o artigo completo, intitulado *Computational Simulation of Categorical Constructions*, publicado no contexto do Eurocast 2005, o qual encontra-se em fase de preparação para ser incluído em um volume do Lecture Notes in Computer Science.

a

b

a

b

a

b