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

CaTLeT:
Ferramenta Computacional de Apoio ao Ensino/Aprendizado de Teoria das Categorias

por

FÁBIO VICTOR PFEIFF

Dissertação submetida à avaliação,
como requisito parcial para a obtenção do grau de Mestre
em Ciência da Computação

Prof. Dr. Paulo Blauth Menezes
Orientador

Porto Alegre, abril de 2002.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Pfeiff, Fábio Victor

CaTLeT: ferramenta de apoio ao ensino / aprendizado de Teoria das Categorias / por Fábio Victor Pfeiff. – Porto Alegre: PPGC da UFRGS, 2002.

90 p.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR – RS, 2002. Orientador: Menezes, Paulo Blauth.

1. Ensino à distância. 2. Teoria das Categorias. 3. Análise e programação orientada a objetos. 4. Java. I. Menezes, Paulo Blauth. II. Título.

Sumário

Agradecimentos

A Deus, por ter me concedido a perseverança necessária ao exercício de minhas aptidões. A diferença entre os talentosos e os bem-sucedidos é a transpiração destes últimos na luta por seus objetivos.

A meu orientador, Paulo Fernando Blauth Menezes, por todo apoio prestado ao longo de três anos de convivência.

À professora Ana Maria de Alencar Price, por ter me proporcionado a experiência como bolsista de iniciação científica durante a graduação, ao professor Clésio Saraiva dos Santos, por ter sido meu orientador no trabalho de conclusão da graduação, e a ambos por terem me ajudado e incentivado a ingressar no Mestrado.

Ao Instituto de Informática da UFRGS e todos que o integram, pela infraestrutura e apoio providos desde o início de minha graduação até o mestrado que ora se encerra.

A meus pais, Geri e Regina, por terem acreditado que eu faria bom uso do melhor presente que puderam me dar, além da vida: formação e educação.

A minha esposa, Daniela, por ter me impulsionado em muitos momentos que esmoreci.

Lista de Figuras

Lista de Tabelas

Lista de Símbolos

WWW	World Wide Web
SGEAD	Sistemas de Gerenciamento para Ensino à Distância
HTML	Hypertext Markup Language
EAD	Ensino à Distância
IMS	Instructional Management Systems
PPGC	Programa de Pós-Graduação em Computação
TEIA	Técnicas de Ensino
UFRGS	Universidade Federal do Rio Grande do Sul
PUC/RJ	Pontifícia Universidade Católica do Rio de Janeiro
CTDT	Category Theory Database Tools
DBC	A Database of Categories
CGI	Common Gateway Interface
HTTP	Hypertext Transfer Protocol
$\in$	Pertence
$\notin$	Não-pertence
Ob_C	Objetos da categoria C
Mor_C	Morfismos da categoria C
C(A,B)	Coleção de morfismos da categoria C com origem no objeto A e destino no objeto B
$f: A \rightarrow B$	F é morfismo com origem em A e destino em B
$A \leq B$	A menor ou igual a B
GPF	General Protection Fault
VGJ	Visual Graphics for Java
STI	Sistema Tutor inteligente
Set	Categoria dos conjuntos
Poset	Categoria dos conjuntos parcialmente ordenados
Pfn	Categoria das funções parciais
Gr	Categoria dos grafos
RGr	Categoria dos grafos reflexivos
TFG	Trabalho Final de Graduação
JVM	Java Virtual Machine
CAI	Computer Aided Instruction
API	Application Programming Interface
OO	Orientação a Objetos
AOO	Análise Orientada a Objetos
DFD	Diagrama de Fluxo de Dados
JFC	Java Foundation Classes
JAI	Java Advanced Imaging
CASE	Compoter Aided Sofwtare Engineering

Resumo

Teoria das Categorias é uma ramificação da Matemática Pura relativamente recente, tendo sua base sido enunciada ao final da primeira metade do século XX.

Embora seja Teoria de grande expressividade, sua aplicação efetiva tem encontrado até o momento grandes obstáculos, todos decorrência natural da brevidade de sua História. A baixa oferta de bibliografia (e predominantemente em 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.

Consegue, a despeito destes obstáculos, arrebanhar admiradores em inúmeros centros de reconhecida excelência técnica e científica. Dentre todas as áreas do conhecimento, atrai em especial a atenção da Ciência da Computação, por características como *independência de implementação, dualidade, herança de resultados, possibilidade de comparação da expressividade de outros formalismos, forte embasamento em notação gráfica* e, sobretudo, pela *expressividade de suas construções* [MEN2001].

No Brasil, já conta com o reconhecimento de seu papel no futuro da Ciência da Computação por parte de instituições como SBC e MEC.

Os obstáculos aqui descritos, entretanto, ainda necessitam ser transpostos. O presente trabalho foi desenvolvido visando contribuir nesta tarefa.

O projeto consiste em uma iniciativa aplicada em Ciência da Computação, a qual visa oportunizar o franco acesso aos conceitos categoriais introdutórios: uma aplicação de computador que faça amplo uso de representação diagramática para apresentar a proposição de conceitos básicos do grupo de pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS.

A proposição e implementação de uma ferramenta, embora não constitua iniciativa inédita no mundo, até onde se sabe é a segunda experiência desta natureza. Ademais, vale destacar que os conceitos tratados, assim como os objetivos visados, são atendidos de forma única e exclusiva por esta aplicação.

Conjuntamente, vislumbra-se a aplicação desenvolvida desempenhando importante papel de agente catalisador na propagação da visão dos Grupos de Pesquisa em Teoria das Categorias da UFRGS e da PUC/RJ do que sejam os "conceitos categoriais introdutórios".

Palavras-chave: Ensino à Distância, Teoria das Categorias, Análise e Programação Orientada a Objetos, Java.

TITLE: "CaTLeT: COMPUTATIONAL SUPPORT FOR THE TEACHING/LEARNING OF CATEGORY THEORY"

Abstract

Category Theory is a relatively recent branch of Pure Mathematics, having had its basis enunciated at the end of the first half of the 20th century.

Regardless of its great expressiveness, actual application of the Theory has been prevented by several factors which naturally result from the brevity of its history. 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 the propagation of the Theory: the small number of educational courses that emphasize it.

In spite of these obstacles, Category Theory has gathered admirers in several centers of recognized technical and scientific excellence. Amongst all areas of human knowledge, it particularly draws the attention of Computer Science because of properties such as *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* [MEN2001].

In Brazil, the importance of the Theory to the future of Computer Science has already been recognized by institutions such as the Brazilian Computer Society (SBC) and the Ministry of Education and Culture (MEC).

However, the obstacles we have just described still need to be surpassed. The work presented here was developed with the intent of contributing to achieve that goal.

The project consists in an applied initiative in Computer Science, which aims to provide democratic access to introductory concepts of Category Theory: a computer application that makes extensive use of diagrammatic representation in order to present those basic concepts, as we view them at the Category Theory Research Group at UFRGS.

The proposal and implementation of such a tool, although not an entirely original idea, is only the second attempt we know of. Furthermore, it is worth noting that the concepts we cover, as well as the goals we have established, are uniquely satisfied by this application.

Additionally, we see the application fulfilling an important role as a catalytic agent in the propagation of the view of "introductory categorical concepts" as perceived by the Category Theory Research Groups at UFRGS and PUC-RJ.

Keywords: Distance Learning, Category Theory, Object Oriented Analysis and Programming, Java.

1 Introdução

A presente dissertação, submetida à avaliação como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação junto ao Programa de Pós-Graduação em Computação (PPGC) da Universidade Federal do Rio Grande do Sul, aborda a análise e implementação de uma aplicação de computador destinada a apoiar o processo de ensino/aprendizado de Teoria das Categorias.

O presente trabalho integra o Projeto TEIA - Técnicas de Ensino Interativas assistidas por Computador [WEB98], em realização no Instituto de Informática da Universidade Federal do Rio Grande do Sul. O objetivo desse projeto é o desenvolvimento de um canal de integração institucional entre graduação e pós-graduação. Visa a disponibilização na *Internet* das pesquisas dos grupos, os conteúdos das disciplinas de graduação e pós-graduação e os resultados de dissertações de mestrado, teses de doutorado e pesquisas científicas desenvolvidas em âmbito acadêmico.

1.1 Contexto do Trabalho

1.1.1 Teoria das Categorias

Teoria das Categorias [BAR90] [WAL91] constitui uma ramificação da Matemática Pura relativamente recente, tendo sido criada por S. Eilenberg e S. Mac Lane em 1945 [EIL45], como decorrência de seus trabalhos em topologia algébrica.

Apesar de ainda desfrutar do status de novidade até mesmo nos meios acadêmico e científico, quem com esta trava contato rapidamente identifica o potencial de aplicação da mesma em Ciência da Computação.

A importância que Teoria das Categorias potencialmente representa para a Ciência da Computação já é oficialmente reconhecida por instituições nacionais de relevância fundamental à sua propagação em nosso país:

- "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 linguagem Assembler" [MEC2001];
- a Sociedade Brasileira de Computação inclui esta Teoria em seu currículo de referência para os cursos da área [SBC97].

Suas características mais destacadas são sua expressividade e seu caráter de Teoria "auto-contida". O conhecimento necessário ao emprego deste ferramental, entretanto, não está se disseminando proporcionalmente à sua potencialidade. Sua difusão ainda esbarra em obstáculos característicos de teorias recentes:

- a oferta de material bibliográfico sobre o assunto ainda é bastante pequena. Além disso, as obras freqüentemente utilizam abordagem adequada a cursos em nível de pós-graduação ou com enfoque puramente matemático;
- os tópicos abordados variam muito entre as obras existentes (o que acarreta dificuldades a quem se propõe a estudá-la), evidenciando que ainda falta para Teoria das Categorias a maturidade verificada em outras áreas (e.g. Cálculo Diferencial e Integral);
- experiências didáticas de ensino em Teoria das Categorias são em número bastante pequeno. Em nível nacional, destaca-se pelo pioneirismo a disciplina de graduação com caráter obrigatório *Categorias Computacionais*, integrante do currículo do Bacharelado em Ciência da Computação da UFRGS. Recentemente, entretanto, disciplinas dedicadas ao estudo de Teoria das Categorias foram incorporadas a cursos de Computação de diversos outros centros nacionais (e.g. Rio de Janeiro, Recife, Belém do Pará e Pelotas).

É possível apontar, adicionalmente, uma circunstância particular que tem dificultado a experiência didática acima citada, apresentada em [COS99]: o desenvolvimento dos conceitos categoriais exige do estudante uma sólida base matemática e lógica. A ausência da fundamentação teórica que se faz necessária tem sido verificada em parcela expressiva dos freqüentadores dos cursos ministrados. A capacidade de abstração - que em Teoria das Categorias é bastante exigida - também manifesta-se estar aquém do necessário.

Como potencial grande utilizadora do ferramental provido por Teoria das Categorias, é conseqüência natural o surgimento, no âmbito da Ciência da Computação, de contribuições ao seu amadurecimento técnico e didático, bem como à sua difusão.

1.1.2 Teoria das Categorias no Instituto de Informática da UFRGS

Os últimos dez anos foram cenário da instalação e disseminação do estudo de Teoria das Categorias no Instituto de Informática da UFRGS. Alguns integrantes do Corpo Docente do Instituto destacam-se nacionalmente na condução de pesquisas, produção bibliográfica e implantação e apresentação de disciplinas dedicadas ao assunto, tanto no âmbito de graduação quanto em nível de pós-graduação *strictu sensu* (Mestrado e Doutorado). É dada, entretanto, a devida ênfase à aplicação potencial da mesma no âmbito da Ciência da Computação.

A disciplina *Categorias Computacionais*, ministrada à graduação do Bacharelado em Ciência da Computação da UFRGS a partir do primeiro semestre de 1997, é de caráter obrigatório e integra o elenco de conteúdos abordados no terceiro semestre deste Curso.

O grande diferencial desta experiência em relação a outras semelhantes em andamento no país, e até mesmo em nível internacional, consiste no caráter de obrigatoriedade conferido a esta. Ainda que não possibilite o pleno emprego dos conceitos desenvolvidos, serve ao propósito de ampliação da base de conhecedores. Outras instituições de ensino superior, por todo o mundo, já oferecem disciplinas sobre Teoria das Categorias a seu Corpo Discente de graduação. Poucas, entretanto, a definem como obrigatória (e.g. *Teoria das Categorias e Desenvolvimento Formal de Software* é disciplina experimental do curso de Ciência da Computação de Stanford; *Categorias*

para Informática I e *Categorias para Informática II* são disciplinas eletivas do curso de Informática da Universidade Técnica de Berlim). [MEN2001]

Até o segundo semestre de 1999, a disciplina integrava o sétimo semestre do curso. A estratégia então utilizada era colocá-la como ponto de chegada de uma seqüência de disciplinas de ênfase matemática e lógica, pretendendo que o aluno que a atingisse já tivesse desenvolvido certa gama de conteúdos e, assim, estivesse em condições de absorver os novos conceitos de forma mais natural e tranqüila. Tal seqüência iniciava-se em Matemática Discreta, e avançava, na ordem, para Lógica, Teoria da Computação, Linguagens Formais e Semântica Formal.

A organização inicialmente implementada não mostrou-se suficiente para garantir a uniformidade de aprendizado por parte dos freqüentadores da disciplina. Conceitos aos quais todos são desde muito cedo expostos em suas vidas, e que por tal encontram-se muito integrados à forma de pensar (e.g. Teoria dos Conjuntos), tornam a compreensão e o amadurecimento dos conceitos introduzidos por Teoria das Categorias uma tarefa, para muitos, de reeducação, imputando a quem ministra a disciplina o ônus de um acompanhamento extremanente próximo ao instruendo, por vezes individualizado.

Outro contraponto detectado é que, por estar situada curricularmente ao final do curso, pouco tempo e oportunidades restavam aos alunos para amadurecimento dos conceitos e efetiva aplicação.

Visando justamente tornar maior o tempo disponível para identificação de oportunidades à aplicação dos conceitos abordados pela disciplina, a partir do primeiro semestre de 2002 esta será deslocada para o quarto semestre do curso, persistindo como pré-requisito desta apenas o curso de Teoria da Computação.

A súmula da disciplina *Categorias Computacionais* ministrada aos cursos de graduação em Ciência da Computação da UFRGS enumera os seguintes tópicos:

- Categorias: conceituação básica e estudo das categorias mais conhecidas (Set, Pfn, Mon, etc);
- Morfismos especiais: mono, epi e isomorfismos;
- Objetos inicial, final e terminal;
- Produtos e co-produtos de aridade arbitrária;
- Igualadores e co-igualadores (ou equalizadores e co-equalizadores);
- Produtos Fibrados e Somas Amalgamadas;
- Limites e Colimites;
- Funtores.

A carga horária prevista para desenvolvimento de todo o conteúdo programático compreende 60 horas, distribuídas ao longo de um semestre letivo, com freqüência semanal de duas aulas, cada aula com duração de duas horas.

1.1.3 Ensino à Distância

Atualmente, uma das vertentes de pesquisa mais efervescentes é a que preocupa-se com Ensino à 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,EDU98] 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 de 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 extra-classe, etc. Enfim, tudo o que se faz necessário em um ambiente de ensino.

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

1.2 Caracterização do Contexto

É inaceitável que as dificuldades identificadas como responsáveis pela lenta difusão de Teoria das Categorias persistam em uma época na qual praticamente todas as áreas do conhecimento tiram proveito das possibilidades que a Ciência da Computação provê. Pelo menos o Grupo de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS enxerga que já existe cultura, conhecimento e tecnologia suficientes para reverter a situação atual:

- pode-se citar, em nível nacional, a proposta visando à unificação e sequenciação dos conceitos básicos para disciplinas de Teoria das Categorias apresentada em [MEN2001], elaborada em conjunto por um grupo de Matemáticos e Cientistas da Computação, integrantes do Corpo Docente da UFRGS e da PUC/RJ;
- em [MAC2000], é apresentado o SGEAD *Hyper-Automaton*, o qual constitui um sistema semi-automatizado para o suporte a cursos na *Web* (com base em uma arquitetura cliente/servidor e interface desenvolvida em *HTML*) através da aplicação de conceitos inerentes à Ciência da Computação, em especial da Teoria de Autômatos. A apresentação dos conceitos de Teoria das Categorias nesse ambiente já se encontra inclusive planejada, constituindo subconjunto considerável da súmula de [MEN2001];
- Ciência da Computação provê a todas as áreas do conhecimento humano a possibilidade de execução de simulações e representações de teorias e projetos em ambiente computacional. A utilização de tal advento representa, invariavelmente, maior nível de confiança no resultado final, associado a custo e risco menores. Apresenta-se como perfeitamente natural, portanto, o estudo da viabilidade de suporte computacional ao processo de ensino / aprendizado de Teoria das Categorias.

Como decorrência de todos os fatos aqui apresentados, uma aplicação de computador que abarque e desenvolva os conceitos introdutórios de Teoria das Categorias, e que possa ser integrada a um fluxo didático implementado em um SGEAD surge como altamente desejável e, mais do que isso, extremamente necessária.

1.3 Objetivos da Dissertação

De posse das necessidades identificadas no âmbito do processo de estudo de Teoria das Categorias, e tendo-se constatado uma possibilidade concreta de contribuir para a minimização das mesmas, uma série de atividades foi empreendida.

O trabalho cuja apresentação ora se inicia é resultado da soma de uma série de atividades, as quais podem ser agrupadas em duas etapas distintas.

Integraram a primeira etapa as seguintes atividades:

- fundamentou-se a viabilidade, e mais do que isso, o grau de adequação de uma ferramenta computacional de apoio ao processo de ensino / aprendizado de Teoria das Categorias no auxílio ao tratamento da complexidade inerente a seus conceitos;
- foi apurada e planificada a sistematização praticada pelo Departamento de Informática Teórica do Instituto de Informática da UFRGS no ensino de Teoria das Categorias;
- conduziu-se discussões entre os membros do Grupo de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS, cujo objetivo era apontar os fundamentos para uma aplicação de computador que contemplasse a sistematização apurada;
- procedeu-se à pesquisa de softwares pré-existentes cujo escopo fosse Teoria das Categorias;
- foram efetuadas análises sobre as duas aplicações de computador pré-existentes das quais se tem conhecimento até o presente (ambas integrantes da mesma experiência de pesquisa de um grupo canadense) confrontando-as com a sistemática pretendida para uma ferramenta dessa natureza.

As demais atividades desenvolvidas integraram a segunda etapa deste trabalho, todas relacionadas ao desenvolvimento da aplicação de computador que vem a ser a principal contribuição do mesmo. Sua concepção deveu-se integralmente à falta de êxito na busca e identificação de uma aplicação de computador pré-existente que atendesse às expectativas dos Grupos de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS e da PUC/RJ.

Tais atividades são aqui apresentadas segundo a seguinte organização:

- são descritas as etapas que integraram o ciclo de desenvolvimento de uma nova ferramenta computacional de apoio ao processo de ensino / aprendizado de Teoria das Categorias;
- apresenta-se o ferramental de projeto e as técnicas utilizadas na modelagem e desenvolvimento da aplicação;
- destaca-se a aderência conceitual da aplicação desenvolvida à sistemática de ensino proposta e efetivamente implementada pelo Departamento de Informática Teórica do Instituto de Informática da UFRGS;
- apresentam-se as funcionalidades incorporadas à aplicação decorrentes das necessidades identificadas, acrescidas das facilidades de uso adicionadas no intuito

de estabelecer rápida empatia dos usuários pela aplicação, destacando a tecnologia subjacente que tornou possível tais características;

- são abordadas as experiências que foram desenvolvidas paralelamente utilizando como tema a implementação de uma ferramenta computacional de apoio a cálculos categoriais - uma experiência didática desenvolvida no segundo semestre de 2000 com alunos regulares de graduação do Bacharelado em Ciência da Computação da UFRGS matriculados na disciplina *Categorias Computacionais*, e um Trabalho Final de Graduação desenvolvido por um formando do Bacharelado em Ciência da Computação da UFRGS no primeiro semestre de 2001;
- por fim, apresentam-se as formas como é disponibilizado o acesso à ferramenta implementada: como pacote de classes para execução local, em versão *standalone*, e através da Internet, numa composição de processos cooperantes do lado cliente (applet) e servidor (servlet), integrando uma versão eletrônica de [MEN2001] a ser editada no ambiente Hyper-Automaton [MAC2000].

A meta abrangente do presente trabalho é contribuir efetivamente com o processo de maturação e disseminação de Teoria das Categorias, deixando como legado uma forte base conceitual e técnica a ser utilizada em experiências futuras relacionadas a aplicações computacionais de suporte a Teoria das Categorias.

Entretanto, uma das principais contribuições específicas deste trabalho, a motivação em torno da qual todas as atividades desenvolvidas anteriormente citadas gravitaram, é uma aplicação de computador que atende aos requisitos conceituais identificados como necessários pelo Grupo de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS.

A ferramenta desenvolvida ambiciona tornar-se um agente catalisador na propagação, não apenas de Teoria das Categorias, mas conjuntamente da súmula de conteúdos proposta para o contato inicial de estudantes de qualquer área com a referida Teoria.

1.4 Organização da Dissertação

A apresentação da dissertação foi organizada conforme segue:

- O Capítulo 2 justifica e exemplifica as dificuldades enfrentadas por estudantes no estudo de Teoria das Categorias com base na complexidade associada a muitos de seus conceitos e cálculos. O capítulo encerra com uma fundamentação para a busca por apoio computacional ao processo de ensino/aprendizado/aplicação da teoria;
- O Capítulo 3 trata da pesquisa empreendida por ferramentas computacionais com enfoque no desenvolvimento de conceitos categoriais, reportando a descoberta de duas aplicações. Apresentam-se as avaliações técnicas das duas ferramentas sob os pontos de vista conceitual, funcional e de usabilidade. Deriva das análises que nenhuma das ferramentas atende às expectativas do grupo, o que dá início ao processo de discussão de nova aplicação, cuja especificação preliminar é apresentada. Por fim, é apresentada uma experiência didática executada com turma da disciplina de *Categorias Computacionais* no segundo semestre de 2000 nessa instituição, cujo tema foi exatamente o desenvolvimento de aplicação computacional que suportasse um subconjunto da súmula de conceitos categoriais básicos;

- O Capítulo 4 apresenta CaTLeT, a aplicação de computador desenvolvida segundo as idéias e princípios apresentados nos capítulos anteriores. Destaca-se a principal característica de Teoria das Categorias utilizada pela ferramenta. Apresentam-se as funcionalidades disponíveis na aplicação implementada. São relacionados e descritos os principais algoritmos implementados;
- O Capítulo 5 comenta a metodologia de desenvolvimento e o paradigma de programação utilizados no projeto e implementação da aplicação. Visando embasar tecnicamente a atenção dada a características como *Usabilidade*, *Portabilidade* e *Reusabilidade* tanto nas avaliações efetuadas sobre as ferramentas pré-existentes quanto no desenvolvimento da nova ferramenta, apresenta-se suscintamente um modelo hierárquico de qualidade em produtos de software. São apresentados os diagramas de *packages* e *classes* da aplicação. São enumeradas as ferramentas utilizadas na implementação;
- O Capítulo 6 estabelece a ligação entre simuladores como CaTLeT e o ambiente Hyper-Automaton, e entre estes e o projeto TEIA;
- O Capítulo 7 finaliza o trabalho. Dedica-se a apresentar as conclusões alcançadas a partir dos estudos efetuados e da construção da ferramenta, destacando as contribuições da dissertação. Adicionalmente, registra as possibilidades de trabalhos futuros que estendam e/ou utilizem os resultados atingidos por CaTLeT.

2 Contextualização das Necessidades

2.1 Caracterização da complexidade envolvida em cálculos categoriais

Em Teoria das Categorias, o termo utilizado para fazer referência a todos os objetos e todos os morfismos de uma categoria é *coleção*, e não *conjunto*. Tal diferenciação é imediatamente destacada em [BAC95] e [MEN2001], quando da apresentação do conceito de categoria.

A notação por compreensão utilizada na Teoria dos Conjuntos permite a definição de algo que, de fato, não é um conjunto, como segue:

$$S = \{ A \mid A \text{ é um conjunto e } A \notin A \}$$

o qual é interpretado como o conjunto de todos os conjuntos que não tem a si mesmos como elementos. De fato, se for suposto que S é um conjunto, tem-se que:

- se $S \in S$, então, por definição, $S \notin S$
- se $S \notin S$, então, por definição, $S \in S$

Essa contradição, denominada formalmente *Paradoxo de Russel*, mostra que não existe tal conjunto. Conseqüentemente, não existe o conjunto de todos os conjuntos e, portanto, nem toda coleção constitui um conjunto. Aqui surge a primeira diferença de amplitude entre Teoria dos Conjuntos e Teoria das Categorias quando confrontadas: a categoria de todas as categorias também é uma categoria, o que permite tratar categorias de categorias com o ferramental disponível na própria Teoria.

Obviamente, existem categorias definidas sobre conjuntos de objetos e morfismos. Tais categorias são referidas como *Pequenas*. Qualquer categoria que não possa ser enquadrada como *Pequena* é dita *Grande* [MEN2001]. Categorias, Pequenas ou Grandes, as quais para todo A, B $\in Ob_C$, C(A,B) (coleção de morfismos paralelos com origem em A e destino em B) constituem conjuntos, são ditas *Localmente Pequenas*, sendo os conjuntos da forma C(A,B) designados *hom-sets* [BAC95]. Qualquer categoria Pequena é, obviamente, Localmente Pequena.

A categoria ω contém como objetos os ordinais finitos (naturais) e existe um morfismo $f: A \rightarrow B$ sse $A \leq B$. Este exemplo tem a particularidade da cardinalidade máxima dos *hom-sets* ser a unidade. Designam-se as categorias nestas circunstâncias de *pré-ordens* [BAC95].

As categorias Pequenas são o tipo mais frequentemente utilizado na exemplificação dos conceitos categoriais introdutórios, pois oferecem ao estudante um domínio ao qual o mesmo, via de regra, está perfeitamente familiarizado.

A aplicação computacional de apoio ao aprendizado categorial aqui proposta e descrita faz uso justamente desta particularidade. O propósito da ferramenta, no presente estágio, é permitir representação e cálculos sobre categorias cujas coleções de objetos e morfismos são *conjuntos finitos*, sendo os componentes construídos um a um pelo usuário da aplicação. Categorias não-finitas cujo estudo faz-se extremamente relevante, e.g., Set, Poset, Pfn, Gr, RGr, não são cobertas de qualquer forma pela aplicação em sua versão corrente.

Como será demonstrado ao longo deste texto, o aprendizado e a aplicação dos conceitos básicos de Teoria das Categorias reservam armadilhas aos mais afoitos, mesmo quando manipulam-se categorias Pequenas (categorias em que Ob_C e Mor_C constituem conjuntos) e finitas.

Tome-se como exemplo o conceito mais elementar de Teoria das Categorias: a própria definição de categoria. As avaliações efetuadas sobre várias turmas da disciplina *Categorias Computacionais* (ministrada à graduação em Ciência da Computação desta instituição), aliadas às experiências pessoais dos membros do grupo, indicam que as armadilhas mais comuns quando da determinação se um diagrama de fato representa uma categoria relacionam-se à operação de composição e à exigência de que esta operação deve ser associativa.

O exemplo de teste da associatividade ilustrado nas Figuras 2.1 e 2.2 retrata o erro mais comum cometido frente à definição de categoria. Em seus primeiros passos à frente da definição, o estudante sempre concentra-se na identificação de morfismos identidade, quase sempre na verificação de composições, e só eventualmente lembra-se da necessidade de verificação da associatividade.

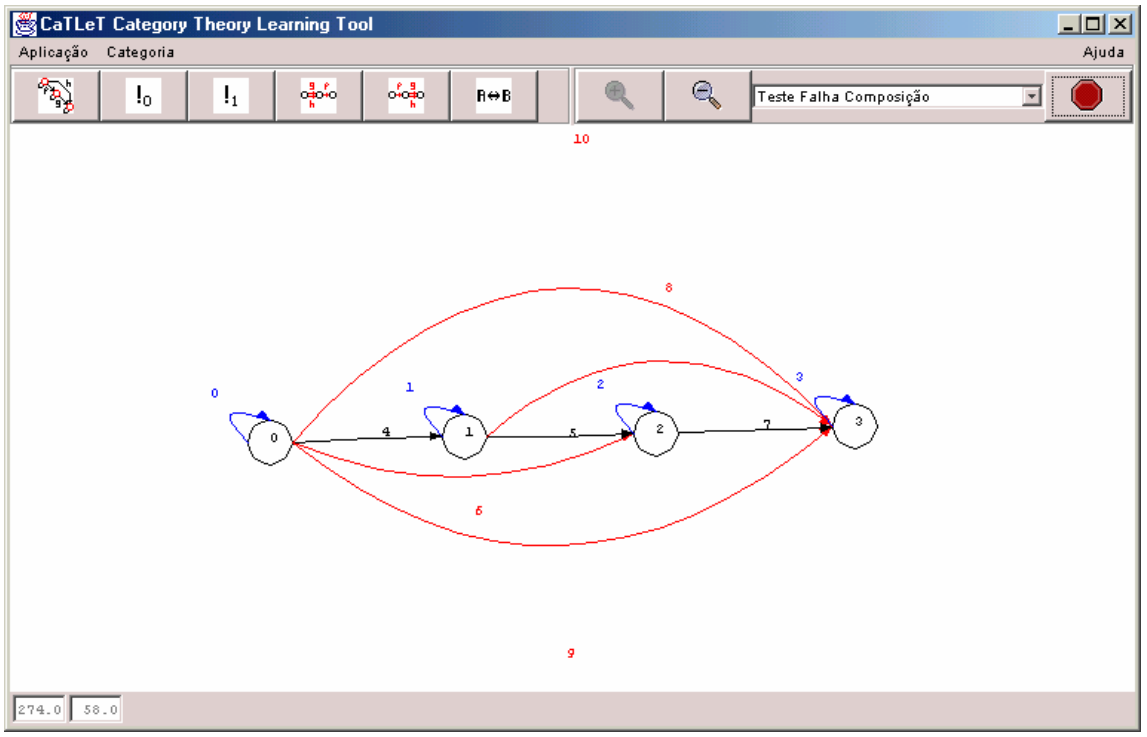


FIGURA 2.1 - Representação de diagrama não-associativo

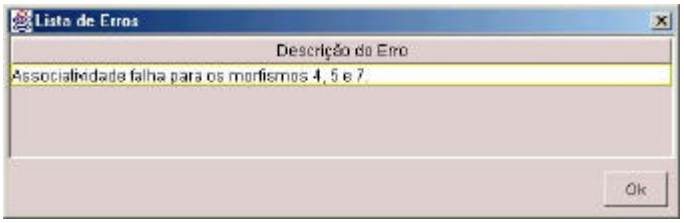


FIGURA 2.2 - janela de mensagens de erro para o diagrama representado na Figura 2.1

A partir das experiências didáticas do Grupo de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS, imagina-se que o aprendizado deste conceito pode ser acelerado se o estudante, após a apresentação da definição de categoria, for questionado diante de certos exercícios críticos, sendo a este apresentada, imediatamente após sua resposta, a eventual falha. Após uma falha perante a associatividade, o estudante certamente memorizará que a associatividade da composição também integra a definição.

Sem uma ferramenta que aponte imediatamente erros conceituais dessa natureza, a eventual detecção da incorrência em erros dessa espécie passa a depender de interação com o meio externo. O prolongamento da latência entre um erro e a detecção do mesmo pode determinar uma sedimentação de conhecimento defeituoso e implicar na utilização errônea deste como base para a compreensão de conceitos construtivamente dependentes.

2.2 Diagramas

Em Teoria das Categorias, um conceito de extrema relevância é o que versa sobre *diagrama*. É por meio desse ferramental que são expressas - graficamente - muitas equações e propriedades categoriais.

Teoria das Categorias possui uma nuance ainda pouco explorada pela bibliografia: diagramas, por definição, são *multicoleções* de objetos e morfismos [MEN2001]. A possibilidade de repetição dos componentes de uma categoria C em qualquer diagrama D elaborado sobre C, entretanto, não é destacada quando da apresentação formal do conceito por muitas publicações. Outra alternativa, uma alusão informal a esta característica, seria o desenvolvimento de exemplos com componentes replicados. Entretanto, os autores ainda não estão fazendo uso dessa possibilidade em bibliografias introdutórias.

A possibilidade de replicação é um problema que introduz complexidade adicional aos cálculos efetuados sobre diagramas. Esta necessidade só foi identificada tardiamente no desenvolvimento da aplicação. O impacto sobre os resultados atingidos pelo trabalho foi devidamente mensurado e será detalhado em seções futuras.

2.3 Ciência da Computação apoiando a disseminação de Teoria das Categorias

Até o presente momento este texto apresentou referências técnicas que atestam a importância de Teoria das Categorias para a Ciência da Computação. Fundamentou-se também a carência de material técnico-didático de apoio ao estudo desta Teoria, especialmente quando o idioma de trabalho é o Português. Por força da quase inexistência de recursos bibliográficos e, mais do que isso, de recursos humanos capacitados a aplicar e difundir esta Teoria, a Ciência da Computação e, de uma maneira ampla, toda a Indústria da Informática, tão carentes da aplicação de formalismos mais expressivos que os atualmente utilizados, continuam presos a ferramentais técnicos obsoletos, virtualmente incapazes de lidar com o volume e a complexidade dos problemas que hoje os desafiam.

Teoria das Categorias deve tornar-se, tão rápido quanto possível, parte do cerne de todas as áreas aplicadas da Matemática. Somente um contingente muito grande de cérebros dominando esta poderosa ferramenta poderá desbravar toda a sua potencialidade. É possível antever Teoria das Categorias atualmente como muitos vislumbraram o primeiro computador pessoal à época de seu surgimento - um "brinquedo" interessante. Poucos, entretanto (talvez seja possível enumerar entre estes apenas alguns escritores de ficção científica), foram capazes de imaginar o volume de

dados que estas máquinas são hoje capazes de processar e o trabalho que são capazes de executar.

Teoria das Categorias possui pouca bibliografia disponível e exige razoável base lógica e matemática e alto grau de abstração de quem a estuda. Seu poder de expressão, entretanto, justifica o esforço da comunidade científica em absorvê-la e espraiá-la. Formularam-se, a partir desse quadro, as seguintes perguntas:

- há alguma forma de oportunizar um acesso mais fácil ao estudo de Teoria das Categorias?
- existe alguma característica de Teoria das Categorias que possa ser explorada visando maior resultado no processo de ensino/aprendizado?

A Ciência da Computação pode efetivamente contribuir de forma decisiva na disseminação de Teoria das Categorias. O advento do WWW horizontalizou e democratizou o acesso à informação e aquisição de conhecimento. Este texto, e.g., foi amplamente suportado por versões eletrônicas de publicações que antes da Web ficariam restritas às bibliotecas.

Existe uma característica muito importante de Teoria das Categorias que pode e deve ser explorada ao máximo por Ciência da Computação: a possibilidade de representação diagramática de categorias.

Em Ciência da Computação, pode-se enumerar vários exemplos de representações gráficas que se fazem muito úteis quando do estudo de certas áreas, e.g., autômatos em Linguagens Formais [MEN98] e as representações das máquinas de Turing e Post [DIV99].

A representação gráfica de qualquer categoria finita ocorre na forma de um grafo orientado, correspondendo os nodos do grafo aos objetos da categoria e os arcos entre os nodos aos morfismos desta.

Teoria dos Grafos, inclusive, é parte integrante de qualquer currículo de Graduação em Ciência da Computação que se pretenda eficaz, o que reforça a afeição de nossa área de conhecimento por representações gráficas.

Adicionalmente, várias aplicações acadêmicas e comerciais podem ser enumeradas como bons exemplos de ferramentas que se utilizam de representações diagramáticas para atingir seus propósitos, demonstrando a adequação deste tipo de representação ao tratamento computacional. Pode-se citar, neste contexto, as ferramentas CASE.

Por todos os indicadores apontados, constata-se que a representação de grafos é um problema perfeitamente adequado à implementação em ambiente computacional.

Por conta dos fatos aqui enumerados, uma ferramenta computacional de apoio ao estudo de Teoria das Categorias apresenta-se possível e, mais importante, prenuncia-se como um meio democratizante de acesso à mesma.

3 Ferramentas existentes

Há cerca de dois anos, surgiu no Grupo de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS a idéia de prover suporte computacional ao processo de ensino/aprendizado da Teoria, tomando por premissa que uma ferramenta poderia, e mais do que isso, efetivamente deveria fazer tanto uso quanto possível das vantagens inerentes à representação diagramática de categorias e seus conceitos.

A partir do momento em que o grupo vislumbrou a possibilidade técnica de amparar computacionalmente o processo de ensino/aprendizado de Teoria das Categorias, imediatamente cogitou-se a existência de trabalhos científicos, já desenvolvidos ou em curso, de enfoque similar.

Como não se tinha conhecimento, até aquele momento, de nenhuma experiência desenvolvida com base em tal motivação, o passo seguinte foi empreender uma busca visando levantar o que porventura existisse.

A busca empreendida resultou na descoberta de duas aplicações.

O capítulo que se inicia dedica-se a:

- registrar a existência das duas aplicações encontradas;
- apresentar os resultados das análises efetuadas sobre as mesmas, executadas com o intuito de determinar o seu eventual aproveitamento como ferramentas de apoio à atividade didática de Teoria das Categorias empreendida no Instituto de Informática da UFRGS. Os pontos sob os quais as aplicações foram analisadas são quanto à aderência conceitual das mesmas à súmula utilizada nesse Instituto, às funcionalidades propostas e efetivamente disponíveis, e às características de usabilidade oferecidas.

3.1 Ferramenta 1 - Category Construction Program [FLE95]

Ferramenta desenvolvida em C Ansi, com interface orientada a caracter, desenvolvida pelo Grupo de Pesquisa em Teoria das Categorias da Universidade de Mount Allison (Canadá). Sua implementação foi realizada pelos pesquisadores Ryan Ghunter e Mike Fleming.

Apesar do nome que aparece na tela de apresentação da ferramenta, o grupo canadense utiliza a sigla DBC, acrônimo para "A Database of Categories", quando refere-se à mesma.

A ferramenta foi concebida sobre plataforma Unix. O fato de ter sido desenvolvida utilizando-se bibliotecas padrão de C, entretanto, tornou possível executá-la tanto em seu ambiente original quanto portá-la para o ambiente IBM/PC. Originalmente utilizou-se nessa empreitada o compilador GNU DJGPP. Posteriormente descobriu-se versão executável para esse ambiente disponível para download.

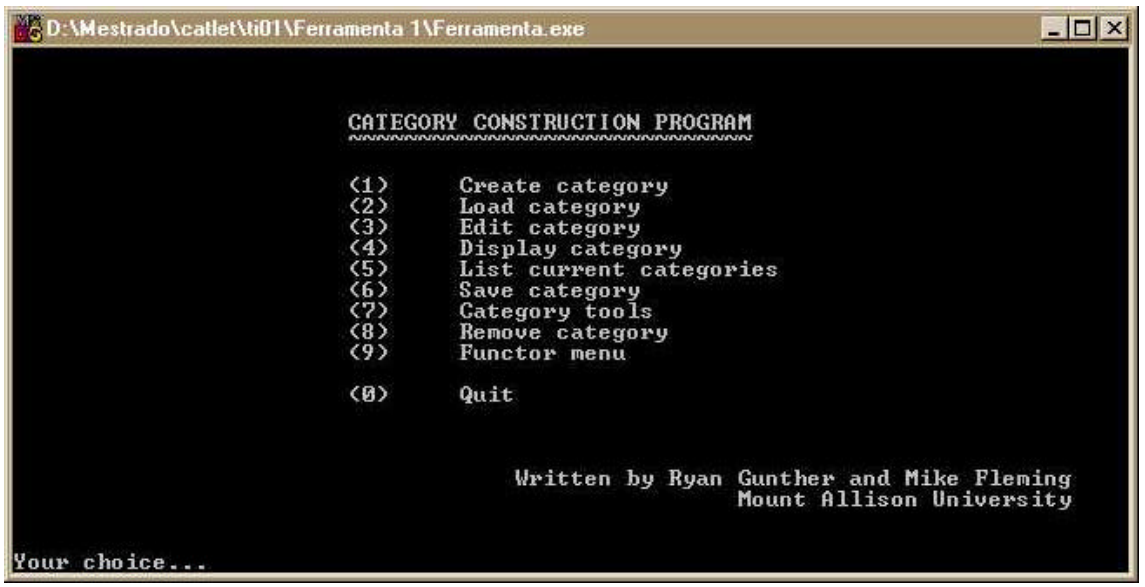


FIGURA 3.1 - Tela de apresentação da ferramenta 1 (A Database of Categories)

3.2 Ferramenta 2 - CTDT [ROS98]

CTDT é o acrônimo para Category Theory Database Tools.

Esta aplicação consiste em uma *applet* Java desenvolvida pelo grupo de Matemática e Ciência da Computação da Universidade de Mount Allison (Canadá), dentro do projeto de pesquisa do qual a ferramenta herdou seu nome.

O fato desta aplicação ter sido desenvolvida em Java torna-a completamente independente de plataforma. Adicionalmente, a versão de linguagem utilizada na implementação desta applet (1.0, a primeira *release* comercial) credencia qualquer equipamento dotado de browser que suporte applets a executá-la, uma vez que os primeiros browsers com suporte a Java embutiam JVMs desta versão.

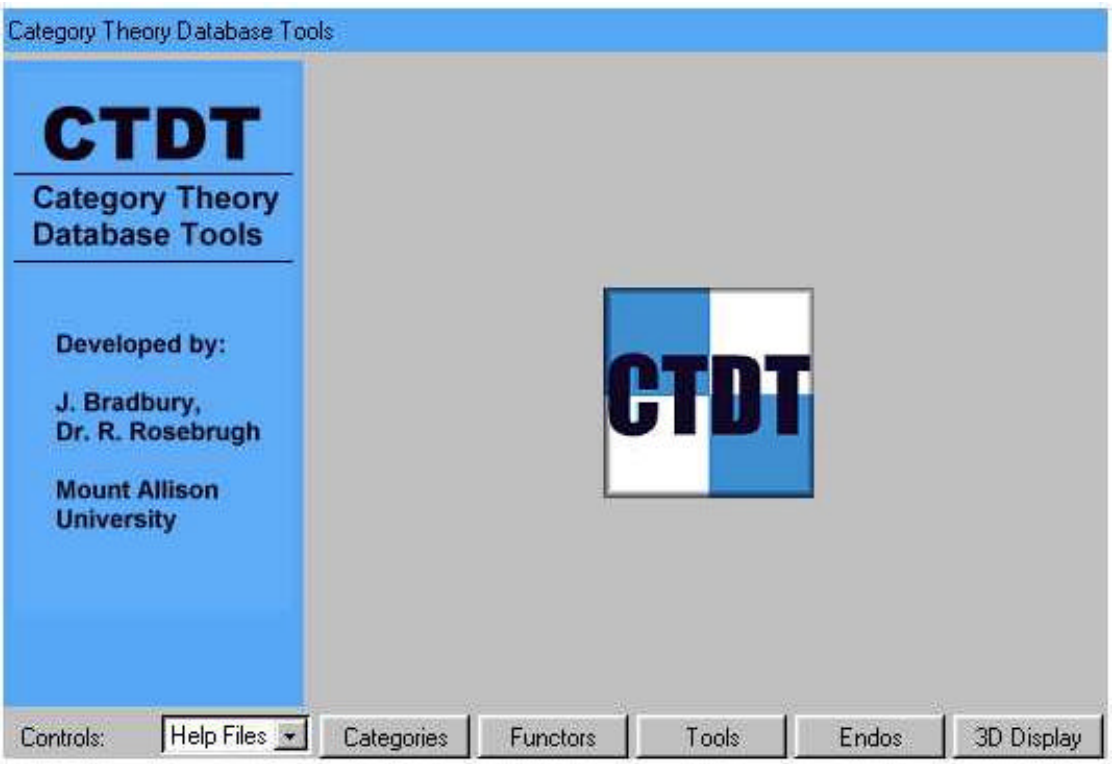


FIGURA 3.2 - Tela de apresentação da ferramenta 2 (CTDT)

3.3 Avaliação das Ferramentas

À primeira vista, as duas aplicações apresentavam dessemelhanças técnicas que não sugeriam qualquer relação entre as mesmas. Uma ferramenta apresenta interface orientada a caracter, tendo sido desenvolvida em C Ansi. A interface da outra aplicação é bem mais atraente. Desenvolvida em Java e disponibilizada na Internet na forma de applet, apresenta alguns elementos característicos de GUIs, possibilitando, adicionalmente, a visualização gráfica de diagramas.

Entretanto, bastou que se iniciasse o levantamento bibliográfico e a análise técnica das mesmas para apurar que um mesmo grupo de pesquisa - o Grupo de Pesquisa em Teoria das Categorias da Universidade de Mount Allison (Canadá) - coincidentemente era o responsável único pelo desenvolvimento de ambas.

A identificação da origem comum das duas ferramentas teve como conseqüência imediata o reconhecimento de que o conjunto de operações tratado, bem como a terminologia utilizada por ambas as aplicações, praticamente coincidia.

A forte similaridade identificada no conjunto de operações disponibilizado por cada uma das mesmas, entretanto, não descaracterizou a validade e a necessidade da análise técnica de cada aplicação em separado. Afinal, não é correto pressupor que a convicção de um grupo de pesquisa acerca de um conjunto de tópicos de estudo em Teoria das Categorias de qualquer forma impeça que se desenvolvam projetos seqüentes e construtivamente crescentes. Ambas as aplicações poderiam demonstrar valores diferenciados, principalmente sob o ponto de vista didático. O próprio trabalho que aqui é apresentado originou similar equivalente, totalmente desenvolvido em regime de cooperação (seção 5.8).

Era necessário, portanto, comprovar se havia seqüência cronológica entre as aplicações, ou se constituíam projetos desenvolvidos separadamente, e principalmente avaliar se os conceitos desenvolvidos e as funcionalidades providas por qualquer das mesmas eram condizentes com as necessidades identificadas.

O aprofundamento das análises revelou que o relacionamento entre as duas aplicações era muito mais estreito do que a origem comum das mesmas permitia supor. De fato, constatou-se que a CTDT (a ferramenta Java) era sucessora de Category Construction Program (a aplicação C). Enumeram-se a seguir os indicativos encontrados nas análises que apoiaram tal conclusão:

- a equivalência na modelagem das estruturas utilizadas na representação de categorias;
- a semelhança, a despeito das diferentes linguagens de programação utilizadas em cada uma das mesmas, entre muitos algoritmos de cálculo implementados nas duas ferramentas;
- a terminologia e a representação uniformes utilizadas nas ferramentas para a representação das estruturas por essas manipuladas.

Até o momento em que se teve o resultado das análises, o objetivo da busca por aplicações já existentes era avalizar o aproveitamento de alguma ferramenta diretamente no suporte ao processo de ensino de Teoria das Categorias no Instituto de Informática da UFRGS, incorporando-a como material de apoio didático.

As análises efetuadas sobre as duas aplicações, entretanto, revelaram pouca aderência das mesmas à súmula de conceitos que pautava as análises. Mas foi quando as

avaliações focalizaram a usabilidade e o rigor formal no tratamento dos conceitos que as maiores dissonâncias em relação ao comportamento pretendido, e que nelas era buscado, transpareceu.

Apesar da origem comum de ambas, quando vistas tanto sob a ótica da usabilidade quanto da funcionalidade, cada ferramenta apresentou um conjunto diferente de características, embora não totalmente disjunto. Certos comportamentos, tanto positivos quanto negativos, foram coincidentes. A investigação da origem das diferenças detectadas, entretanto, apontou para duas variantes básicas entre os projetos:

- a ferramenta 1 foi desenvolvida para interação do usuário totalmente orientada a caracter, enquanto a ferramenta 2 tirou algum proveito de componentes típicos de GUIs (e.g., botões e campos para entrada de dados), sendo que ainda integrou certa funcionalidade diagramática. Apesar disso, nem sempre a comparação pendeu favoravelmente à ferramenta 2;
- a ferramenta 1 executa localmente à máquina em que os arquivos componentes do software estão hospedados. A ferramenta 2, entretanto, é uma applet Java desenvolvida a partir de JDK1.0, e para executá-la pode-se tanto descarregá-la *on-line* a partir do site do grupo de pesquisa canadense, quanto obtê-la de forma integral em um pacote de classes (mas a aplicação continua executando na forma de applet, carregada por página HTML local, também integrante do pacote). Em ambas as formas de execução o modelo de segurança Java [GON98] aplicado a applets bloqueia algumas funcionalidades da aplicação. Funções executadas perfeitamente pela aplicação orientada a caracter não foram devidamente implementadas na versão Java/Internet.

Ambas as aplicações, contudo, apresentam deficiências. Algumas de solução bastante simples. Outras, entretanto, evidenciam falta de maior rigor conceitual. Visando destacar as virtudes buscadas, e que serão detalhadas quando da apresentação da nova aplicação implementada, serão avaliadas as vulnerabilidades conceituais e de usabilidade identificadas nas duas ferramentas analisadas.

3.3.1 Avaliação Conceitual

3.3.1.1 Ferramenta 1 - Category Construction Program

As opções de cálculo disponíveis diretamente a partir do menu de cálculos da aplicação, e que integram a súmula de conceitos de interesse, são:

- criação da categoria dual;
- objeto inicial: dado um determinado objeto pelo usuário, é verificado se este constitui objeto inicial da categoria.

Os demais cálculos disponíveis são:

- make confluent: verifica a existência de subcaminhos sobrepostos e relações que porventura não existam, visando tornar a categoria 'confluente'. A descrição aqui apresentada foi retirada de um comentário sobre a função diretamente do código-fonte da aplicação, pois de fato nada apurou-se quando se executou tal operação;
- equals: testa se o diagrama definido por dois caminhos paralelos (seqüências de arcos) comuta. Não consiste cada caminho individualmente, i.e., é possível fornecer caminhos em que, para dois arcos consecutivos de um caminho, a origem do segundo não seja o destino do primeiro (o que é passível de crítica em uma ferramenta de aprendizado);
- sum: cálculo do coproduto binário;
- right kan extension: cálculo que bibliografias de Teoria das Categorias não abordam como conceito introdutório;
- operações sobre funtores.

```
category
        teste
objects
        a, b, c.
arrows
        1:a->b, 2:b->c, 3:a->c.
relations
        21 = 3.
```

FIGURA 3.3 - Formato de apresentação dos componentes de uma categoria pela ferramenta 1

3.3.1.2 Ferramenta 2 – CTDT

O conjunto de operações encontrado na ferramenta 1 está propriamente contido no conjunto de operações da ferramenta 2. A relação entre os conjuntos de operações das duas ferramentas, entretanto, não é biunívoca. As seguintes operações adicionais são introduzidas na ferramenta 2:

- extensão kan esquerda;
- alteração de endomorfismos.

3.3.1.3 Avaliação Conceitual das Ferramentas

Pelo importante princípio da dualidade presente em Teoria das Categorias, a existência, nas duas aplicações, de uma operação que constrói a dual de qualquer categoria permite que se execute, adicionalmente aos cálculos diretamente disponíveis via menu:

- verificação de objetos terminais: verificação se um objeto na categoria dual constitui objeto inicial indicará que, na categoria original, esse é um objeto terminal;
- extensão kan esquerda: na applet esse cálculo já está disponível diretamente em menu. No entanto, pode-se calcular a extensão kan direita na categoria dual, transpondo-se o resultado para a categoria original.

A súmula de conceitos abordada por ambas as ferramentas diverge das características que o grupo da UFRGS identifica como necessidades. Tomando como base apenas os cálculos considerados relevantes, pode-se enumerar as seguintes carências:

- verificação se a construção informada constitui, de fato, categoria;
- verificação de morfismos especiais: mono, epi e isomorfismos;
- cálculo de produto e coproduto, no mínimo, de aridades 0, 1 e 2;
- determinação de cone e cocone;
- determinação de limite e colimite;
- determinação de produto fibrado e soma amalgamada.

Algumas divergências encontradas na terminologia utilizada também merecem destaque. Entre as diferenças de terminologia utilizadas nas ferramentas e a adotada pelo grupo de pesquisa da UFRGS, pode-se destacar:

- o conceito que é referido nas duas ferramentas pelo nome 'relação', pelo grupo da UFRGS é chamado 'composição';
- a operação categorial de coproduto ou soma não corresponde à operação implementada sob esta identificação nas ferramentas.

O desencontro de terminologias e conceitos entre a visão do grupo de pesquisa canadense e o grupo de pesquisa da UFRGS valida, com base em fatos, a afirmação anteriormente feita quando da explanação do estágio atual de Teoria das Categorias, em que afirmou-se ainda não existir nessa Teoria a maturidade verificada em outras áreas do conhecimento, no tocante à uniformidade terminológica e aos conceitos julgados básicos. De forma alguma é possível algum grupo postar-se como 'correto', imputando ao outro grupo necessariamente estar 'errado'. São apenas visões distintas de pessoas com um objetivo comum maior: difundir o estudo e a aplicação de Teoria das Categorias.

Entretanto, tais divergências por si só já constituem justificativa suficiente para a decisão do grupo da UFRGS pelo projeto e desenvolvimento de uma ferramenta própria, a despeito do que viesse a ser apurado nas análises de funcionalidade e usabilidade.

3.3.2 Funcionalidade

3.3.2.1 Ferramenta 1 - Category Construction Program

Todas as opções que integram a ferramenta 1 encontram-se em estado operacional.

As requisições para verificação de alguns conceitos tratados pela aplicação apresentam entradas de dados bastante restritas.

Visando embasar essa afirmação, a entrada de dados dos dois cálculos em que pode-se identificar essa característica são apresentados a seguir.

O primeiro cálculo é o objeto inicial. A Figura 3.4 destaca em negrito os parâmetros fornecidos pelo usuário.

```
category
        pfeiff
objects
        a, b, c.
arrows
        2:b->c, 1:a->c.
relations
        .

Enter object to be tested-->b

NOT INITIAL..no path to object a.
```

FIGURA 3.4 - Reprodução de execução do objeto inicial pela ferramenta 1

O segundo cálculo é o coproduto. A Figura 3.5 destaca em negrito os parâmetros fornecidos pelo usuário.

```
category
        pfeiff
objects
        a, b, c.
arrows
        2:b->c, 1:a->c.
relations
        .

Object of sum: c

Enter alpha: 1

Enter beta: 2

Enter maximum number of visits: 1

c is a sum.
```

FIGURA 3.5 - Reprodução de execução do cálculo do coproduto (Sum) pela ferramenta 1

Quando um estudante está iniciando o estudo de determinado conceito, é muito interessante que este interaja com uma ferramenta enfrentando semelhante nível de exigência. Com a superação das dificuldades inerentes ao conceito, entretanto, torna-se muito mais importante contar com a possibilidade de determinação automática de todos

os objetos iniciais a um só tempo, ou até mesmo com algum recurso para identificação gráfica de todos os objetos especiais.

Quanto ao cálculo da soma, acredita-se que o ideal em uma ferramenta seja possibilitar cálculos de qualquer aridade, não restringindo as possibilidades unicamente ao cálculo do coproduto (e, por dualidade, do produto) binário. Vale também para a determinação de produtos e coprodutos a colocação feita para o cálculo do objeto inicial: durante o aprendizado pode ser interessante exigir a discriminação de cada componente envolvido. Em produtos e coprodutos, isso representa solicitar, além dos objetos que estão sendo operados, a identificação dos morfismos projeção.

Em um momento posterior ao primeiro contato com o conceito de coproduto, porém, pode-se desonerar o estudante da tarefa de apontar cada morfismo injeção. A entrada de dados então passa a ser, unicamente, um multiconjunto de objetos dos quais deseja-se determinar o coproduto (se existir).

Sobre a resposta apresentada pela aplicação, quando da solicitação de verificação se determinada coleção de parâmetros constitui coproduto, vale destacar que a resposta retornada (e que pode ser conferida na Figura 3.5), 'c is a sum.', está, no mínimo, incompleta. Ainda que esteja subentendido que 'c', juntamente com os morfismos 'alpha' e 'beta' informados pelo usuário, constituam a tupla que de fato é um coproduto, não estão identificados em local nenhum os objetos dos quais esta tupla representa coproduto.

De forma a corresponder corretamente ao conceito que fundamenta este cálculo, e visando principalmente não habituar o estudante a pensar erroneamente que um objeto apenas constitui um coproduto (um erro bastante comum no início do estudo deste conceito) seria correto, e sem dúvida didaticamente mais eficiente, que a resposta apresentada na forma '< c , { 1 , 2 } > is a sum of < a , b >.'.

Depreende-se dos comentários aqui registrados que o Grupo de Pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS considera ideal a existência de, no mínimo, três níveis de verificação conceitual a ser requerido do usuário / estudante:

- no primeiro nível, equivalente ao encontrado nos cálculos de objeto inicial e coproduto das duas ferramentas analisadas, é exigida do usuário a discriminação de cada elemento constituinte da verificação;
- no segundo nível, o usuário é poupado do trabalho de prover certas entradas, e.g., discriminar os morfismos projeção / injeção no cálculo de produto / coproduto;
- no terceiro nível, certos cálculos são efetuados e apresentados ao usuário *on-line*. Uma representação categorial diagramática, e.g., poderia identificar automática e visualmente (com cores e/ou grafismos diferenciados) os objetos especiais (iniciais, terminais e zero).

A aplicação idealizada apresenta-se, no primeiro nível proposto, como questionadora, exigindo cada detalhe relacionado ao conceito em foco. Os demais níveis alçam a aplicação à condição de 'calculadora categorial', apoiando o usuário na elaboração de diagramas e execução de cálculos.

3.3.2.2 Ferramenta 2 - CTDT

Esta ferramenta, apesar de enfrentar algumas restrições associadas à sua implementação como *applet*, supera em funcionalidade sua predecessora.

Existe uma base de arquivos de categorias e funtores disponibilizados como exemplo no site que hospeda a ferramenta, os quais podem ser livremente carregados. Pode-se também criar e manipular categorias livremente em memória, editá-las e efetuar cálculos sobre as mesmas. Não é possível, entretanto, alterar de forma persistente o conteúdo desses arquivos, o que, antes de constituir problema, configura salvaguarda contra corrupção dessa base, seja esta de caráter acidental ou mal-intencionado. Tal característica não exigiu qualquer esforço adicional de seus desenvolvedores. A arquitetura de Java é a responsável única e direta por esta segurança.

A maior restrição enfrentada na utilização da mesma, entretanto, deriva da impossibilidade de o usuário armazenar persistentemente suas próprias categorias para uso futuro, quer seja a execução em modo on-line (remoto) ou local.

Todas as opções de cálculos e manipulações categoriais existentes na ferramenta 1 repetem-se na applet. A inversão da sentença, entretanto, não é verdadeira. Algumas funções existem apenas nesta aplicação.

Uma dessas opções é a possibilidade de verificação automática de todos os objetos da categoria perante o conceito de objeto inicial, denominada "Check all objects". A opção existente na ferramenta anterior, para verificação se um determinado objeto é inicial, continua presente.

Um ponto altamente positivo desta ferramenta é a possibilidade de visualização e edição de categorias em modo gráfico, como grafos. Ainda que limitado (e.g. não é possível especificar composições nesse modo), é de utilidade relevante.

O módulo da aplicação utilizado na representação gráfica de categorias é disponibilizado ao usuário a partir da carga de pelo menos uma categoria pela aplicação. Embora tenha recebido adaptações que o integram à ferramenta, esse módulo não é de autoria do grupo de pesquisa canadense.

O módulo, originalmente denominado VGJ, acrônimo inglês para *Visual Graphics for Java*, constitui um pacote de classes totalmente implementado em Java, também na versão 1.0 da linguagem, e destina-se à manipulação de quaisquer estruturas gráficas representáveis por meio de grafos.

Considerando-se a versão de Java utilizada em sua implementação, mostra-se altamente eficiente. A principal característica, sem dúvida, é o conjunto de alternativas de manipulação tridimensional oferecidas ao usuário. É possível rotacionar o gráfico sobre qualquer dos três eixos, alterar a escala de todo o diagrama e reposicioná-lo, tudo executado de forma muito eficiente.

O formato de arquivos utilizado por CTDT para persistência é o formato proprietário de VGJ, o que garante a integração destas.

3.3.3 Usabilidade

Qualquer interface com o usuário deve preocupar-se prioritariamente com a entrada e a saída de informações. Deve ser digna de muita atenção e esforço a tentativa de estabelecimento de um bom fluxo de comunicação entre a máquina e o usuário do sistema.

Não se pode em momento algum perder a noção de que, para estar apto a utilizar o software em seu aprendizado, o estudante precisa saber interagir com o mesmo. Portanto, uma das principais considerações de projeto deve ser a facilidade de manuseio da aplicação, o mais natural e intuitiva possível. Hix e Hartson afirmam: "Para os usuários, a interface é o próprio sistema".[HIX93]

A análise das duas aplicações canadenses sob a ótica da usabilidade, ou seja, da qualidade associada à interação com o usuário, aponta obstáculos ao uso de ambas.

Um grande limitador de usabilidade da ferramenta 1, de solução extremamente simples, reside no tamanho dos identificadores alfanuméricos aceitos pela ferramenta para os componentes de uma categoria (objetos, morfismos e composições), correspondente à unidade. A ferramenta aceita a digitação de mais caracteres, no entanto despreza todos à exceção do primeiro. A solução para tal problema consiste na aplicação de algum dos muitos formalismos disponíveis em Teoria das Linguagens Formais para reconhecimento de entradas válidas, e.g., expressões regulares.

A ferramenta 2, entretanto, corrige essa deficiência, possibilitando ao usuário fornecer identificadores com quantidade de caracteres maior que 1.

Ambas as ferramentas possibilitam a manipulação de múltiplas categorias simultaneamente. Nas duas aplicações, entretanto, a cada operação é necessário informar sobre qual categoria a mesma deve ser executada. Esta circunstância obriga o usuário a excessivas interações. Muito do trabalho executado na seleção de categorias poderia ser poupado se fosse adotado um conceito de 'categoria ativa': se houver pelo menos uma categoria em memória então, dentre todas as categorias carregadas pela aplicação, necessariamente uma é definida como a 'ativa' ou 'de trabalho', ou seja, a categoria carregada sobre o qual todas as operações selecionadas são diretamente executadas.

A ferramenta 2 não faz uso das referências numéricas seqüenciais que ela própria implementa. Apesar da presença da referência seqüencial, o usuário é solicitado a digitar o nome completo da categoria que deseja selecionar. Tem-se então, que para cada operação que o usuário deseje realizar sobre uma categoria:

- um clique é executado sobre a opção desejada. Essa ação apresenta a lista de categorias ativas;
- o usuário efetua um clique para selecionar a categoria de interesse;
- mais um clique ocorre sobre o botão que confirma a seleção efetuada;
- se a opção desejada pelo usuário é a listagem dos componentes da categoria, a próxima ação da ferramenta será apresentar os componentes. Se, entretanto, o objetivo é executar algum cálculo, abre-se o menu 'Category Tools'.

Contabiliza-se, nesse fluxo, dois cliques de mouse totalmente dispensáveis. Adicionalmente, com certeza ocorrerão acionamentos da tecla 'Tab' para navegação entre os componentes da tela ou, opcionalmente, deslocamentos do mouse entre todos os componentes clicados.

Um requisito que pode ser apontado na aplicação 1 como plenamente satisfeito, mas que estranhamente não foi reproduzido na applet, é o que se poderia chamar de 'coerência semântica' associada à seleção de componentes para manipulação dos mesmos.

Em ambas as ferramentas, sempre que uma operação é solicitada (ainda que seja somente a visualização dos componentes de uma categoria), é apresentada a lista de categorias ativas. As categorias são apresentadas uma por linha, cada uma sendo precedida à esquerda por um identificador seqüencial inteiro. Para selecionar uma dentre as categorias listadas, basta digitar o identificador seqüencial correspondente.

A referência seqüencial é utilizada em outros pontos das duas ferramentas (e.g. quando se deseja excluir um morfismo ou composição, indica-se o morfismo ou composição a excluir por sua referência seqüencial). O objetivo desta referência claramente é facilitar as interações do usuário. Adicionalmente, a uniformidade com que esta lista seqüencialmente identificada é usada em diferentes pontos abrevia o tempo dispendido pelo usuário no estudo e uso da interface.

O conforto proporcionado ao usuário pela sensação de coerência que a interface da ferramenta 1 transmite na seleção de componentes, entretanto, não foi propagado para a applet.

Na applet, embora algumas operações façam uso do identificador seqüencial para seleção de componentes, a seleção da categoria sobre a qual se vai operar ocorre pela digitação, por extenso, do nome da categoria.

A ferramenta 1, pelos recursos simples de interface de que fez uso, chegou ao final de nossa análise impressionando positivamente. Sem fazer uso de qualquer recurso de interface pré-concebido, e sem o grande apelo visual de sua sucessora, demonstrou ter um fluxo de comunicação homem-máquina simples e relativamente racional.

A usabilidade da applet, entretanto, exige que se proceda mais alguns comentários, visando justificar a avaliação que dela se fez.

A applet, pela qualidade dos recursos utilizados para desenvolvimento de sua interface, inspirou inicialmente maior simpatia do que a ferramenta 1. Os modernos recursos de que fez uso, aliados a uma forte identidade visual, entretanto, não traduziram-se em ganho de usabilidade. Em algumas situações (e.g. os identificadores seqüenciais anteriormente relatados) a aplicação 2 sequer manteve as qualidades observadas em sua predecessora.

Uma das características que mais conspira contra o usuário iniciante reside na forma como as opções são apresentadas na tela. Há cinco botões localizados na área inferior da applet, os quais são o ponto de partida para qualquer ação. Ao pressionar qualquer um destes, as opções associadas ao botão selecionado são disponibilizadas de forma dinâmica na área lateral esquerda da applet, também sob a forma de botões. A área restante da aplicação constitui a área de trabalho da mesma. Os menus sucessivos centralizados utilizados pela aplicação 1 apresentaram-se muito mais eficientes

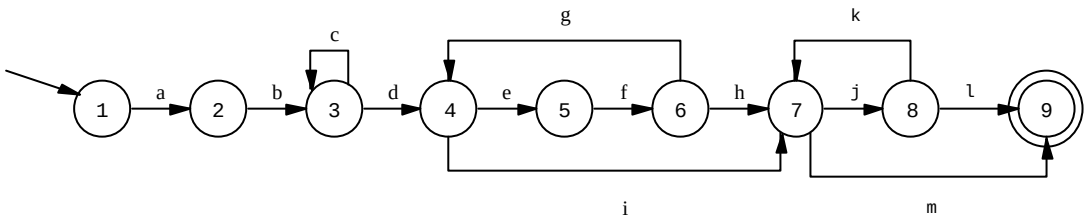
Como exemplo, pode-se citar a criação de uma categoria. Ao efetuar essa opção, são apresentados três componentes de interface na área de trabalho da aplicação:

- um campo multi-linha, cujo label, apresentado ao alto, identifica este como a ‘View Box’ da categoria que se está criando;

- um campo de texto simples (uma única linha), de label ‘Work Box’. É por meio desse componente que o usuário efetua suas entradas de dados, um a um (pela ordem, respectivamente, nome da categoria, objetos, morfismos e relações/composições);
- um botão de label ‘Done’.

Nos testes realizados, a cada entrada efetuada (nome da categoria, objeto, etc) acionava-se o botão ‘Done’, supondo-se que fosse este o evento gerador do registro da entrada, mas nada acontecia, permanecendo o prompt a solicitar como entrada o nome da categoria. Motivo: o evento responsável pelo registro é o acionamento da tecla ‘Enter’, como pode ser observado no fluxograma apresentado na Figura 3.6.

A ferramenta também abre espaço para alguns absurdos conceituais, facilmente evitáveis se utilizadas técnicas adequadas de modelagem dos eventos. Como exemplo tome-se a criação de uma categoria sem quaisquer objetos. Conseqüentemente, tal categoria não pode possuir morfismos. Mas a ferramenta, mesmo requisitando ordenadamente a entrada de objetos, morfismos e relações, questiona o usuário pela criação de morfismos, mesmo sem a criação de qualquer objeto, o que a deixa em um loop na criação desta. A única alternativa que resta ao usuário, então, é abortar a criação da mesma, acionando outra opção de menu.




```
Eventos
a - no menu "Categories", pressiona o botão "Create Category"
b - digita o nome da categoria na "Work Box" e tecla "Enter"
c - digita o nome de objeto e tecla "Enter"
d - pressiona "Done"
e - digita o nome de morfismo e tecla "Enter"
f - digita o objeto domínio do morfismo em criação e tecla "Enter"
    Se o objeto não existe, permanece no estado 5
g - digita o objeto codomínio do morfismo em criação e tecla "Enter"
    Se o objeto não existe, permanece no estado 6
h - pressiona "Done"
i - pressiona "Done"
j - digita morfismo composição e tecla "Enter"
k - digita os dois morfismos cuja composição é o anterior
    * Na ordem usual da operação
    * Sem qualquer caracter que a denote
l - pressiona "Done"
m - pressiona "Done"
```

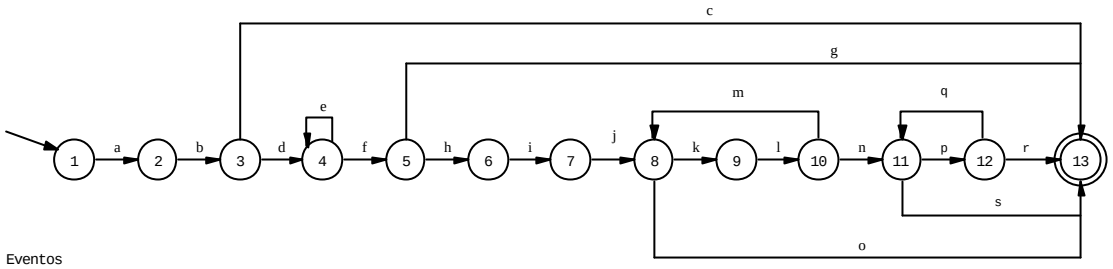
FIGURA 3.6 - Fluxo da ferramenta 2 (CTDT) na criação de uma nova categoria

A avaliação geral de usabilidade que se pode fazer da applet é que a mesma ficou muito aquém do que dela se esperava. Louvável sob muitos aspectos, surpreende que uma aplicação da natureza de CTDT, que visa estimular o aprendizado por meio da interação de seus usuários com um ambiente computacional, não tenha lançado mão de técnicas e recursos ricos e fartos de concepção e implementação de fluxos de informação e interfaces de usuário.

Um exemplo para ilustrar esta afirmação: se a entrada das informações componentes de uma categoria necessitava ser textual, era suficiente lançar mão da construção de um autômato de semântica semelhante ao da Figura 3.6, apenas tomando-

se o cuidado de introduzir estados que identificassem que ocorreu o atendimento de certas condições nos níveis anteriores.

Um autômato possível resultante das alterações necessárias é apresentado na Figura 3.7.



Eventos
a - no menu "Categories", pressiona o botão "Create Category"
b - digita o nome da categoria na "Work Box" e tecla "Enter"
c - pressiona "Done"
d - digita o nome de objeto e tecla "Enter"
e - digita o nome de objeto e tecla "Enter"
f - pressiona "Done"
g - pressiona "Done"
h - digita o nome de morfismo e tecla "Enter"
i - digita o objeto domínio do morfismo em criação e tecla "Enter"
    Se o objeto não existe, permanece no estado 6
j - digita o objeto codomínio do morfismo em criação e tecla "Enter"
    Se o objeto não existe, permanece no estado 7
k - digita o nome de morfismo e tecla "Enter"
l - digita o objeto domínio do morfismo em criação e tecla "Enter"
    Se o objeto não existe, permanece no estado 9
m - digita o objeto codomínio do morfismo em criação e tecla "Enter"
    Se o objeto não existe, permanece no estado 10
n - pressiona "Done"
o - pressiona "Done"
p - digita morfismo composição e tecla "Enter"
q - digita os dois morfismos cuja composição é o anterior
    * Na ordem usual da operação
    * Sem qualquer caracter que a denote
r - pressiona "Done"
s - pressiona "Done"

FIGURA 3.7 - Fluxo proposto para a ferramenta 2 (CTDT) na criação de uma nova categoria

O estado 3 passaria a ter a seguinte semântica: "aguardando a digitação de um objeto". Se o usuário pressionasse "Done" com o autômato posicionado neste estado, implicaria que a categoria criada é vazia (sem quaisquer objetos e, conseqüentemente sem morfismos). Conseqüentemente, não seria solicitado ao usuário a criação de morfismos (pois a existência de um morfismo só é possível com pelo menos um objeto na categoria), tampouco composições (composições são inviáveis até que exista pelo menos um par de morfismos em que o destino de um e a origem de outro sejam o mesmo objeto).

O autômato apresentado poderia sofrer alterações que introduzissem sofisticação ainda maior. Um exemplo: poder-se-ia impedir o software de solicitar composições ao usuário se não houver na categoria ao menos três morfismos 'f', 'g' e 'h' em condições de serem compostos.

A Tabela 3.1 resume os requisitos atendidos por cada uma das ferramentas analisadas, conforme todas as nuances segundo as quais as mesmas foram analisadas (conceitos, funcionalidade e usabilidade).

TABELA 3.1 – Súmula de conceitos X Funcionalidade das ferramentas

	DBC	**CTDT**
Múltiplas categorias	X	X
Edição gráfica	-	X
Valida construção	-	-
Suporte à correção de erros	-	-
Objetos especiais	X	X
Morfismos especiais	-	-
Criação da categoria dual	X	X
Produto	X	X
Produto de aridade maior que 2	-	-
Equalizador	-	-
Produto Fibrado	-	-
Cone	-	-
Limite	-	-
Funtor	X	X

4 Ferramenta Implementada

4.1 Apresentação

As análises efetuadas sobre as duas aplicações de computador categoriais encontradas, DBC e CTDT, sob os três aspectos relacionados, a saber, cobertura da súmula de conceitos categoriais, funcionalidade e usabilidade, permitiu verificar que nenhuma delas atendia satisfatoriamente tais requisitos.

A necessidade é uma aplicação de computador que suporte os conceitos listados por este trabalho como básicos e relevantes, aliado a rigor formal e facilidade de utilização.

A idéia que se fazia inicialmente de uma ferramenta desta natureza, entretanto, sofreu bastante transformação durante o processo de análise. Aliado à maturação natural da concepção inicial, a análise das duas aplicações permitiu identificar e incorporar várias outras características desejáveis.

A ferramenta implementada, batizada CaTLeT, acrônimo inglês para "Ferramenta para o Aprendizado de Teoria das Categorias" (Category Theory Learning Tool), busca materializar muitas das aspirações do Grupo de Pesquisa em Teoria das Categorias da UFRGS, tanto pela súmula de conteúdos que a mesma desenvolve, quanto pela funcionalidade e usabilidade oferecidas.

CaTLeT é uma aplicação de computador desenvolvida com o propósito de permitir a execução automática de cálculos categoriais sobre quaisquer dígrafos que graficamente representem formas possíveis de serem mapeadas para alguma categoria válida.

Implementada no decorrer dos anos de 2000 e 2001, foi totalmente desenvolvida utilizando-se a versão 1.3 da linguagem Java.

Poder-se-ia dizer que categorias e diagramas são representados por grafos. O mais correto é fundamentar suas representações gráficas em dígrafos: [LEO95]

“Um dígrafo D é definido como um par (V(D),A(D)), onde V(D) é uma coleção finita não-vazia de vértices e A(D) é uma multicoleção de pares ordenados de elementos de V(D).”

Por conveniência, e visando concordar com a bibliografia sobre o assunto, encerra-se aqui a discussão terminológica acerca do termo a utilizar no tratamento da metalinguagem gráfica categorial, sendo utilizado no restante do texto o termo genérico “grafo”.

Para os propósitos deste texto, pode-se utilizar, por simplicidade, uma descrição intuitiva de diagrama (a definição formal pode ser encontrada em [MEN2001]):

“Seja C uma categoria. Um diagrama D em C é uma multicoleção (coleção na qual podem existir elementos distinguidos) de objetos e de morfismos pertencentes a C tal que, para cada morfismo f pertencente à multicoleção de morfismos de D, $\partial_0(f)$ (objeto origem) e $\partial_1(f)$ (objeto destino) necessariamente pertencem à multicoleção de objetos deste.”

Note-se que diagrama e subcategoria são conceitos distintos. Diagrama especifica apenas os possíveis elementos constituintes do mesmo, bem como o comportamento das operações origem e destino. Subcategoria define, além disso, o

respeito às operações identidade e composição, devidamente restritos aos elementos tomados pela subcategoria. Conclui-se, portanto, que toda subcategoria S de uma categoria C constitui diagrama D sobre a mesma (desde que abstraídas as operações de identidade e composição, o que pode ser trivialmente obtido por um funtor esquecimento), não sendo o inverso necessariamente verdadeiro.

4.2 Especificação da ferramenta desejada

As premissas que compõem a especificação da aplicação idealizada são:

- o elenco de conceitos a ser desenvolvido pela aplicação deve corresponder à súmula utilizada na disciplina introdutória de Teoria das Categorias ministrada no Instituto de Informática da UFRGS (seção 1.1.2);
- as interações com o usuário devem fazer uso extensivo de uma importante característica de Teoria das Categorias, que é a sua notação diagramática;
- deve-se impedir construções categoriais mal-formadas;
- a aplicação deve permitir a manipulação de múltiplas categorias. Tal característica será de importância fundamental quando do tratamento de funtores;
- uma vez que a ferramenta manipulará várias categorias simultaneamente, será introduzido o conceito de 'categoria ativa'. A aplicação sempre executará qualquer computação sobre a categoria que estiver nessa condição. Ao usuário será disponibilizada a opção de, a qualquer tempo (diferentemente das aplicações analisadas), promover qualquer das categorias carregadas a ativa;
- a cada componente manipulado pela ferramenta poder-se-á atribuir identificadores alfanuméricos de tamanho arbitrário;
- a capacidade das estruturas de dados responsáveis pelo armazenamento dos componentes de categorias, bem como a quantidade de categorias possível simultaneamente em uma sessão de trabalho da aplicação, devem ser limitadas apenas pelo hardware subjacente;
- a aplicação deve ser independente de plataforma, ou seja, portável e executável em múltiplos ambientes computacionais;
- a interface da aplicação deve ser flexível e robusta, adaptando-se a diferentes resoluções de vídeo e sistemas operacionais;
- os componentes de interface responsáveis pela execução de todas as operações estarão permanentemente visíveis. Tais componentes serão itens de menu (convencional ou de contexto) ou botões em *toolbars*, todos integrantes do cotidiano de qualquer usuário de aplicações de microinformática. A disponibilidade de uso dos mesmos, porém, poderá estar contextualmente vinculada ao preenchimento de condições específicas a cada operação (e.g, a opção que identifica um morfismo como identidade exigirá que um único morfismo esteja selecionado, e que este seja um endomorfismo – morfismo com mesmo nodo origem e destino).

Durante a fase de análise das aplicações pré-existentes, identificou-se que estas permitiam a execução de cálculos sobre qualquer grafo. Entretanto, sob a ótica do Grupo de Pesquisa em Teoria das Categorias da UFRGS, o uso da definição de categoria na correta identificação se uma representação qualquer efetivamente constitui categoria é a primeira exigência a ser feita a qualquer estudioso desta Teoria. Deriva desta necessidade a incorporação, em tempo de projeto, da verificação formal automática efetivamente implementada em CaTLeT.

Atendendo a esta primeira necessidade, a interface da aplicação mantém o usuário permanentemente informado da situação atual do grafo corrente perante a definição categorial. Sempre que o grafo corrente apresenta-se mal-formado perante a definição de categoria, o estudante pode visualizar a relação de requisitos que o mesmo não preenche perante a definição.

Dentre as características relevantes da aplicação no que tange à usabilidade, vale destacar que, se o usuário assim desejar, as entradas de dados podem ser, em sua quase totalidade, pela edição de elementos gráficos.

As solicitações de cálculos são oferecidas na forma de opções da barra de menu, botões das *toolbars* ou opções do menu de contexto.

Muitos dos cálculos categoriais implementados em CaTLeT só são disponibilizados para grafos que atendem à definição básica de categoria, como forma de sequenciar o processo de estudo e aprendizado dos usuários da ferramenta.

Por definição conceitual, as duas únicas entidades gráficas possíveis em um grafo são objetos (ou nodos), representados por círculos, e morfismos (ou arcos), representados por linhas unindo dois círculos, sendo a orientação do morfismo devidamente destacada por seta em sua extremidade destino.

Em CaTLeT, objetos podem ser livremente adicionados à categoria corrente. Morfismos, por sua vez, necessariamente exigem que se indique seus objetos origem e destino. A conseqüência deste artifício implementado em CaTLeT é que qualquer representação gráfica construída pela ferramenta constitui grafo bem-formado.

4.3 Limitações que a ferramenta proposta apresenta

A ferramenta proposta visa atender a um conjunto de necessidades bastante específico. Certamente um usuário já familiarizado aos conceitos categoriais por ela desenvolvidos poderá considerar suas possibilidades bastante restritas.

Deve-se ter sempre em mente, entretanto, o principal objetivo desta: *apoio ao estudo* ***introdutório*** *de Teoria das Categorias*. O público-alvo a que a mesma visa atender prioritariamente, portanto, são estudantes em início de estudo da Teoria.

A principal limitação associada à aplicação reside no fato de que a mesma não apresenta qualquer possibilidade de especificação algébrica de categorias. Só é possível representar categorias discretas finitas, pois a única forma provida pela aplicação para construção de uma categoria consiste na especificação de cada componente pelo usuário.

Em termos práticos, significa que categorias cujas coleções de objetos e/ou morfismos são infinitas (independente de serem ou não conjuntos), como Set e Pfn, não podem ser trabalhadas pela aplicação em sua versão atual.

Outra restrição é que objetos e morfismos são os componentes atômicos da aplicação. A ferramenta não oferece suporte a objetos estruturados.

Certos cálculos categoriais somente têm sentido sobre diagramas [MEN2001]. Um diagrama é uma multicoleção de objetos e morfismos, ou seja, a coleção de objetos

e morfismos de um diagrama pode conter repetições de objetos e morfismos da categoria sobre a qual o mesmo está definido.

A ferramenta suporta a representação de componentes repetidos em diagramas. Entretanto, cabe destacar que a identificação desta necessidade ocorreu em momento bastante avançado do projeto, durante o desenvolvimento dos últimos algoritmos de cálculo, os responsáveis pela determinação de cones e limites, que necessariamente trabalham sobre diagramas. Foi avaliado, à época em que a decisão foi tomada, que a incorporação desta característica comprometeria o cronograma originalmente proposto.

4.4 Funcionalidades da Aplicação

A aplicação desenvolvida busca conduzir o instruendo/usuário através dos principais conceitos categoriais de forma didaticamente crescente.

A ferramenta possui as seguintes características:

- a execução de todos os cálculos categoriais disponíveis só é habilitada para uso quando o grafo corrente constitui categoria. Até que se atinja tal condição, apenas as opções necessárias à construção de categorias ficam disponíveis, ainda conforme as seleções do usuário;
- a manipulação de categorias e seus elementos pode ser executada de forma puramente diagramática;
- permite a manipulação de múltiplas categorias, totalmente isoladas uma das outras;
- o rigor conceitual adotado no desenvolvimento tornou possível traçar morfismos somente quando são informados dois objetos, um como origem e outro como destino;
- a ferramenta encarrega-se de habilitar contextualmente ao usuário a criação de morfismos identidade e composição somente para seleções de morfismos conceitualmente adequadas;
- pontos de inflexão para morfismos paralelos são automaticamente calculados, de modo que os morfismos não se sobreponham uns aos outros;
- o algoritmo desenvolvido e implementado para as operações de produto e coproduto permite o cálculo de produtos e coprodutos de qualquer aridade;
- a área de edição de diagramas adequa-se a qualquer resolução de monitor que esteja sendo utilizada, de forma a ocupar toda a área disponível. Pode-se a qualquer tempo, entretanto, customizar o tamanho da janela de edição para áreas menores. A ferramenta adequadamente redimensiona todos os seus componentes de interface (por fazer uso dos Layout Managers de Java) e reposiciona o diagrama correntemente visualizado (cálculos especificamente implementados para a aplicação);
- a ferramenta assume a identidade visual (Look&Feel) da plataforma em que está sendo executada, permitindo, adicionalmente, que se customize sua interface para a identidade padrão Java.
- interface com usuário totalmente desenvolvida utilizando componentes GUI do pacote Swing da Sun;
- adaptada à internacionalização/localização - toda informação textual utilizada na interface com o usuário está depositada em arquivo texto externo à aplicação [SUN2000a]. Quando executando como applet, entretanto, necessita cooperar com processo servidor (servlet) para obter os arquivos de localização de acordo com a

configuração regional da máquina cliente, uma vez que a execução de applets é rigidamente controlada;

- não utiliza formatos de arquivos proprietários para armazenamento persistente de categorias. Ao invés disso, utiliza a serialização de objetos provida por Java [HAL2000a], o que permite a qualquer aplicação que se venha a desenvolver a acessar os arquivos por esta gerados;
- a estrutura CollectionMorfismos, a qual agrupa todos os morfismos paralelos, é o equivalente computacional à definição de *hom-set*, sendo de grande utilidade nos algoritmos de cálculo;
- a aplicação é totalmente portável a qualquer plataforma para a qual exista máquina virtual Java implementada.

A Tabela 4.1 resume os requisitos atendidos pelas ferramentas pré-existentes sob as nuances segundo as quais as mesmas foram analisadas (conceitos, funcionalidade e usabilidade), confrontando as mesmas com CaTLeT.

TABELA 4.1 – Súmula de conceitos X Funcionalidade das ferramentas

	DBC	CTDT	CaTLeT
Múltiplas categorias	X	X	X
Edição gráfica	-	X	X
Valida construção	-	-	X
Suporte à correção de erros	-	-	X
Objetos especiais	X	X	X
Morfismos especiais	-	-	X
Criação da categoria dual	X	X	X
Produto	X	X	X
Produto de aridade maior que 2	-	-	X
Equalizador	-	-	X
Produto Fibrado	-	-	X
Cone	-	-	X
Limite	-	-	X
Funtor	X	X	X

4.5 Interface

A presente seção dedica-se à identificação dos componentes de interface da principal tela da aplicação, denominada DiagramBuilder. É nesta tela que se constróem e manipulam grafos, valida-se os mesmos contra a definição categorial e efetuam-se os cálculos sobre diagramas e categorias.

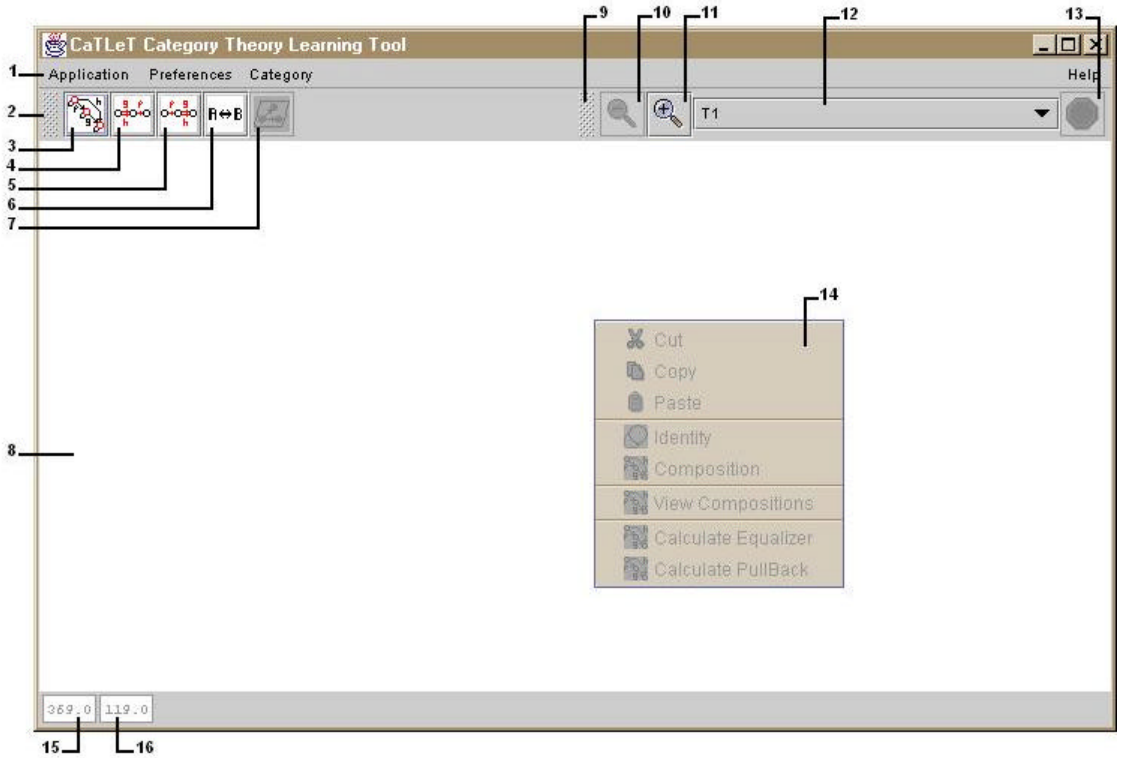


FIGURA 4.1 - Tela principal de CaTLeT (DiagramBuilder)

TABELA 4.2 - Enumeração dos componentes de interface de DiagramBuilder

Identificador	Descrição
1	Área de Menu
2	Toolbar 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	Toolbar de opções utilitárias
10	Botão "Zoom In"
11	Botão "Zoom Out"
12	Combo-box das categorias ativas, para seleção da categoria corrente
13	Botão de mensagens de erro. Só habilitado se diagrama corrente não representa categoria
14	Menu de Contexto
15	Coordenada vertical
16	Coordenada horizontal

1 Área de Menu

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

- Aplicação: agrupa predominantemente as operações de entrada e saída (criação, carga e salvamento de categorias). Adicionalmente, contém opção para alteração do Look & Feel da aplicação;
- Categoria: agrupa as opções oferecidas pela aplicação como auxílio à customização de sua operação (e.g. introdução automática ou não de morfismos-identidade), e chamadas a alguns dos cálculos categoriais implementados;
- Ajuda: área que reúne acesso à documentação de apoio on-line ao usuário.

2 Toolbar de cálculos categoriais

Procura oferecer ao usuário acesso rápido e prático a opções dos menus 'Aplicação' e 'Categoria' mais comumente utilizadas.

9 Área de Trabalho

É o painel em que se desenham os grafos.

10 Toolbar de opções utilitárias

As opções encontradas nesta toolbar não configuram atalho, pois não estão disponíveis em nenhum menu da aplicação. Agrupa os botões de Zoom (10 e 11), a combo-box utilizada para visualização das categorias carregadas em memória pela aplicação, bem como da atualmente ativa, e um botão cuja finalidade é indicar e o diagrama ativo atende ou não à definição de categoria. Estando este botão habilitado, o diagrama ativo não constitui categoria, e basta pressionar o mesmo para verificar detalhadamente cada falha do diagrama perante a definição categorial.

15 Menu de Contexto

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

Este menu contém três 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 na 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 das mesmas. Atendem, entretanto, a restrições específicas a 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 e último grupo, composto pelas opções de cálculo, respectivamente, de Equalizador e Produto Fibrado, habilitam 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.

4.6 Avaliação geral da complexidade dos algoritmos

A primeira característica a ser destacada para os algoritmos categoriais é que a complexidade de qualquer algoritmo tende a ser proporcional a uma potência do número de morfismos que o conceito para o qual se está desenvolvendo um algoritmo manipula simultaneamente.

Tome-se como exemplo o teste da associatividade na verificação se um grafo qualquer constitui categoria. É preciso emparceirar os morfismos três a três, para só após começar os testes relativos à associatividade, o que nos conduz a uma complexidade proporcional ao cubo da cardinalidade da coleção de morfismos da categoria.

Algumas soluções para estes problemas foram enunciadas e implementadas. A classe cujas instâncias agrupam os morfismos por paralelismo é uma alternativa. Quando se descarta uma coleção de morfismos de um teste de associatividade porque sua origem e/ou seu destino não atendem ao conceito, está-se descartando todos os morfismos contidos nesta coleção de uma só vez. Logo, a performance de um algoritmo de verificação de associatividade terá, no mínimo, a mesma performance de um algoritmo que efetua o mesmo teste morfismo a morfismo, sendo que será tão mais eficiente quantos forem os morfismos paralelos presentes no diagrama.

Existe *overhead* na administração desta estrutura intermediária. Entretanto, este dilui-se nas operações de manutenção de categorias - inclusão e exclusão de morfismos - que, no pior caso, tem complexidade linear e proporcional à quantidade de coleções de morfismos. Adicionalmente, a *collection* utilizada por CaTLeT possui performance insuperável em operações de inserção, exclusão e acesso aleatório a objetos (seções 5.6.2 e 5.6.6).

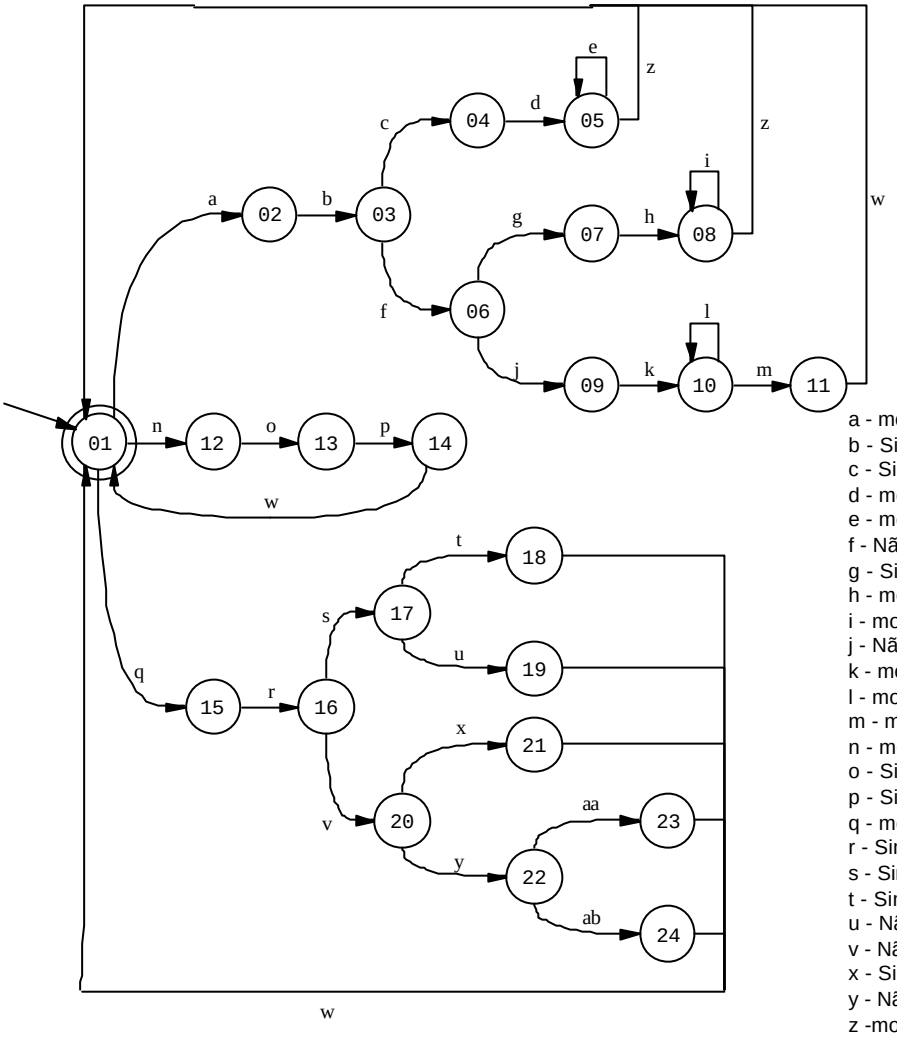
Outras operações que tiraram proveito das coleções de morfismos paralelos foram, e.g., o cálculo do produto e a determinação de cones e limites.

4.6.1 Modelo Formal de Entrada de Dados para Manutenção de Categorias

Grande atenção foi dada ao processo pelo qual o usuário constrói categorias em CaTLeT. A avaliação executada sobre as ferramentas canadenses apontou muitos problemas. O primeiro objetivo para o modelo de entrada de dados desenvolvido para CaTLeT consistia em não incorrer nos mesmos erros apurados para estas.

Autômatos são excelentes ferramentas para apoio à modelagem de fluxos de dados, fato comprovado pelos DFDs. Interfaces com características de validação dinâmica, caso de CaTLeT, podem ser vistas como fluxos de dados que provocam mudanças no estado da própria interface. Partindo dessa hipótese, pode-se pensar em desenvolver autômatos que modelem os estados e comportamentos de uma interface. Implementar autômatos programaticamente é tarefa de complexidade já dominada pela Ciência da Computação. A partir desse raciocínio foi elaborado o autômato a seguir, o qual gerou o modelo de comportamento da aplicação.

A solução mostrou-se eficaz não apenas no apoio à construção de grafos categoriais. Com pequenas modificações introduzidas nas ações associadas aos estados 18 e 22, o mesmo modelo pôde ser utilizado para a implementação do editor de diagramas sobre os quais efetuam-se cálculos de cones e limites.



a - mousePressed
b - Sim
c - Sim
d - mouseDragged
e - mouseDragged
f - Não
g - Sim
h - mouseDragged
i - mouseDragged
j - Não
k - mouseDragged
l - mouseDragged
m - mouseReleased
n - mouseMoved
o - Sim
p - Sim
q - mouseClicked
r - Sim
s - Sim
t - Sim
u - Não
v - Não
x - Sim
y - Não
z -mouseReleased
aa - Sim
ab - Não
w - movimento vazio

01 - Aguarda evento
02 - Há categoria ativa?
03 - O evento ocorreu sobre ponto de inflexão previamente selecionado de algum morfismo?
04 - Aguarda evento
05 - Redesenha o morfismo conforme a nova coordenada do ponto de inflexão 'arrastado'
06 - O evento ocorreu sobre objeto?
07 - Prepara todos os objetos que se encontram selecionados para movimentação. Aguarda evento
08 - Redesenha todos os objetos selecionados segundo o deslocamento aplicado ao mouse
09 - Inicia a abertura de retângulo de seleção nas coordenadas em que o mousePressed ocorreu. Aguarda evento
10 - Redesenha o retângulo de seleção, tomando vértice destino a nova coordenada do mouse
11 - Seleciona todos os objetos e morfismos contidos na área demarcada pelo retângulo de seleção
12 - Há categoria ativa?
13 - Há objeto origem definido (está sendo criado morfismo) ?
14 - Traça linha entre centro do objeto origem e ponto em que ocorreu o evento
15 - Há categoria ativa?
16 - O evento ocorreu sobre um objeto ?
17 - Há objeto origem definido (está sendo criado morfismo) ?
18 - Cria morfismo definindo objeto em que ocorreu o click como destino
19 - Inicia processo de criação de morfismo, definindo objeto em que ocorreu o click como origem
20 - O evento ocorreu sobre um morfismo ?
21 - Inverte seleção do morfismo (e.g. seleciona se não está selecionado)
22 - Há algum objeto ou morfismo selecionado?
23 - Desseleciona objetos e morfismos selecionados.
24 - Cria objeto, assumido as coordenandas em que ocorreu o click como o centro do mesmo

FIGURA 4.2 – Fluxo de manutenção de categorias implementado em CaTLeT

4.6.2 Verificação se um grafo representa categoria

Há um método da API Java 2D para manipulação de construções gráficas que é responsável pela tarefa de, a cada manutenção de um elemento em uma área de edição gráfica, um JPanel, redesenhar os elementos que o mesmo contém, o método paint().

Em CaTLeT este método sofreu *overriding*. As duas ações que ocorrem sobre o grafo corrente agregadas ao método original são:

- é acionado um método de instância na categoria corrente, cuja atribuição é calcular objetos especiais do grafo e verificar se o mesmo atende à definição de categoria. Em caso de não-conformidade com a definição, uma lista de erros, pertencente à própria instância, é construída, sendo o conteúdo dessa lista mostrado sob demanda do usuário (ver Tabela 4.2, identificador 13);
- o grafo é redesenhado conforme as cores parametrizadas pelo usuário para os objetos, morfismos e identificadores.

O algoritmo que determina se o grafo corrente é categoria consiste de quatro passos:

- supõe-se inicialmente que o grafo representa uma categoria válida. A lista de erros associada ao grafo é esvaziada;
- itera-se sobre os objetos do grafo, verificando se para cada objeto há um e somente um endomorfismo identificado como identidade. Cada objeto para o qual esta regra falha gera uma instância da classe *ErroIdentidade*, a qual é inserida na lista de erros do grafo;
- itera-se sobre os morfismos do grafo, combinando-os todos com todos aos pares:
  - para cada par de morfismos (f,g) que exige uma composição (destino do morfismo **f** coincide com origem do morfismo **g**), procura-se por um morfismo **h** que denote a composição de **f** com **g**. Caso esta busca falhe, é gerada uma instância da classe *ErroComposicao*, a qual é inserida na lista de erros do grafo;
  - a verificação da associatividade exige uma terceira iteração sobre a lista de morfismos, de modo a emparceirar com **f** e **g** um morfismo **h**. São buscadas as composições de **f** com **g** (e.g **fg**) e de **g** com **h** (e.g. **gh**). A seguir, verifica-se se **fg**o**h** = **f**o**gh**. Caso falhe, é gerada uma instância da classe *ErroAssociatividade*, a qual é inserida na lista de erros do grafo;
- o retorno do método é um valor lógico, obtido pelo teste se a lista de erros está vazia, cuja semântica é “grafo sem erros perante definição categorial”.

4.6.3 Algoritmo para cálculo do produto categorial

O algoritmo desenvolvido para determinação do produto de quaisquer objetos de uma categoria (e, por dualidade, do coproduto) é de expressiva complexidade e, conseqüentemente, tendo sido um dos que mais tempo consumiu em sua implementação.

A grande dificuldade associada ao suporte computacional da operação de cálculo do produto categorial, e que até então não havia sido enfrentada por qualquer outra ferramenta, referia-se ao tratamento de produtos de aridade arbitrária (ou seja, qualquer quantidade de objetos envolvidos) e à possibilidade de replicação de objetos.

As ferramentas canadenses determinam apenas produtos binários. As ferramentas fomentadas na disciplina de Categorias Computacionais (seção 4.7) desenvolveram versões que suportam produtos de aridade zero, um e dois, da mesma forma que SimCat (seção 4.8). Nenhum programa de que se tem notícia, entretanto, ousou ou preocupou-se em implementar uma operação com aridade superior a dois.

CaTLeT constitui, portanto, a ferramenta pioneira na determinação de produtos de grau superior ao binário, conferindo ao usuário da aplicação a possibilidade de cálculo de produtos de qualquer aridade.

A principal idéia que embasa o algoritmo desenvolvido para cálculo do produto categorial de aridade arbitrária implementado em CaTLeT decorre da generalização do teorema da unicidade do produto. [MEN2001]

O objeto resultante do produto finito de quaisquer três objetos A, B, C é A x B x C, o qual é isomorfo a (A x B) x C e A x (B x C), de onde conclui-se que o produto categorial é associativo, a menos de isomorfismo.

Uma conseqüência muito importante desta conclusão é o fato de que o produto finito de três ou mais objetos pode ser calculado a partir de produtos binários, operados em qualquer ordem.

O fato de o produto categorial ser uma operação associativa suscitou a possibilidade do uso de recursividade para seu tratamento em ambiente computacional.

Quando decidiu-se implementar esta operação com aridade arbitrária, o primeiro desafio a ser enfrentado foi especificar e implementar uma interface com o usuário para entrada de dados que fosse simultaneamente confortável e ágil no momento de se informar os objetos sobre os quais se deseja calcular o produto.

FIGURA 4.3 - Janela utilizada para indicação dos objetos sobre os quais se deve calcular o produto

A solução adotada, que pode ser vista na Figura 4.3, faz uso de uma JTable com três colunas. Cada linha representa um objeto da categoria. A coluna postada inicialmente à esquerda apresenta um campo do tipo checkbox, utilizada para que se indique se o objeto fará parte do produto a ser calculado. A coluna situada inicialmente

ao centro identifica o objeto. E a coluna postada inicialmente à direita permite ao usuário informar a cardinalidade com que o objeto contribui para o cálculo do produto. A flexibilidade da JTable permite ao usuário reposicionar as colunas entre si, bem como redimensioná-las. A janela que contém a JTable também pode ser livremente redimensionada.

Com esta solução, o usuário pode visualizar simultaneamente múltiplos objetos (e, se o grafo não for extenso, até mesmo todos os objetos), marcando os check boxes correspondentes aos objetos que deseja operar. A aridade da operação é determinada diretamente a partir da quantidade de objetos selecionados e da cardinalidade informada para cada um dos mesmos. Cada objeto selecionado incrementa em uma unidade a aridade da operação.

Fazendo uso do teorema do produto binário, desenvolveu-se um método recursivo, cuja entrada de dados é a lista ordenada de objetos a serem operados. A resposta retornada, se houver produto para a coleção de objetos informada como entrada, é uma instância da classe Produto, composta por um objeto e uma coleção de morfismos projeção cuja cardinalidade é igual à quantidade de objetos passados ao método de cálculo do produto.

O algoritmo efetua três cálculos distintos. A escolha por um dos cálculos está associada à aridade do produto a calcular.

Se a aridade do produto é zero (ou seja, não foi selecionado qualquer objeto), utiliza-se o para o cômputo do resultado o mesmo método que determina o objeto terminal do diagrama, sendo este, caso exista, a resposta retornada. A tupla de morfismos projeção para zero objetos é necessariamente vazia.

Se a aridade do produto é unitária, a determinação do produto é trivial, sendo retornado como resposta o próprio objeto selecionado juntamente com o seu morfismo identidade na condição de único elemento da tupla de morfismos projeção. A única preocupação deste cálculo é com a verificação se o objeto possui morfismo identidade. A não-existência deste morfismo identidade implica a não-existência de produto para o objeto selecionado.

Se, entretanto, a aridade do produto categorial a ser calculado é igual ou superior a dois, então o método recursivo desenvolvido por CaTLeT é quem toma conta das ações. A técnica de recursão utilizada no desenvolvimento do algoritmo é *top-down*.

A base do algoritmo (o passo para encerramento da recursão) é que a entrada de dados, ou seja, a coleção de objetos sobre os quais será calculado o produto no passo corrente, apresente cardinalidade igual a dois. A determinação de produtos categoriais de ordem igual ou superior a dois será, por conseqüência, uma sucessão de um ou mais cálculos de produtos categoriais binários.

De modo a ilustrar a explanação, suponha-se uma lista de objetos informados pelo usuário de comprimento quatro, comprimento suficiente para que ocorram duas invocações recursivas. Seja a lista da forma [A,B,C,D].

```
calculaProdutoBinario(LinkedList objectsProduct) : Produto
```

FIGURA 4.4 - Assinatura do método de cálculo de produto categorial de ordem igual ou superior a 2

A classe **Produto** tem papel fundamental no algoritmo. Como pode ser observado na Figura 4.4, o resultado de cada iteração recursiva, bem como o resultado final retornado pelo algoritmo, constituem instâncias desta classe.

Um importante ponto a observar é que a tupla de morfismos projeção da classe **Produto** é definida a partir de uma LinkedList, a qual é uma estrutura que garante a ordem dos elementos conforme sua inclusão. Adicionalmente, pode-se perceber que também a coleção de objetos passada como argumento de entrada para o método recursivo está definida sobre a mesma estrutura. De fato, nem sempre se obteve o resultado esperado quando inicialmente utilizou-se uma estrutura sem ordenação.

Não se deve esquecer que o produto é composto por um objeto e uma tupla de morfismos (morfismos em quantidade igual à aridade da operação). Logo, quando os objetos são operados, os morfismos também devem ser compostos.

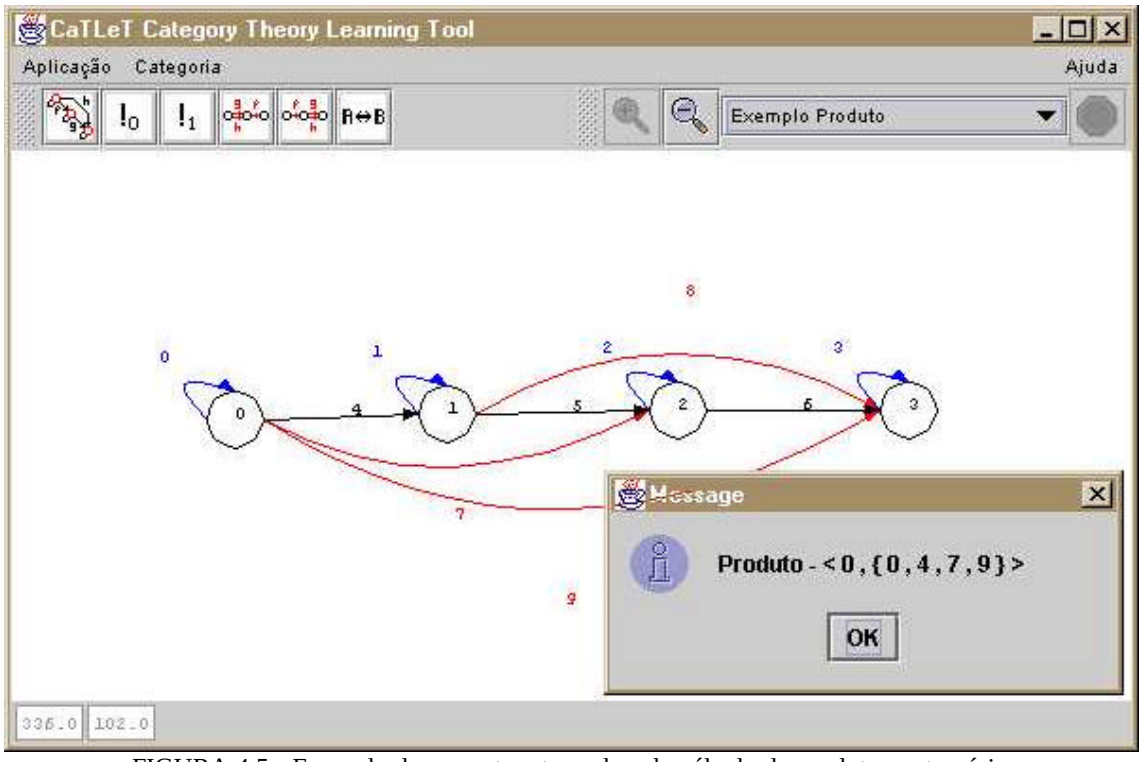
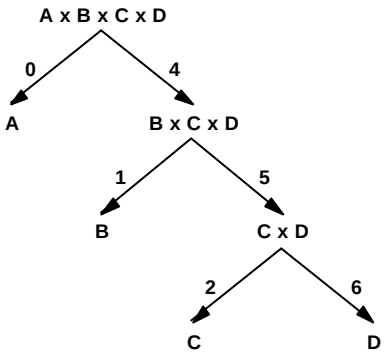


FIGURA 4.5 - Exemplo de resposta retornada pelo cálculo do produto quaternário

Se, em algum passo intermediário do algoritmo, não for encontrado produto binário, então o resultado final é a inexistência de produto. O retorno do método é o valor **null**.

Para cada passo $\mathbf{P_i}$ em que a cardinalidade **n** da coleção de entrada $\mathbf{C_1}$ é superior a dois, é efetuada uma invocação recursiva $\mathbf{P_{i+1}}$, passando-se como argumento para esta uma nova coleção $\mathbf{C_2}$, constituída por todos os elementos de $\mathbf{C_1}$ à exceção do primeiro, resultando $\mathbf{C_2}$ em uma nova coleção com **n-1** objetos.

Quando encerra a execução de $\mathbf{P_{i+1}}$ e o controle retorna a $\mathbf{P_i}$, a primeira verificação executada é se $\mathbf{P_{i+1}}$ retornou um produto válido, $\mathbf{p_i}$. Em caso afirmativo, $\mathbf{P_i}$ efetua nova invocação recursiva, à qual chamaremos $\mathbf{F(P_i)}$ (Fecho do Passo i). Desta vez, a coleção passada como argumento possui garantidamente apenas dois objetos, pela ordem, o primeiro objeto recebido na coleção de entrada por $\mathbf{P_i}$ e o objeto integrante do produto calculado por $\mathbf{P_{i+1}}$. $\mathbf{F(P_i)}$, portanto, não encadeia outras invocações.



Passo 1 < A , { 0 , 1o4 , 5o4 , 8o4 } > = < A , { 0 , 4 , 7 , 9 } >

Passo 2 < B , { 1 , 2o5 , 6o5 } > = < B , { 1 , 5 , 8 } >

Passo 3 < C , { 2 , 6 } >

FIGURA 4.6 - Árvore montada na execução do algoritmo de cálculo do produto quaternário

Ao final da execução de $\mathbf{F(P_i)}$, e conseqüente retorno do controle a $\mathbf{P_i}$, verifica-se se foi retornado produto válido, $\mathbf{f(p_i)}$. Em caso afirmativo, uma nova instância da classe **Produto** será construída a partir de $\mathbf{p_i}$ e $\mathbf{f(p_i)}$, da seguinte forma:

- o objeto deste novo produto é o objeto do produto $\mathbf{f(p_i)}$;
- o primeiro morfismo projeção integrante da tupla de morfismos do novo produto é o primeiro morfismo projeção de $\mathbf{f(p_i)}$;
- os demais morfismos projeção do novo produto são obtidos a partir da composição do segundo morfismo projeção de $\mathbf{f(p_i)}$ com cada morfismo projeção de $\mathbf{p_i}$.

Finalmente, o novo produto construído em $\mathbf{P_i}$ a partir destas regras é retornado ao passo $\mathbf{P_{i-1}}$.

4.6.4 Cálculos sobre Diagramas : Cone e Produto Fibrado

Os cálculos de Cones e Produtos Fibrados só têm sentido quando efetuados sobre diagramas.

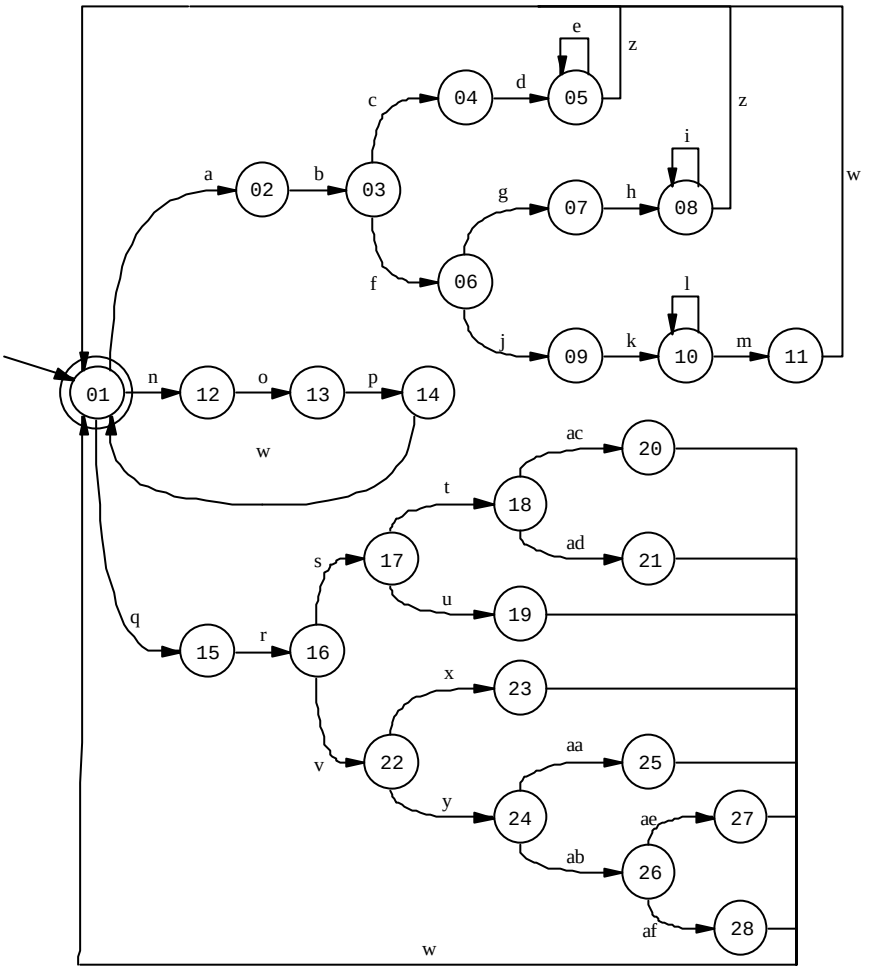
SimCat (seção 4.8) adotou uma solução parcial ao problema, implementando uma forma de se construir diagramas sobre a categoria corrente em uma janela auxiliar, e efetuando sobre os diagramas expressos nessa janela os referidos cálculos. A questão não foi plenamente resolvida em virtude de sua solução não contemplar a possibilidade de replicação de componentes.

CaTLeT atende integralmente a esta necessidade. Sua última etapa de projeto e implementação foi a interface destinada a possibilitar a construção de diagramas com componentes categoriais replicados, e a codificação dos algoritmos de cálculo de Cones e Produtos Fibrados segundo esta representação.

Inicialmente, idealizou-se a estrutura que modelaria diagramas. Criou-se a classe br.ufrgs.inf.catlet.categoria.Diagrama.java, a qual é bastante simples pois possui apenas dois atributos (uma coleção de objetos e outra coleção de morfismos), e modificou-se a classe Categoria, que passou a contar com um atributo de instância da classe recém-criada. Desta forma, cada categoria ativa na aplicação possui seu próprio diagrama, totalmente independente das demais.

O segundo passo foi projetar uma interface com o usuário. A mecânica encontrada para consistir as entradas do usuário na definição de diagramas seguiu autômato praticamente idêntico ao utilizado na construção de categorias, ao qual somou-se apenas algumas premissas:

- foi criada uma janela auxiliar para a edição de diagramas, a qual pode ser livremente redimensionada;
- um objeto só pode ser introduzido em um diagrama se o usuário aponta o objeto original na categoria a que o mesmo se refere. A identificação visual do objeto, assim como a vinculação do mesmo pelos algoritmos de cálculo ao objeto original da categoria, ocorre pelo nome do objeto;
- de forma similar ao que ocorre com objetos, morfismos só são introduzidos se existem na categoria original.



a - mousePressed
b - Sim
c - Sim
d - mouseDragged
e - mouseDragged
f - Não
g - Sim
h - mouseDragged
i - mouseDragged
j - Não
k - mouseDragged
l - mouseDragged
m - mouseReleased
n - mouseMoved
o - Sim
p - Sim
q - mouseClicked
r - Sim
s - Sim
t - Sim
u - Não
v - Não
x - Sim
y - Não
z -mouseReleased
aa - Sim
ab - Não
ac - Sim
ad - Não
ae - Sim
af - Não
w - movimento vazio

01 - Aguarda evento
02 - Há categoria ativa?
03 - O evento ocorreu sobre ponto de inflexão previamente selecionado de algum morfismo?
04 - Aguarda evento
05 - Redesenha o morfismo conforme a nova coordenada do ponto de inflexão 'arrastado'
06 - O evento ocorreu sobre objeto?
07 - Prepara todos os objetos que se encontram selecionados para movimentação. Aguarda evento
08 - Redesenha todos os objetos selecionados segundo o deslocamento aplicado ao mouse
09 - Inicia a abertura de retângulo de seleção nas coordenadas em que o mousePressed ocorreu. Aguarda even
10 - Redesenha o retângulo de seleção, tomando vértice destino a nova coordenada do mouse
11 - Seleciona todos os objetos e morfismos contidos na área demarcada pelo retângulo de seleção
12 - Há categoria ativa?
13 - Há objeto origem definido (está sendo criado morfismo) ?
14 - Traça linha entre centro do objeto origem e ponto em que ocorreu o evento
15 - Há categoria ativa?
16 - O evento ocorreu sobre um objeto ?
17 - Há objeto origem definido (está sendo criado morfismo) ?
18 - Verifica se existe morfismo na categoria entre os objetos origem e destino.
Se existe, abre janela listando todos estes morfismos.
Selecionou um morfismo?
19 - Inicia processo de criação de morfismo, definindo objeto em que ocorreu o click como origem
20 - Cria réplica do morfismo selecionado
21 - Não faz nada
22 - O evento ocorreu sobre um morfismo ?
23 - Inverte seleção do morfismo (e.g. seleciona se não está selecionado)
24 - Há algum objeto ou morfismo selecionado?
25 - Desseleciona objetos e morfismos selecionados.
26 - Abre janela para seleção do objeto a replicar, listando todos os objetos existentes na categoria.
Selecionou algum objeto?
27 - Cria réplica do objeto selecionado
28 - Não faz nada

FIGURA 4.7 - Fluxo de manutenção de diagramas em CaTLeT

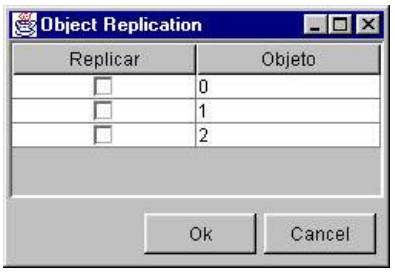


FIGURA 4.8 – Dialog de identificação de objetos a replicar em CaTLeT

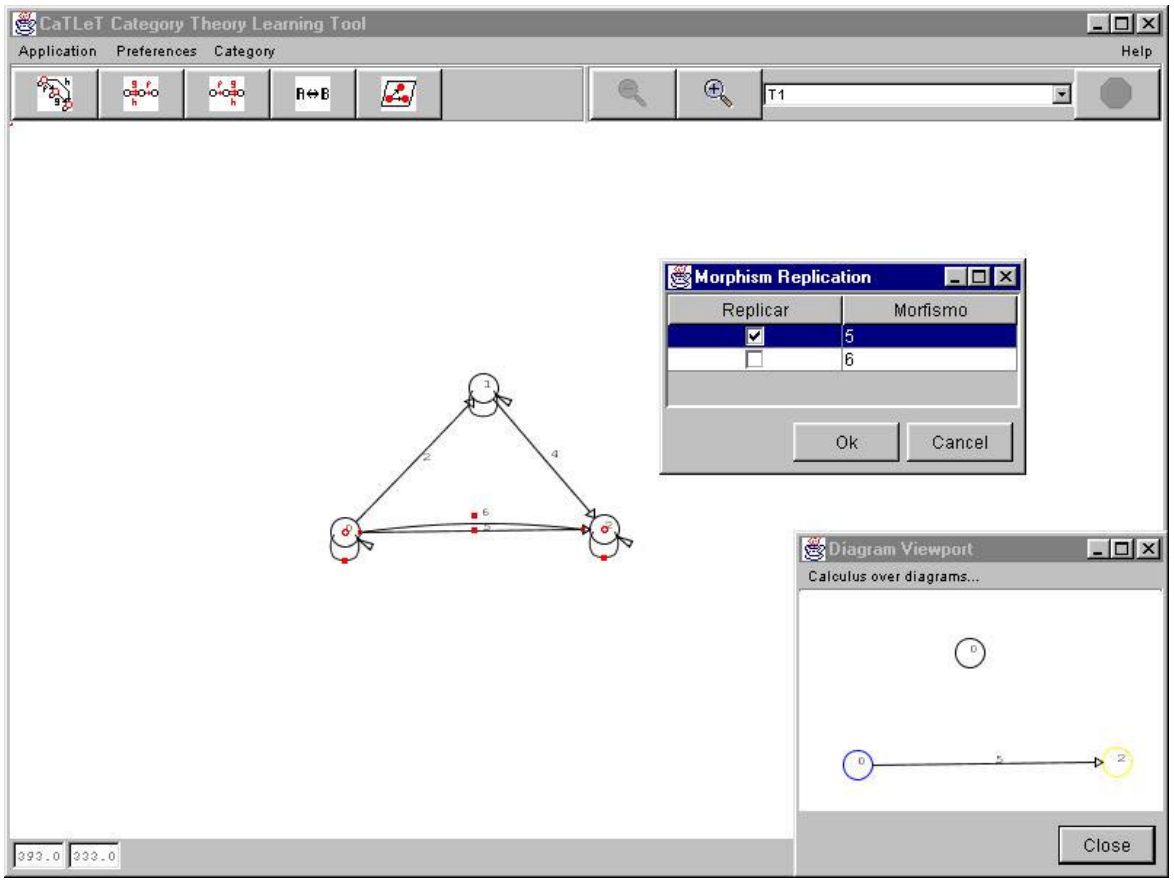


FIGURA 4.9
Window de Construção de Diagramas (Diagram Viewport)
Dialog de identificação de morfismos a replicar em CaTLeT

4.7 Experiência Didática - construção de ferramenta em disciplina de graduação

Desenvolveu-se, no segundo semestre letivo do ano 2000, na disciplina *Categorias Computacionais* ministrada ao Bacharelado em Ciência da Computação da UFRGS, uma experiência paralela ao projeto aqui apresentado. Tal iniciativa fundamentou-se nas idéias apresentadas e defendidas por este trabalho.

Foi enunciado aos integrantes da disciplina o desenvolvimento de aplicação categorial cujo espectro de funcionalidades constituía subconjunto relativamente expressivo da súmula de conteúdos que embasa tanto a disciplina quanto o projeto ora apresentado.

Os objetivos associados a esta iniciativa, entretanto, eram bastante específicos. Visava-se, com esta iniciativa, induzir os instruendos a uma maior exposição aos conceitos e maior reflexão sobre os mesmos. Afinal:

- implementar um algoritmo implica a existência de uma fase anterior, em que o algoritmo é definido;
- definir um algoritmo implica compreender o problema que o mesmo se propõe a resolver;
- compreender o problema implica necessariamente na formulação do enunciado do problema;
- por fim, enunciar o problema implica estudar cuidadosamente o problema - na experiência em questão, estudar a definição para a qual se está criando o algoritmo.

O único limite técnico estabelecido para a tarefa consistia da súmula de conceitos que a versão final de cada ferramenta deveria ser capaz de trabalhar:

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

Nada foi especificado sobre plataforma ou ferramentas a utilizar, tampouco estabeleceu-se qualquer premissa para as características de interface. A única providência tomada no sentido de reduzir o esforço daqueles que tivessem maior ousadia e criatividade, e por tal dispendessem maior tempo com a tarefa, foi permitir a execução do trabalho em duplas.

A grande possibilidade de êxito inicialmente antevista baseava-se em certas características do cenário da experiência que prenunciavam-se favoráveis:

- os conceitos categoriais, conforme já discutido, demandam certo tempo e esforço para maturação. A metodologia de apresentação e avaliação até então utilizada não apresentava os mesmos resultados expressivos de outras disciplinas, deixando espaço para que algo novo fosse experimentado;
- espera-se de alunos de Ciência da Computação que os mesmos tenham grande aptidão e profundo gosto e interesse pelas atividades associadas ao projeto e à programação de software. A proposição de um trabalho prático de programação, desta forma, poderia ser um estímulo e e complementaria o estudo puramente teórico de Teoria das Categorias.

Como estímulo adicional, grande parte da aprovação na disciplina foi condicionada ao bom desenvolvimento da aplicação proposta. O percentual atribuído ao trabalho no somatório de avaliações da disciplina poderia atingir até 60%, o suficiente para eximir os alunos de qualquer outra avaliação, em caso de aproveitamento máximo nesta atividade.

A súmula foi subdividida em três partes, e foram planejados dois checkpoints intermediários no semestre, antes da apresentação da versão final. Em cada checkpoint, era avaliado o atingimento dos objetivos traçados para aquele momento, sempre coincidentes com o ponto em que a disciplina se encontrava. Efetuavam-se as críticas e o intercâmbio de conhecimento necessário à resolução dos problemas.

Os checkpoints atingiram plenamente seus objetivos, pois induziram os alunos a desenvolver o trabalho de forma gradativa desde o início do semestre, e não ocorreu acúmulo demasiado de dúvidas.

Os resultados da experiência foram, de forma geral, bastante satisfatórios. Em sua grande maioria, os freqüentadores da disciplina trabalharam em duplas, e conforme a expectativa inicial, demonstraram ter assimilado e compreendido os conceitos em prazos mais exíguos e com maior profundidade que seus predecessores.

A grande maioria dos trabalhos desenvolvidos, entretanto, desenvolveu interfaces homem-máquina centradas em entradas e saídas de dados orientadas a caracter. Como foi dito anteriormente, não havia qualquer requisito técnico imposto às soluções, exceto o de atender à sumula proposta. Conseqüentemente, não constituía objetivo apurar se soluções desta natureza viriam a ser adotadas por falta de percepção, arrojo, conhecimento ou até mesmo por limitações de tempo (afinal deve ser sempre considerado que o aluno de Ciência da Computação da UFRGS tem liberdade para planejar as disciplinas que irá cursar em um semestre, observados os pré-requisitos, e é regra que compartilhe seu tempo entre várias disciplinas). O fato é que, por qualquer das razões aqui enumeradas, ou até mesmo por outras alheias ao nosso conhecimento, a grande maioria abstraiu as possibilidades oferecidas por interfaces com recursos de manipulação gráfica dos elementos componentes de categorias. Apenas três trabalhos fizeram algum uso deste requintado expediente técnico.

4.8 SimCat

O princípio do projeto CaTLeT, que consiste da especificação, projeto e implementação de uma ferramenta computacional de apoio à representação e cálculos categoriais, pôde ser aplicado com grande propriedade como especificação técnica para atividade prática desenvolvida no âmbito de uma disciplina de graduação em Ciência da Computação, conforme apresentado anteriormente.

Os bons resultados alcançados por essa proposta despertaram e imbuíram o grupo do espírito de realimentação do processo de desenvolvimento de CaTLeT. Vislumbrou-se a possibilidade de obter maior proveito de experiências similares.

Um dos alunos de melhor performance no trabalho proposto na disciplina *Categorias Computacionais*, Tiago Mourão, coincidentemente era formando no semestre seguinte (2001/01). O grupo apresentou-lhe o projeto e os objetivos de

CaTLeT, e convidou-o a integrar a equipe e tomar parte especificamente neste projeto, propondo a ele um objetivo bastante 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, um laboratório experimental para o ideário conceitual e funcional de CaTLeT.

Os objetivos do grupo e de Tiago com a iniciativa foram alcançados. O resultado prático é uma ferramenta de nome SimCat [MOU2001]. Em função da limitação de tempo a que se deve sujeitar um TFG (Trabalho Final de Graduação), que no período delimitado pelos eventos inicial (formulação e submissão da proposta) e final (defesa perante Banca) do trabalho desenvolvido compreende aproximadamente cinco meses, o principal elo entre SimCaT e CaTLeT foi a especificação e implementação conjunta dos principais algoritmos de cálculo. Algumas características que as pesquisas e discussões em grupo apontaram como relevantes (e.g. cálculo de produto e coproduto categoriais de aridade maior que 2) não foram incorporadas a SimCat. Em contrapartida, foram exploradas pelo grupo em maior profundidade várias possibilidades de funcionalidades e características de interface com o usuário, o que configurou resultado complementar não-proposto, mas bastante satisfatório. Parte destas contribuições foi incorporada a CaTLeT.

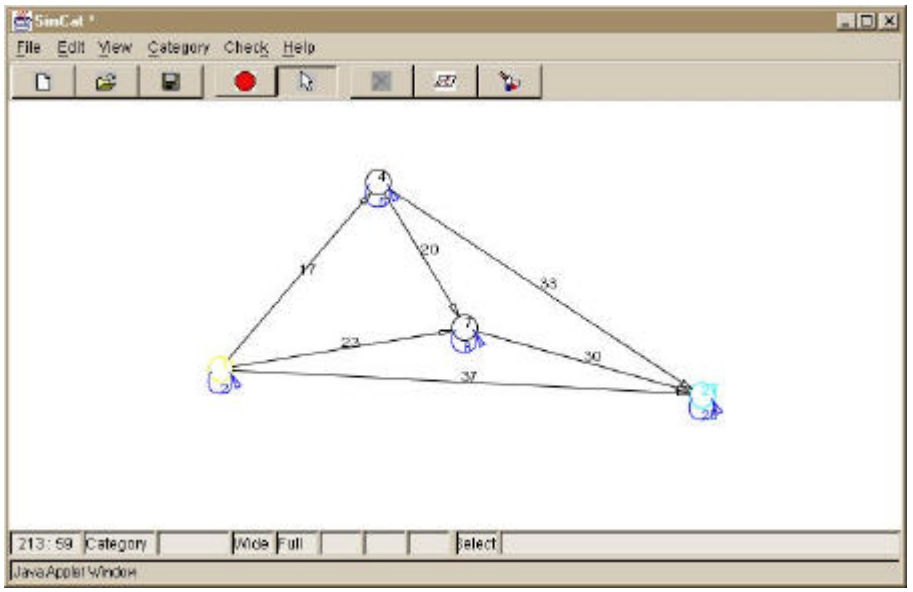


FIGURA 4.10 - Tela principal de SimCat

5 Características de Implementação

5.1 Metodologia de Desenvolvimento e Paradigma de Programação

Os últimos anos foram palco do desenvolvimento de toda uma nova cultura associada ao paradigma de Orientação a Objetos. Inicialmente bastante mistificados, gradativamente estes conceitos amadureceram e foram compreendidos, assimilados e incorporados. Atualmente suas variantes metodológicas de análise, projeto e desenvolvimento competem fortemente com metodologias consagradas como Análise Estruturada e Engenharia da Informação. [FOW97]

Os conceitos inerentes às linguagens de programação orientadas a objeto, entretanto, amadureceram com maior velocidade, e consequentemente foram assimilados com muito mais agilidade que as metodologias de análise e projeto. De fato, as linguagens de programação orientadas a objeto foram as principais responsáveis pela inserção e difusão da filosofia OO tanto no meio acadêmico quanto no mercado de Tecnologia da Informação.

A conseqüência deste "descasamento de impedâncias" entre a maturação das técnicas de análise e projeto e as ferramentas de programação OO é facilmente depreendida da análise do mercado de TI, a partir do qual percebe-se que linguagens de programação OO, e.g. C++ e Java, a última em especial, estão rapidamente tomando o lugar de suas antecessoras procedurais. As possibilidades de uso da Internet em ambiente corporativo tem gerado e rapidamente incorporado conceitos como *Business to Business* e *Business to Consumer* a áreas empresariais estratégicas. Linguagens de programação OO tem sido muito aplicadas neste ambiente, em razão de sua natureza inerentemente extensível.

Já existe alguma procura por profissionais com conhecimento e experiência na aplicação de metodologias de análise e projeto OO. A demanda de mercado por profissionais aptos a trabalhar com ferramentas CASE baseadas em Análise Estruturada e Engenharia da Informação, entretanto, ainda predomina amplamente. E quando se fala de sistemas gerenciadores de bases de dados, a soberania dos SGBDs relacionais sobre sistemas OO é absoluta, a despeito das dificuldades técnicas que decorrem da implementação de uma aplicação desenvolvida utilizando linguagem de programação orientada a objetos sobre uma base de dados relacional. [CAT98]

Desde o início do projeto percebeu-se que as técnicas e linguagens de modelagem oferecidas por Análise Estruturada e Engenharia da Informação pouco contribuiriam para a adequada especificação da aplicação que então se idealizava.

Análise e Programação Orientada a Objetos, entretanto, permitem expressar e utilizar conceitos como *Reuso de Software por Herança* e *Encapsulamento de Operações*, bastante úteis em nosso contexto, como será apresentado adiante.

Os muitos recursos encontrados somente em Linguagens de Programação Orientada a Objetos (e.g. Herança, Encapsulamento, Polimorfismo) foram amplamente utilizados em todo o processo. Entretanto, lançou-se mão apenas de um subconjunto dos recursos de Análise Orientada a Objetos. Por conta do caráter inovador do projeto, alguns recursos (e.g. Use Cases) não foram explorados. As ferramentas utilizadas da AOO no projeto do software categorial de que trata este trabalho restringem-se a:

- diagrama de packages;
- diagrama de classes.

5.2 Caracterização de Qualidade em Produtos de Software

Alguns termos associados à qualidade de produtos de software foram utilizados ao longo do trabalho até o presente momento em nível informal. Entretanto, a intensidade com que tais termos são empregados a partir de agora torna necessário que se defina a semântica associada a cada um destes.

Em uma visão restrita, a qualidade do produto de software é considerada sinônimo de confiabilidade [CON86]. Entretanto, um sistema de software que não possui erros, mas cujo código-fonte é de difícil compreensão e manutenção não pode ser considerado de boa qualidade [BOE78].

Vários modelos hierárquicos de qualidade de software já foram propostos. Um dos mais citados e abrangentes, entretanto, é o modelo de McCall, proposto em 1977. Este modelo é endereçado a desenvolvedores de software, a fim de ser utilizado durante o processo de construção do sistema. Este modelo identifica três áreas de trabalho de software:

- Operação do Produto: requer que o software seja aprendido facilmente, operado eficientemente e que os resultados sejam os esperados pelo usuário;
- Revisão do Produto: está relacionada à correção dos erros e à adaptação do sistema a novas características;
- Transição do Produto: pode não ser importante em todas as aplicações. Entretanto, a transição para sistemas distribuídos e o rápido aumento na troca de hardware fazem com que esta área tenha sua importância enfatizada.

Os critérios de qualidade definidos por McCall são apresentados a seguir:

- **Usabilidade** é a facilidade encontrada pelo usuário na utilização do software;
- **Integridade** é a proteção do programa contra o acesso não autorizado;
- **Eficiência** está relacionada à utilização de recursos. Subdivide-se em eficiência na execução e eficiência no armazenamento;
- **Correteza** é a garantia de que o programa atende à especificação do qual se originou;
- **Confiabilidade** é a habilidade do programa de não incorrer em falhas;
- **Manutenibilidade** é o esforço necessário para localizar e corrigir uma falha no programa dentro de seu ambiente operacional;
- **Flexibilidade** é a facilidade de se realizar quaisquer modificações necessárias no software após este já estar em ambiente operacional;
- **Testabilidade** é a facilidade de testar o programa, assegurando que ele não contém erros e satisfaz sua especificação;
- **Portabilidade** é o esforço necessário para transferir um programa de um ambiente;
- **Reusabilidade** é a facilidade de reusar o software, ou porções do mesmo, em um contexto diferente;
- **Interoperabilidade** é o esforço necessário para integrar um sistema com algum outro.

É bom destacar que, à época da definição deste modelo, a Reusabilidade não possuía mais do que uma pequena fração da importância de hoje. Atualmente, este

critério constitui um dos alicerces sobre o qual as tecnologias orientadas a objeto estão situadas.

Adicionalmente, componentes de software (middleware) que implementam camadas intermediárias entre código-fonte e sistema operacional, como os *plug-ins* para processamento de linguagens de script e as máquinas virtuais Java, eram recursos impensáveis. Portabilidade adquire grande importância com a proliferação da microinformática e o avanço do *downsizing*.

Dentre todos os pontos constantes no modelo de McCall, as características que mais almeja-se incorporar a CaTLeT são Usabilidade, Eficiência, Correteza, Confiabilidade, Portabilidade e Reusabilidade. Muito dos esforços de projeto e desenvolvimento empreendidos ocorreram pautados por pelo menos uma destas diretivas.

5.3 Diagrama de Packages

O princípio da Divisão e Conquista permeia amplamente vários campos do conhecimento humano. Ciência da Computação também utiliza, e de forma bastante intensa e diversificada, esta técnica. Pesquisas em Complexidade de Algoritmos, e.g., deram origem a um método de desenvolvimento de algoritmos por decomposições e recombinações.

Uma das questões mais discutidas em métodos de análise e projeto de software é justamente a segmentação de uma aplicação em porções de complexidade proporcionalmente inferior.

Métodos estruturados utilizam decomposição funcional, sendo o Diagrama de Fluxo de Dados (DFD) a ferramenta de apoio básica a tal técnica. Processos e dados, entretanto, são vistos separadamente. Algumas técnicas associadas à Engenharia da Informação buscam agrupar os dados por critérios semânticos, e utilizam matrizes para estabelecer o relacionamento entre as funções e os dados sobre os quais estas funções atuam.

É sob esta ótica que se pode perceber a maior mudança que a orientação a objetos proporciona. A separação de dados e processos, bem como a decomposição funcional, há muito surgiram como técnicas para o particionamento de aplicações, mas o problema ainda persiste.

Classes são abstrações que encapsulam os atributos que definem o estado de uma entidade de interesse, bem como os procedimentos (métodos) com potencial de modificação deste estado, definindo suas reações às interações com o meio externo - em outras palavras, o comportamento das instâncias da classe. Ou seja, dados e processos são tratados conjuntamente, agrupados em pequenas unidades semânticas. Uma aplicação que faça bom uso das técnicas de modelagem de classes especifica todos os seus processos com base na verificação do estado de seus objetos, e ações sobre objetos (desencadeadas por métodos) para a alteração do estado do sistema.

A exemplo dos métodos estruturados, o paradigma OO também propõe uma técnica para o tratamento da complexidade de sistemas. A técnica oferecida consiste em agrupar as classes da aplicação em unidades semanticamente maiores, denominadas *packages*, e expressar as relações de dependência entre os vários packages de uma aplicação, em um *diagrama de packages*.

Existe uma dependência entre dois packages se alterações à definição de um elemento do package origem pode implicar alterações no estado de alguma instância do package destino.

Os tipos de dependência que podem existir entre classes são:

- uma classe envia uma mensagem (executa um método) a outra;
- uma classe utiliza outra na definição de seu estado (como algum de seus atributos);
- uma classe utiliza outra como estrutura auxiliar ao processamento no corpo de algum método;
- uma classe utiliza outra na definição da assinatura de algum método (como parâmetro de entrada ou valor de retorno do método).

Com o desenvolvimento de classes e interfaces Java ocorrendo paralelamente em todo o mundo, é perfeitamente concebível que dois programadores utilizem o mesmo nome em suas classes. Suponha-se agora um integrador que pretenda fazer uso das classes destes dois desenvolvedores. Se as classes não estiverem em packages, o uso

contíguo das mesmas estará inviabilizado. Entretanto, se as classes estiverem em packages com nomes distintos, não haverá problema, desde que o integrador especifique em seu programa, quando acessar estas classes, a qual package cada ocorrência se refere. Mas o quê pode evitar que o nome dos packages coincida? O respeito à seguinte convenção, proposta pela SUN Microsystems, desenvolvedora da linguagem Java, no *pattern* J2EE [ALU2001], e que constitui atualmente um padrão *de facto* mundialmente seguido: ao projetar um package, utilize o reverso do domínio de sua corporação, acrescendo, se julgar necessário, o nome do projeto ou região.

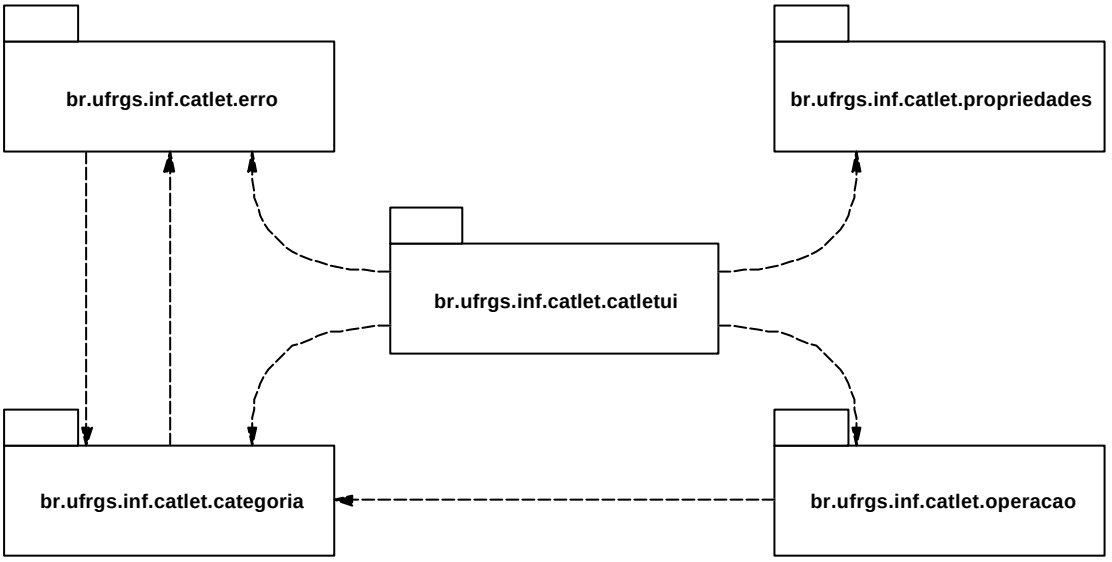


FIGURA 5.1 - Diagrama de packages de CaTLeT

O uso de tal convenção, no projeto de CaTLeT, deu origem aos seguintes packages:

- br.ufrgs.inf.catlet.categoria - contém as classes que definem uma categoria;
- br.ufrgs.inf.catlet.catletui - contém as classes que definem as interfaces com o usuário;
- br.ufrgs.inf.catlet.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.catlet.operacao - neste package estão agrupadas classes de apoio aos algoritmos de cálculo;
- br.ufrgs.inf.catlet.propriedades - composto por apenas uma classe, Propriedades, destinada a prover à aplicação uso facilitado das configurações de propriedades.

Apenas um arquivo, CaTLeT.java, por constituir a classe que inicia a execução da aplicação, não integra qualquer dos packages da mesma. Este encontra-se na raiz da estrutura de diretórios da mesma. Todas as demais integram algum dos cinco packages descritos acima.

Os packages constituem unidades lógicas Java que possuem tradução física, em sistemas operacionais, para estruturas de diretórios. Suponha-se que se está trabalhando em um sistema de arquivos Windows, e a aplicação esteja toda contida em um diretório denominado catlet, localizado diretamente na raiz da unidade de disco principal, identificada pela letra "c".

A organização dos arquivos integrantes da aplicação, então, assume a seguinte conformação:

- c:\catlet\CaTLeT.java
- c:\catlet\br\ufrgs\inf\catlet\categoria
  - Categoria.java
  - CollectionMorfismos.java
  - Diagrama.java
  - Morfismo.java
  - Objeto.java
  - ParMorfismos.java
- c:\catlet\br\ufrgs\inf\catlet\catletui
  - CategoryFileFilter.java
  - DiagramBuilder.java
  - DiagramBuilderPanel.java
  - DiagramViewport.java
  - DialogColorChooser.java
  - DialogCompositionReport.java
  - DialogMorphismReplication.java
  - DialogObjectReplication.java
  - WindowAbout.java
  - WindowErrorsList.java
  - WindowManutCategoria.java
  - WindowManutMorfismo.java
  - WindowManutObjeto.java
  - WindowPreferences.java
  - WindowProduct.java
- c:\catlet\br\ufrgs\inf\catlet\erro
  - ErroAssociatividade.java
  - ErroCategoria.java
  - ErroComposicao.java
  - ErroIdentidade.java
- c:\catlet\br\ufrgs\inf\catlet\operacao
  - Cone.java
  - Equalizador.java
  - PreProduto.java
  - Produto.java
- c:\catlet\br\ufrgs\inf\catlet\propriedades
  - Propriedades.java

5.4 Diagramas de Classes

5.4.1 Package br.ufrgs.inf.catlet.propriedades

Uma única classe integra esta package. Criou-se uma package exclusivamente para receber a classe abstrata Propriedades em virtude de não haver nenhuma relação semântica entre a função desempenhada por esta com qualquer outra classe da aplicação.

O papel desempenhado pela classe Propriedades está relacionado ao projeto de internacionalização / localização da aplicação. Por ser abstrata, não pode dar origem a instâncias, e seus dois únicos métodos são declarados métodos de classe.

Sua função é prover à aplicação uma forma simples de acessar os arquivos com textos para localização.

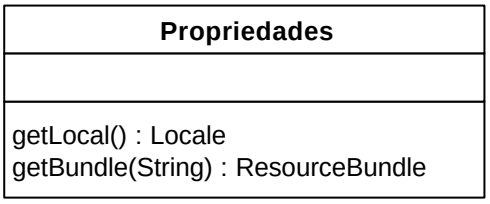


FIGURA 5.2 - Diagrama de Classes do package br.ufrgs.inf.catlet.propriedades

5.4.2 Package br.ufrgs.inf.catlet.erro

As classes desta package visam prover suporte à identificação de erros que um grafo de usuário apresenta perante a definição de categoria.

As classes ErroIdentidade, ErroComposicao e ErroAssociatividade estendem a classe ErroCategoria, herdando portanto os atributos e comportamento especificados nesta. Esta característica comum às classes que identificam os três tipos de erro possíveis a um grafo em CaTLeT perante a definição categorial, além de concentrar definições de atributos comuns a todos os tipos de erro em um único local, também permite que se delegue a um único método a verificação dos três tipos de erro. O retorno deste método é uma coleção de instâncias das várias classes da package br.ufrgs.inf.catlet.erro, as quais são recuperadas via *type casting*.

Uma das funcionalidades que se vislumbrou incorporar à aplicação quando do projeto das estruturas que compõem esta package era a correção automática de erros. Embora esta característica não tenha sido incorporada, e seja sugerida na seção de trabalhos futuros, as estruturas já estão disponíveis para esta extensão das capacidades de CaTLeT.

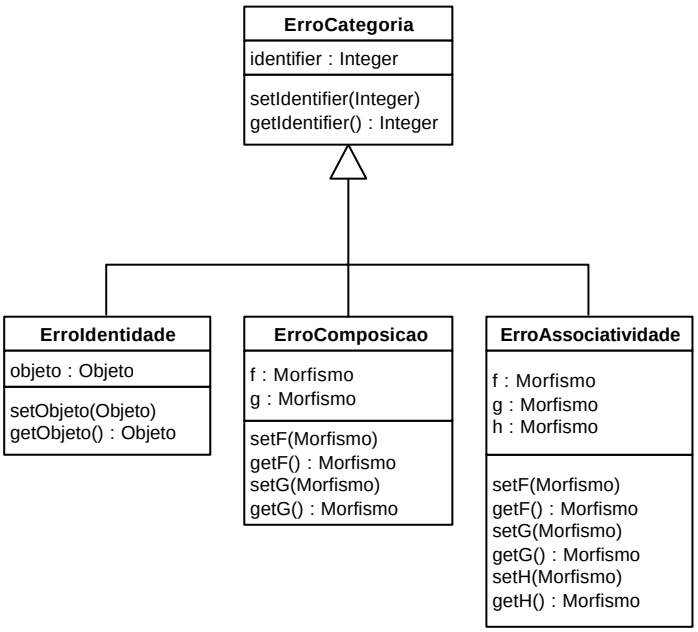


FIGURA 5.3 - Diagrama de Classes do package br.ufrgs.inf.catlet.erro

5.4.3 Package br.ufrgs.inf.catlet.operacao

As integrantes desta package são estruturas auxiliares utilizadas nos cálculos categoriais implementados em CaTLeT.

Instâncias da classe PreProduto armazenam resultados intermediários que decorrem da execução do primeiro passo do algoritmo para cálculo do produto categorial.

A classe Produto, entretanto, constitui tanto uma estrutura auxiliar ao cálculo do produto categorial, quanto o resultado final da execução do mesmo.

A classe Cone é utilizada nos cálculos de Cone e Limite. A resposta fornecida pela execução dos métodos responsáveis pelo dois cálculos é provida como uma instância desta.

A classe Equalizador é utilizada como auxiliar no cálculo do equalizador de dois morfismos.

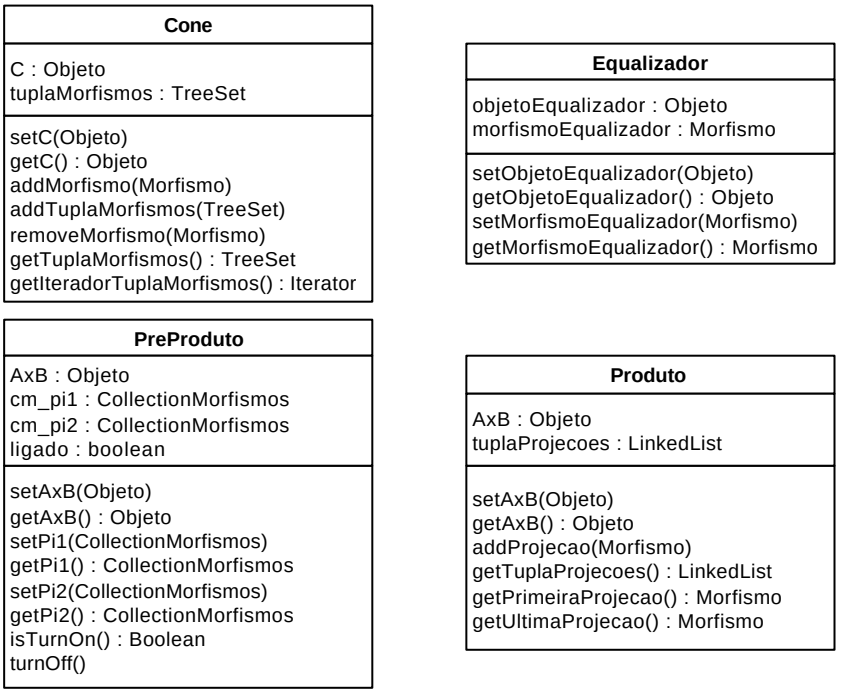


FIGURA 5.4 - Diagrama de Classes do package br.ufrgs.inf.catlet.operacao

5.4.4 Package br.ufrgs.inf.catlet.categoria

Esta package constitui o núcleo da aplicação. É nela que estão inclusas as classes que definem uma categoria e seus componentes, bem como algumas estruturas auxiliares.

Em CaTLeT, a classe Categoria, apesar de possuir muitos atributos, fundamenta sua existência basicamente em dois destes: uma coleção de objetos e uma coleção de morfismos.

Os elementos da coleção de objetos de uma instância da classe Categoria são instâncias da classe Objeto, também uma classe integrante desta package.

Adicionalmente, os elementos da coleção de morfismos de uma instância da classe Categoria são instâncias da classe CollectionMorfismos, mais uma pertencente a esta package. E uma instância desta última é uma coleção de todos os morfismos paralelos da categoria, sendo cada morfismo paralelo uma instância da classe Morfismo. Assim, pode-se dizer que a coleção de morfismos implementada em CaTLeT é, na verdade, uma coleção de coleções de morfismos, agrupados pelo critério de origem e destino coincidentes.

A despeito da idéia originalmente esboçada, não há uma classe Composicao. Esta foi substituída por uma coleção de pares de morfismos que resulta da composição dos dois arcos sequentes e encontra-se definida como atributo da classe Morfismo.

As justificativas para a modelagem assumida serão devidamente discutidas ao longo da seção 5.6 - Técnicas e Tecnologias Utilizadas.

Página que será substituída pelo diagrama de classes para a package br.ufrgs.inf.catlet.categoria

5.4.5 Package br.ufrgs.inf.catlet.catletui

É a package em que estão contidas todas as telas desenvolvidas para interface com o usuário.

As principais classes desta package, e de toda a aplicação, são DiagramBuilder (o container da aplicação) e DiagramBuilderPanel (a área destinada à edição de grafos). É a partir destas que toda a ação de CaTLeT é coordenada.

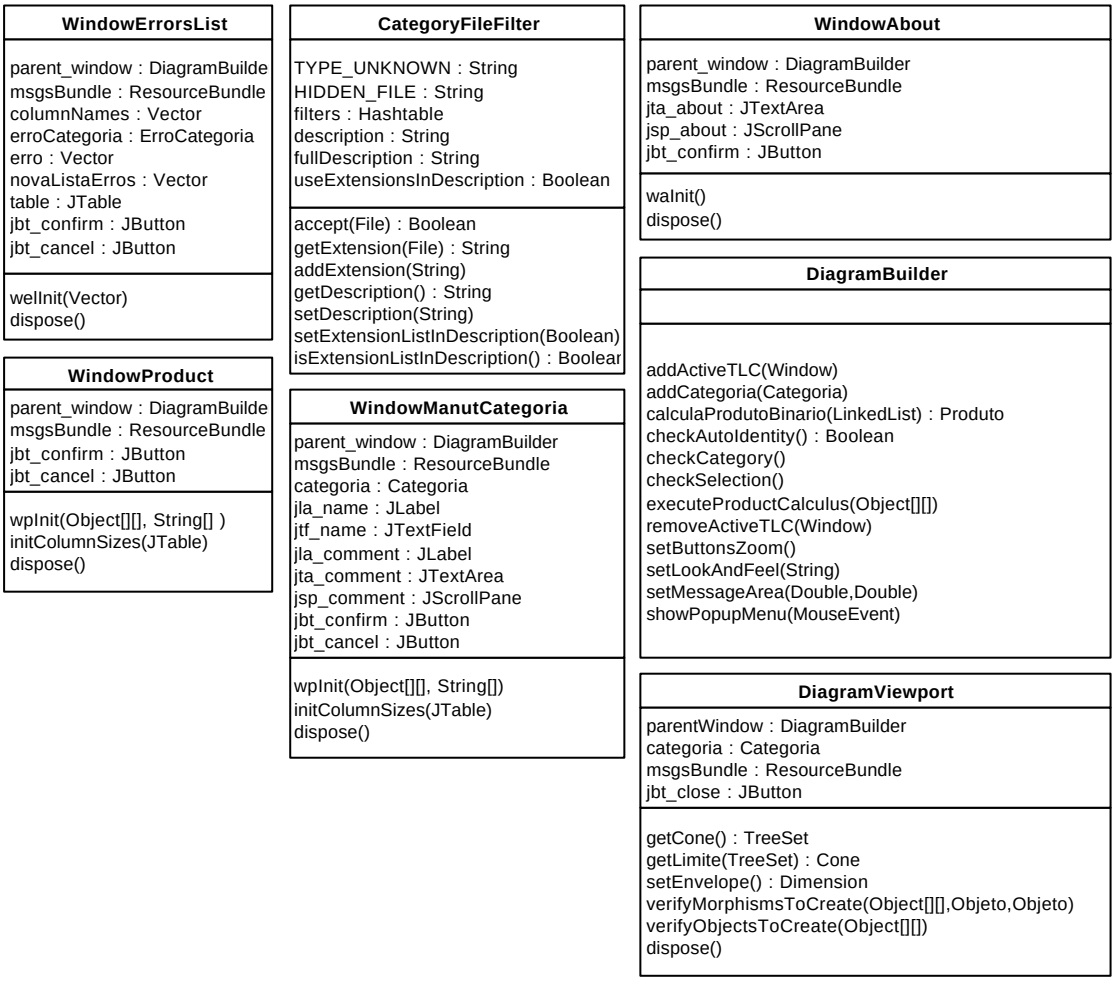


FIGURA 5.6 - Diagrama de Classes parcial (principais classes) do package br.ufrgs.inf.catlet.catletui

5.5 Ferramentas Utilizadas

Conforme o que já foi detalhado no capítulo anterior, durante a análise das ferramentas categoriais pré-existentes, constatou-se que as funcionalidades da aplicação Category Theory Database Tools (CTDT) tinham sido implementadas, basicamente, a partir de uma tradução do código-fonte de sua antecessora, Category Construction Program.

Os primeiros esboços de projeto para CaTLeT embutiram de forma natural várias das características e conceitos encontrados em Programação Orientada a Objetos.

Adicionalmente, portabilidade (independência de plataforma) e possibilidade de utilização da ferramenta em ambiente Internet eram características que muito interessavam ao grupo.

Dentro desse contexto, a escolha pela linguagem Java ocorreu de forma bastante natural, pois a mesma apresentava as três características:

- linguagem de programação orientada a objetos;
- independente de plataforma graças ao conceito/middleware JVM;
- habilitada à execução em ambiente Web, por meio do uso de applets e servlets.

Com o paradigma de programação a ser utilizado em nosso projeto definido, e a escolha coincidindo com o paradigma utilizado na ferramenta pré-existente 2 (a applet Java), uma nova análise, em nível estrutural e mais profunda que a anterior, foi executada sobre os elementos constituintes da mesma. A discussão prossegue, a partir de agora, focalizada nas soluções técnicas adotadas na ferramenta 2.

A análise técnica da applet Java evidenciou a falta de um reprojeto que fizesse bom uso das potencialidades oferecidas pela Programação Orientada a Objetos. Adicionalmente, há recursos específicos à linguagem de programação utilizada que, se tivessem sido explorados, poderiam ter economizado esforço de implementação ao mesmo tempo em que teriam agregado eficiência e flexibilidade, resultando em maior qualidade.

Foram identificados vários pontos em que a aplicação analisada poderia ter ganhos de diferentes ordens:

- a ferramenta utiliza arrays para armazenar as coleções de elementos de uma categoria, que são estruturas com capacidade explicitamente definida em tempo de compilação. Um artifício de projeto utilizado por seus desenvolvedores claramente objetivou minimizar os efeitos negativos da solução utilizada. Definiram-se as constantes que estabelecem as capacidades dos arrays em um único arquivo, visando tornar necessária a compilação exclusivamente deste para que a aplicação reflita novas capacidades. Tal solução, entretanto, pode ser substituída com ganho de flexibilidade e usabilidade por qualquer das várias implementações disponíveis no framework de Collections disponível em Java, cujas capacidades são limitadas apenas pelo hardware subjacente;
- os componentes de uma categoria (objetos e morfismos) são entidades com estado e comportamento próprios, induzindo, desta forma, a modelagem de cada um como uma classe, permitindo a plena utilização das vantagens oferecidas pelo encapsulamento. A applet canadense fez uso de arrays de strings, estrutura que certamente necessita de outras auxiliares, espalhadas ao longo da aplicação, para definir seu estado e comportamento;
- certas estruturas que foram desenvolvidas especificamente para a ferramenta eram desnecessárias, pela existência, já à época do desenvolvimento da aplicação, de

classes já embutidas em Java de funcionalidade similar. Deve-se ter em mente, entretanto, que CTDT utilizou, para a visualização de categorias, um pacote de classes desenvolvido por terceiros totalmente implementado utilizando Java 1.0, anterior, portanto, ao advento de Java API 2D. Por exemplo, a classe DPoint, destinada a armazenar pontos bidimensionais, pode ser substituída utilizando-se qualquer das implementações da classe abstrata java.awt.geom.Point2D (Double ou Float) integrantes da API 2D de Java. Tem-se como conseqüências diretas desta substituição a menor necessidade de desenvolvimento de código específico, bem como de maior eficiência da aplicação, pela execução de código embutido na Máquina Virtual Java local. Em se tratando a ferramenta 2 de uma applet, a menor transferência de código específico também contribui para maior eficiência de execução, uma vez que menor quantidade de *bytecodes* precisa ser transferida do servidor para o host remoto.

Uma das características que se pretendia associar ao projeto é exatamente o uso exclusivo, durante seu desenvolvimento, de ferramentas sem custo de licenciamento. Embora tenha sido desenvolvida sobre plataforma Windows, é totalmente viável executar o software como aplicação *standalone* em qualquer ambiente computacional para o qual exista máquina virtual Java implementada de versão 1.3 ou superior (atualmente a Sun já disponibiliza a versão beta 1.4).

Todos os ganhos que se poderia obter pelo uso das características supracitadas foram importantes na decisão final pela adoção de Java como a linguagem de programação OO para desenvolvimento da aplicação. Contudo, a decisão não foi tomada com base apenas nestas características. Até que a resolução final fosse tomada, pesquisou-se outras potencialidades que a linguagem pudesse possuir.

Todas as virtudes enunciadas a seguir foram decisivas na escolha:

- inúmeras ferramentas para desenvolvimento de aplicações Java podem ser obtidas sem qualquer custo, e.g.:
  - JDK: pacote de ferramentas para compilação e depuração de programas Java, bem como geração automática de documentação, provido pela Sun Microsystems;
  - Tomcat: servidor de aplicação habilitado a Servlets e Java Server Pages (JSP). Desenvolvido pelo mesmo grupo responsável pelo *webserver* Apache, ainda hoje principal servidor Internet em termos de base instalada. A exemplo do servidor Apache, Tomcat também é *freeware*;
  - JCreator: editor de código-fonte Java. Integrado a um pacote JDK, oferece um ambiente de desenvolvimento para a plataforma Windows bastante leve e robusto. Sua principal deficiência é a ausência de uma ferramenta para apoio à prototipação de interfaces;
- não há custo associado à distribuição de cópias de software desenvolvido com o uso da linguagem;
- é possível disponibilizar a aplicação em ambiente Internet, na forma de applet, desde que desenvolvida para tal;
- portabilidade de bytecodes, executáveis em qualquer plataforma para o qual exista máquina virtual implementada;
- suporte à internacionalização (processo pelo qual projeta-se uma aplicação de forma que esta possa adaptar-se a diferentes idiomas sem alterações nos códigos-fonte e, conseqüentemente, sem recompilação da aplicação) [SUN2000a];

- por meio da serialização de objetos [HAL2000a] [HAL2000b], dispensa-se a definição de formatos de arquivo proprietários à aplicação. A serialização é um método pelo qual uma instância em memória é salva de forma persistente (e.g., em disco), podendo ser recuperada em execuções posteriores. A qualquer aplicação que deseje utilizar as instâncias salvas, basta que utilize a classe que gerou a instância. Existe inclusive um esquema para versionamento de classes, de modo que atualizações em uma classe não tornem inviável acessar instâncias serializadas por execuções sobre versões anteriores;
- a cooperação entre uma applet e um ou mais processos servidores, chamados servlets, aparece em CaTLeT na localização da aplicação conforme o idioma da máquina que o executa. Outras opções interessantes podem ser enumeradas (ver sugestões de trabalhos futuros);
- utilização dos layout managers habilita a aplicação a adaptar-se a qualquer resolução de vídeo utilizada em desktops e workstations, com redistribuição planejada das áreas entre os componentes gráficos;
- provê adaptatividade visual às interfaces dos diferentes sistemas operacionais existentes, por meio do *Look And Feel* associado a componentes Swing. Possui, ainda, o seu próprio *Look and Feel* (sua aparência padrão) a qual pode ser uniformemente obtida em qualquer plataforma.

Existe um sentimento enraizado no inconsciente coletivo de que Java ainda não oferece confiabilidade e performance compatíveis com suas pretensões. Considerando nossa falta de experiência em projetos utilizando esta linguagem quando do início do desenvolvimento deste, alguns pontos negativos comumente apontados em Java também foram avaliados.

A comprovação prática de algumas destas vulnerabilidades pode ser enumerada como contraponto às virtudes anteriormente citadas. Nem todas as afirmações, entretanto, procedem. É válido, portanto, efetuar o registro das impressões técnicas e práticas aferidas no curso do projeto, como alerta para as reais dificuldades que o desenvolvimento de software utilizando a linguagem Java traz consigo.

A primeira restrição a ser avaliada quando da adoção de Java como a linguagem de programação de algum projeto é que atualmente é possível desenvolver programas com quatro versões distintas da linguagem (1.0, 1.1, 1.2 e 1.3), sendo os bytecodes compilados sobre uma versão incompatíveis com máquinas virtuais de versões anteriores. Máquinas virtuais Java incorporadas às versões mais recentes dos principais web browsers são capazes de executar applets de todas as versões. No entanto, por exemplo, browsers de quarta geração são capazes de executar apenas applets desenvolvidas com as versões 1.0 e 1.1, exigindo do usuário a instalação de um plug-in (download de 4 a 7 MBytes) ou o upgrade do web browser.

Consequência: desenvolver em Java utilizando sempre a última versão é correr o risco de que grande parcela do parque de máquinas instalado não suporte imediatamente a execução da aplicação.

Outra grande restrição ainda relacionada à linguagem é a de que "Java é lento". Tal afirmação muito preocupava o grupo. Entretanto, as impressões colhidas a partir da execução da applet canadense e de uma aplicação gráfica que utiliza os conceitos de interface hiperbólica desmentiu este mito e comprovou que a performance de Java na manipulação de elementos gráficos é bastante satisfatória. Estimou-se, à época, que os

cálculos associados aos algoritmos categoriais não excederiam a complexidade inerente à manipulação de elementos gráficos. Críticas não-contextualizadas à performance de Java revelaram-se, desta forma, preconceituosas e falsas.

5.6 Técnicas e Tecnologias Utilizadas

O tópico que se inicia detalha algumas soluções técnicas adotadas no desenvolvimento da aplicação. Adicionalmente à finalidade deste trabalho, a principal motivação desta seção relaciona-se a trabalhos futuros que possam vir a ser desenvolvidos, tanto em nível de expansão das capacidades de CaTLeT quanto de projeto e implementação de aplicações que guardem similaridade.

5.6.1 Arquivos Proprietários X Serialização de Objetos

O armazenamento persistente das informações manipuladas representa característica básica das aplicações de computador atuais. Uma ferramenta sem suporte adequado a tal característica é pouco mais do que uma curiosidade. Prova maior de tal afirmação não poderia ter sido dada do que a sugestão emitida por integrante da Banca de Avaliação deste trabalho durante a apresentação do Seminário de Andamento da Semana Acadêmica 2000. Vislumbrando outros usos, sugeriu que fosse incorporada à ferramenta opção para exportação de grafos categoriais em formatos gráficos incorporáveis por processadores de texto.

Não foi incorporada à ferramenta nenhuma possibilidade de exportação dos grafos em formatos manipuláveis por quaisquer outras aplicações. Alguns detalhes sobre como efetuar esta operação, entretanto, podem ser encontrados na referência a Trabalhos Futuros.

Estão presentes em CaTLeT, entretanto, todos os recursos necessários ao armazenamento persistente de categorias por esta manipuladas.

Definir um formato de arquivo específico à aplicação seria tarefa que consumiria tempo de projeto e implementação dedicados ao mesmo. Ainda assim, evoluções pelas quais a aplicação poderia passar no decurso de seu desenvolvimento poderiam determinar alterações neste formato, e consequentemente em todas as rotinas que manipulassem I / O.

Java, entretanto, provê solução altamente flexível, elegante, e além de tudo de utilização bastante simples, denominada serialização de objetos. Os componentes desta solução integram o package *java.io*, e estão presentes desde a versão 1.1 da linguagem.

Serialização de objetos pode ser utilizada com duas finalidades distintas:

- Remote Method Invocation (RMI) - comunicação entre objetos remotos via sockets;
- Persistência - o arquivamento de um objeto para uso em uma invocação posterior da mesma (ou de outra) aplicação.

Os componentes da package *java.io* que provêem a serialização de objetos são:

- Interface Serializable: a classe cujos objetos pretende-se serializar devem implementar esta interface. Seu uso é bastante simples pois esta não apresenta qualquer assinatura de método a ser implementado, possuindo apenas um campo para apoio ao versionamento da classe;
- Classe ObjectOutputStream: por meio de uma instância desta classe o objeto a ser armazenado persistentemente é escrito na saída associada;
- Classe ObjectInputStream: por meio de uma instância desta classe o objeto a ser armazenado persistentemente é lido da entrada associada.

Adicionalmente, cada classe cujos objetos pretende-se serializar pode definir o subgrupo de atributos a persistir. Há duas técnicas providas por Java para implementação de serialização. A forma padrão assume todos os atributos de uma classe declarada Serializable como serializáveis. Atributos que não se deseja serializar devem ser declarados utilizando-se o modificador *transient*.

O único contraponto da serialização apurado durante o desenvolvimento de CaTLeT foi que nem todas as classes definidas serializáveis puderam utilizar a primeira técnica de serialização, por motivo até o momento ignorado. Desta forma, muitas utilizaram o segunda técnica disponível, que consiste na sobrecarga (overriding) dos métodos readObject(ObjectInputStream) e writeObject(ObjectOutputStream). Tal técnica exige que se leia/escreva cada atributo a serializar da/na entrada/saída padrão. Evidentemente que esta é um pouco menos simples do que a primeira técnica.

Se um objeto referencia outros objetos, e tais referências são apontadas como persistentes, então todos os objetos atingíveis a partir do primeiro devem ser escritas simultaneamente, de modo a preservar os relacionamentos existentes. Assim, o método writeObject serializa o objeto especificado, investiga suas referências persistentes a outros objetos recursivamente, e também escreve estes.

CaTLeT utilizou esta técnica para armazenamento persistente. O uso prático desta solução mostrou-se totalmente condizente com os benefícios enumerados pela literatura previamente consultada.

5.6.2 Representação de Morfismos Paralelos

Uma das questões mais delicadas na representação gráfica de grafos por uma aplicação de computador relaciona-se à representação e manipulação de arcos paralelos (arcos com mesmos nodos origem e destino). A ferramenta deve assumir o comando das ações e sugerir a forma dos arcos? Deve sobrepô-los e deixar a cargo do usuário a sua disposição? E quando o diagrama é movido, qual o comportamento que os arcos paralelos devem possuir?

CaTLeT tratou o problema construindo uma classe que traça um paralelo ao conceito de hom-set: a classe CollectionMorfismos. Consiste em agrupar todos os morfismos paralelos em coleções identificadas por seus objetos origem e destino. Os elementos da coleção de morfismos de uma categoria, em CaTLeT, são instâncias da classe CollectionMorfismos.

A classe estende *java.util.TreeSet*, acrescendo à mesma os atributos source, referente ao nodo origem, e target, referente ao nodo destino.

A primeira finalidade desta estrutura é permitir o desenho automático de morfismos paralelos sem que os mesmos se sobreponham, oferecendo comodidade ao usuário da aplicação. Cada morfismo é desenhado segundo um incremento nas coordenadas de seus pontos de inflexão proporcional à ordem que o mesmo ocupa na coleção de morfismos paralelos a que pertence. Desenha-se, desta forma, um feixe de arcos. O incremento alterna-se entre valores positivos e negativos, de forma a produzir um feixe uniforme em torno da reta que une os centros dos dois objetos sobre os quais estão definidos os morfismos.

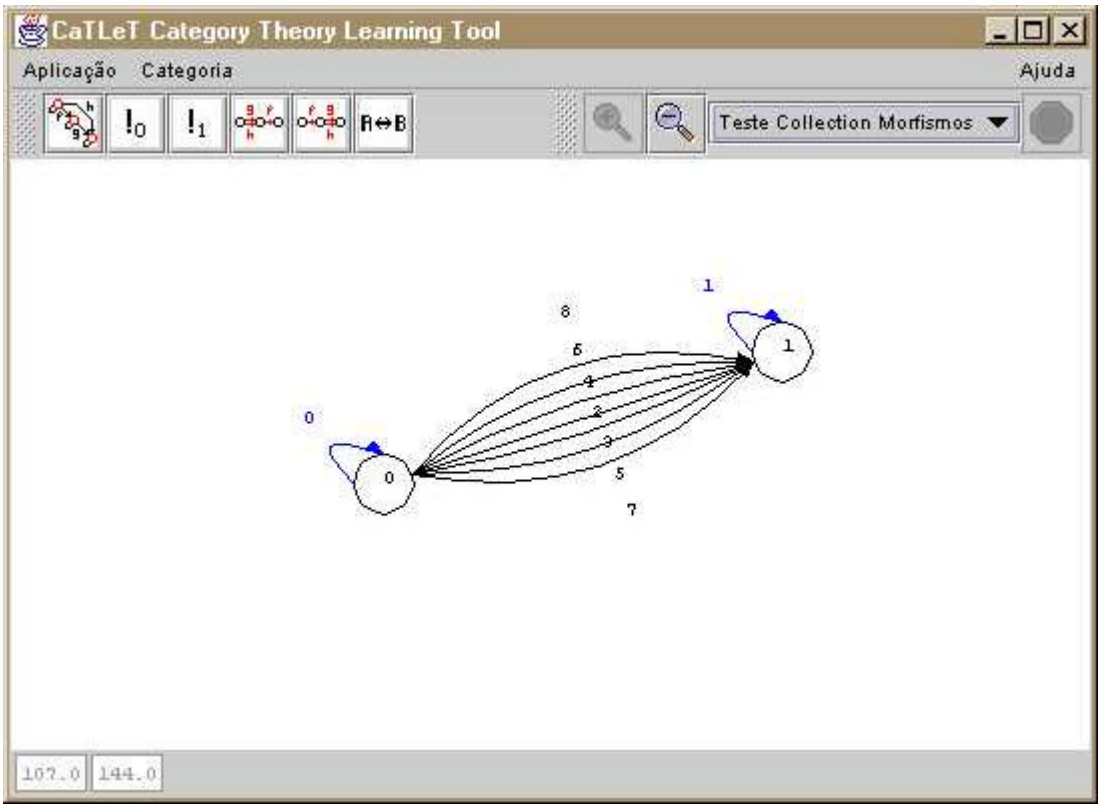


FIGURA 5.7 - Representação de morfismos paralelos em CaTLeT

A segunda finalidade desta estrutura é reduzir o tempo de processamento associado à execução de muitos cálculos categoriais.

Tome-se como exemplo o algoritmo para cálculo de produto. Suponha-se que o produto que se está calculando é binário, sobre os objetos A e B. O produto é um objeto AxB, juntamente com dois morfismos projeção $\pi_1 : AxB \rightarrow A$ e $\pi_2 : AxB \rightarrow B$ tais que, para qualquer objeto pré-produto (candidato a produto), existe um morfismo único tal que o diagrama comuta.

O algoritmo desenvolvido inicialmente determina todos os objetos que são pré-produtos, ou seja, que possuem morfismos projeção para A e B, armazenando estes em uma coleção. Suponha-se, agora, que existam dois algoritmos, um que itera sobre cada morfismo do grafo, e o nosso algoritmo, que itera sobre coleções de morfismos paralelos do grafo. Em um grafo que possua morfismos paralelos, nosso algoritmo apresenta vantagem, pois ao testar se uma coleção de morfismos atende ao requisito de ser projeção do objeto AxB para A ou para B, automaticamente descartam-se todos os morfismos paralelos em um único teste. O algoritmo que testa morfismo a morfismo terá tempo de processamento sempre proporcional à quantidade de morfismos. Nosso algoritmo eventualmente terá tempo de resposta proporcional à quantidade de morfismos, sendo que para categorias com morfismos paralelos será sempre mais eficiente.

5.6.3 Localização da Aplicação

Internacionalização é o processo de projetar uma aplicação de modo que a mesma adapte-se a qualquer idioma e região sem a necessidade de qualquer alteração no código-fonte ou recompilação do mesmo.

Um programa internacionalizado tem as seguintes características:

- a adição de dados específicos a uma localização adapta o mesmo executável à utilização em qualquer parte do mundo;
- elementos textuais, tais como mensagens de status e labels de componentes de interface, não estão codificados diretamente no código-fonte. Ao invés disso, estão armazenados externamente e são acessados dinamicamente, conforme a necessidade;
- suporte a idiomas adicionais não requer compilação;
- formatos de dados culturalmente dependentes, como datas e valores monetários, são apresentados adaptados ao idioma e região do usuário;
- localização torna-se, basicamente, um processo de tradução, podendo ser realizado de forma rápida e sem a necessidade de cooperação com a equipe que desenvolveu a aplicação.

Localização é o processo de adaptação de um software internacionalizado a um idioma ou região específica traduzindo-se textos e adicionando-se componentes específicos ao local. Obviamente, ao desenvolver uma aplicação internacionalizada, necessariamente o software é localizado para um idioma ou região.

As técnicas para internacionalização associadas a aplicações Java *standalone* e applets apresentam muita similaridade. Há um detalhe, porém, que em tempo de execução, quando a aplicação acessa os arquivos de localização, as torna muito diferentes.

Aplicações Java *standalone* tem franco acesso aos recursos do sistema que a está executando. A aplicação pode, por própria conta, acessar os arquivos de que necessita. Applets Java, entretanto, passam por um processo mais longo. A applet necessita obter as configurações regionais da máquina que a está requisitando, transmitir a um processo servlet, e este processo, conforme as configurações regionais recebidas, obtém apenas os arquivos necessários e os transmite à aplicação.

Uma aplicação *web* internacionalizada precisa ser planejada em três pontos:

- como os dados são transmitidos à aplicação *web*;
- como os dados são armazenados pela aplicação *web*;
- como os dados devem ser apresentados ao usuário.

CaTLeT é uma aplicação internacionalizada, desenvolvida para ser executada tanto em modo *standalone* quanto como applet. E, para executar internacionalizada na forma de applet, uma porção da aplicação consistirá em um processo servlet que sirva à applet os arquivos de localização condizentes com as configurações regionais por esta identificada.

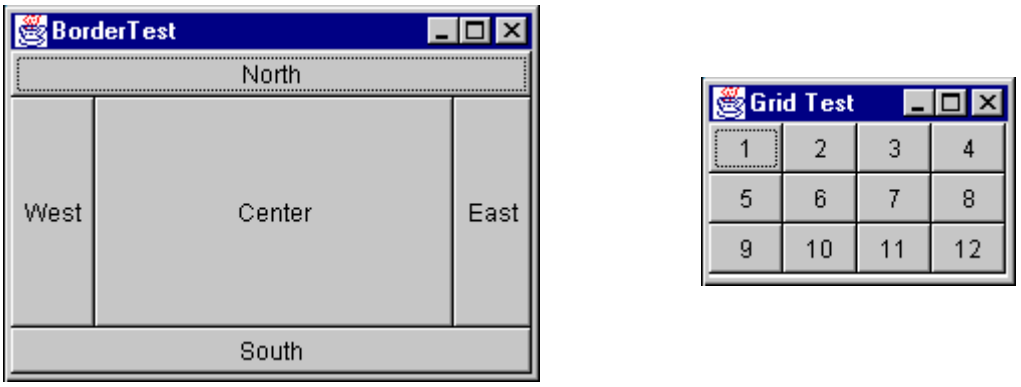
5.6.4 Utilização de Layout Managers

Layout Managers [SCO2000] concedem à aplicação que os utilize adequadamente a adaptatividade a diferentes resoluções de vídeo possíveis, independente de plataforma. Um layout manager encapsula um algoritmo para posicionamento e dimensionamento de componentes GUI.

A função dos layout managers é garantir a cada componente o espaço mínimo a este necessário, estabelecendo uma hierarquia entre os componentes na distribuição da

área destinada à aplicação. Quando um componente de interface do tipo top level container sofre qualquer ação de redimensionamento, os layout managers que este contém redistribuem o novo espaço definido para o container.

Layout Managers podem ser aninhados livremente. Exemplificando, suponha-se uma Window semelhante à de CaTLeT, em que tem-se a área superior reservada a toolbars, concedendo-se o espaço remanescente a um componente de edição de diagramas.



BorderLayout **GridLayout**

FIGURA 5.8 - Aplicações-exemplo de LayoutManagers

Obtém-se o efeito desejado pela combinação de dois layout managers, BorderLayout e GridLayout.

BorderLayout é provavelmente o mais útil dentre todos os layout managers padrão. Define um layout que divide seu container em cinco áreas lógicas. As áreas situadas junto às bordas do container possuem mesmo nível hierárquico, tendo prioridade na aquisição de espaço em relação à área central, que ocupa posição hierárquica inferior.

GridLayout é um layout manager que particiona igualmente a área total de seu container entre todas as células definidas. Por meio de dois parâmetros de seu construtor, define-se a quantidade de linhas e colunas que seu container deve possuir.

Para obter a aparência de DiagramBuilder, a Window principal de CaTLeT, vincula-se à Window um BorderLayout e instancia-se um Panel na região norte da Window, cujo layout é um GridLayout de uma linha e duas colunas. Após isso, basta instanciar o componente para edição de gráficos à região Central do BorderLayout da Window, e cada uma das duas toolbars a uma célula do GridLayout.

5.6.5 Java Foundation Classes

JFC - acrônimo para Java Foundation Classes - engloba um grupo de características destinadas a auxiliar o desenvolvimento de interfaces gráficas com o usuário (GUI). JFC foi anunciada na conferência de desenvolvedores JavaOne de 1997 e sua definição contém as seguintes características:

- define a arquitetura e os componentes da interface Swing;
- Suporte a *Look and Feel* incorporável;
- API de acessibilidade;
- API Java 2D;
- Suporte a *Drag and Drop*.

5.6.6 Collections

Uma das premissas enunciadas para a nova ferramenta definia que a capacidade das estruturas de dados responsáveis pelo armazenamento dos componentes de categorias, bem como a quantidade de categorias possível simultaneamente em uma sessão de trabalho da aplicação, deviam ser limitadas apenas pelo hardware subjacente, e não por restrições internas especificadas no software, como foi apurado nas ferramentas analisadas.

Java provê, em seu pacote `java.util`, um conjunto de classes as quais são logicamente agrupadas, constituindo um framework de *Collections*. Um framework de Collections é uma arquitetura unificada para representar e manipular coleções.

Uma Collection é simplesmente um objeto que agrupa múltiplos elementos em uma única estrutura. Collections tipicamente representam agrupamentos de itens similares (instâncias de uma mesma classe ou com uma superclasse comum).

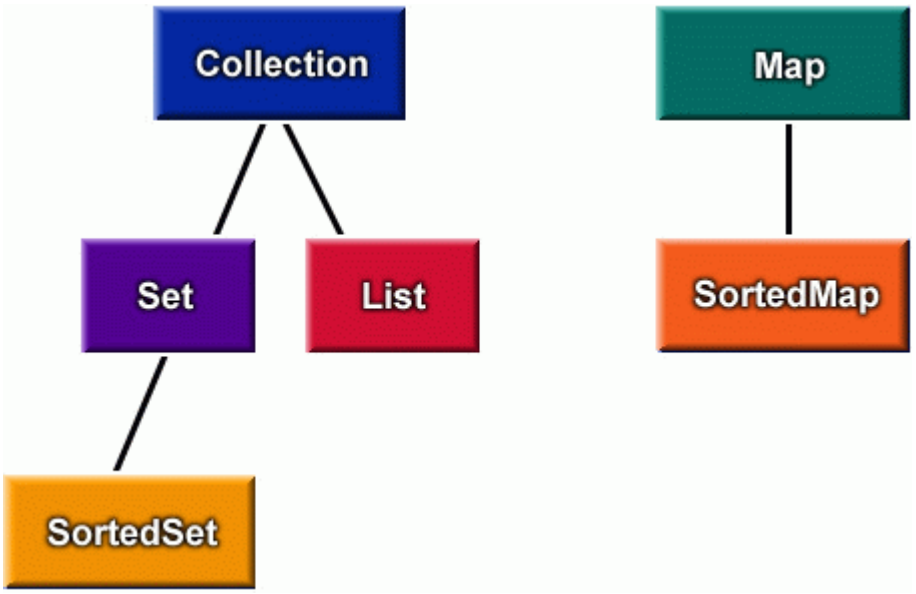


FIGURA 5.9 - Hierarquia de interfaces para o framework de Collections de Java

Uma implementação da interface *Set* é uma classe cujas instâncias são coleções em que não podem ocorrer elementos duplicados. Não há ordenação especificada. Como pode ser percebido, esta interface modela a abstração matemática de conjunto.

Uma implementação da interface *List* é uma classe cujas instâncias são coleções cujos elementos respeitam uma ordenação, sendo permitida a ocorrência de elementos duplicados. Os elementos podem ser acessados diretamente por sua referência posicional.

Uma implementação da interface *Map* é um objeto que associa chaves a valores. Não há duplicatas - chaves só podem mapear no máximo um valor.

Duas razões sustentam a necessidade de uma seção como esta, que apresenta pormenorizadamente as collections.

A primeira justificativa é conseqüência do uso destas estruturas na implementação do suporte a coleções de objetos e morfismos sem limite explícito de capacidade. CatLeT utiliza *TreeSet* como a estrutura para esse fim.

A escolha de CaTLeT por *TreeSets* foi conseqüência de um estudo prévio sobre o framework de collections efetuado logo que iniciou-se seu desenvolvimento (primeiro semestre de 2000). À época, o requisito básico a atender era que as coleções de objetos e morfismos não poderiam aceitar ocorrências duplicadas. Adicionalmente, analisou-se qual estrutura oferecia menor custo associado às operações básicas (inclusão, exclusão e verificação da presença de um elemento na coleção). TreeSet foi a estrutura escolhida pois sua implementação tem a Teoria dos Conjuntos como abstração matemática de base (sem repetições de elementos, portanto), e apresenta custo de acesso da ordem logarítmica para as operações básicas, o que a coloca na classe das estruturas de dados mais eficientes para acesso aleatório.

Infelizmente, pouco proveito pôde ser extraído da performance oferecida por TreeSets. A quase totalidade dos acessos executados por CatLeT às coleções de objetos e morfismos consiste na varredura de todos os elementos. Em algumas situações bastante específicas, entretanto, utiliza-se a verificação da inclusão (método contains()), onde o acesso eficiente de TreeSet traduz-se em vantagem sobre outras estruturas.

A segunda justificativa para esta seção está embasada na necessidade de bom conhecimento das características de todas as estruturas disponíveis no framework de Collections, em virtude de certas necessidades específicas identificadas em algoritmos de cálculo categoriais implementados. Alguns algoritmos utilizaram alguma (até mesmo mais de uma) das implementações de collections como estruturas auxiliares aos cálculos. Certos algoritmos necessitavam que se garantisse certas condições (e.g., ordenação dos elementos), de modo que obrigaram o uso de implementações de collections adequadas à função. Como exemplo, pode-se citar o algoritmo para cálculo de produto e coproduto cuja aridade é livremente arbitrada pelo usuário, exclusividade de CaTLeT.

6 CaTLeT e o ambiente Hyper-Automaton

6.1 Autômatos Finitos: Um Formalismo Para Cursos Na Web

Com a expansão da *WWW,* tornou-se possível oferecer a estudantes um acesso ilimitado a materiais instrucionais de forma independente de tempo e espaço. Entretanto, tais materiais estão dispersos através de inúmeros servidores *Web* e não é fácil encontrar a informação que se procura quando se está aprendendo ou ensinando algum tópico específico. Tornou-se essencial que a Engenharia de *Software* pesquise e desenvolva modelos, linguagens e ferramentas para este novo domínio de aplicação que é a educação e treinamento baseados em tecnologia *Web* a fim de solucionar e contornar problemas como o mencionado anteriormente.

O trabalho Hyper-Automaton apresenta uma forma de modelagem de cursos para o ambiente *Web* utilizando Autômatos Finitos Determinísticos com Saída [MEN98]. 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 Universidade Federal do Rio Grande do Sul. Neste modelo, cursos são autômatos com saída e *links* entre as páginas são transições de um autômato.

A implementação do sistema descrito por Hyper-Automaton tem como base as premissas da criação de uma ferramenta simples, de fácil implementação, manutenção e reuso, independente de outros sistemas (como banco de dados, sistema operacional, etc). Com base nessas premissas, as escolhas de projeto centraram-se na seleção de linguagens de programação suportadas em múltiplas plataformas e na utilização de um sistema de armazenamento de dados com formato interno próprio. O sistema, entretanto, pode ser portado para um banco de dados específico, conforme mudem as necessidades.

Foi adotada uma arquitetura cliente-servidor para o sistema na *WWW*. 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 *WWW* com capacidade de execução de *applets Java.* As páginas *HTML* e material hipermídia em geral estão localizadas no servidor *HTTP* em questão. O sistema não utiliza nenhum editor de *HTML* integrado, podendo o professor utilizar o programa com o qual estiver mais familiarizado.

6.2 Possibilidade de integração de CaTLeT ao ambiente Hyper-Automaton

CaTLeT foi desenvolvida prevendo duas modalidades de execução: standalone e applet. A primeira forma é autocontida e, ao menos no ponto que o projeto se encontra, ainda não prevê interação com outras aplicações. A segunda forma prevista de execução, entretanto, permite que se insira a aplicação em contextos como o ambiente Hyper-Automaton.

O uso de applets em web sites é recurso tecnológico muito simples e já bastante difundido. Sua estréia na *WWW* ocorreu em 1995, quando do lançamento da versão 2.0 do web browser Netscape.

Iniciar um programa desenvolvido para executar na forma de applet consiste em parametrizar uma *tag* HTML desenvolvida especificamente para incorporar a estes scripts tal funcionalidade.

O ambiente Hyper-Automaton prevê que cursos na Web são autômatos finitos com saída. Todo conteúdo é armazenado na forma de pequenos scripts HTML, os quais são fornecidos e combinados pelo instrutor.

Uma vez que CaTLeT pode ser executado na forma de applet, e a chamada a uma applet consiste simplesmente de uma tag HTML, pode-se armazenar no ambiente Hyper-Automaton unidades instrucionais para cursos de Teoria das Categorias que são apenas tags parametrizadas para a execução de CaTLeT.

Existe, por parte do grupo de pesquisa em Teoria das Categorias da UFRGS, o plano de publicar no ambiente Hyper-Automaton uma versão de [MEN2001]. Desde o seu início, o grupo antevê CaTLeT como importante ferramental de apoio a essa publicação.

7 Conclusão e Trabalhos Futuros

Este capítulo final da dissertação está dividido em duas seções, que apresentam as conclusões do trabalho realizado e a definição de trabalhos futuros que podem vir a dar continuidade aos resultados obtidos.

7.1 Conclusão

Teoria das Categorias, no breve período compreendido entre a publicação de seus preceitos e os dias atuais, já foi capaz de atrair interesse das comunidades científica e acadêmica por seu potencial de aplicabilidade em Ciência da Computação. A difusão da Teoria em ritmo condizente com suas qualidades, entretanto, ainda esbarra na própria juventude da mesma, por falta de recursos bibliográficos e material humano.

Este trabalho apresentou uma alternativa às dificuldades enfrentadas, propondo, especificando e efetivamente implementando uma aplicação de computador para a representação gráfica de categorias, dotada de opções para execução de alguns dos cálculos e operações desta teoria considerados básicos segundo a visão dos Grupos de Pesquisa em Teoria das Categorias da UFRGS e da PUC/RJ.

Suporte computacional à representação e operações categoriais não constitui tema mundialmente inédito. O grupo de pesquisa em Teoria das Categorias da Universidade de Mount Allison (Canadá) já apresentou resultados desta natureza, com seus softwares Category Construction Program (aplicação C ANSI com interface caracter) e Category Theory Database Tools (applet Java). Ambas são produto da mesma linha de experiência.

A preexistência destas avalizou a intenção e convicção do grupo de pesquisa local sobre a real viabilidade e grau de servilidade de um trabalho dessa natureza. Adicionalmente, foi possível desenvolver discussões em torno do tema com base em análises pormenorizadas da experiência canadense e seus softwares.

Algumas características do experimento aqui descrito diferenciam o produto final deste, a aplicação computacional CaTLeT, de suas antecessoras. Muito do esforço empreendido na execução deste trabalho está relacionado às três características citadas abaixo, que conjuntamente com a súmula de conceitos adotada, constituem as diretrizes deste projeto:

- a interação do usuário com a aplicação foi planejada visando o máximo aproveitamento de uma das mais relevantes características de Teoria das Categorias, que é a possibilidade de representação de categorias, diagramas e propriedades categoriais na forma de grafos. A possibilidade de fornecer e manipular todos os componentes de uma categoria diretamente por meio do editor de gráficos embutido em CaTLeT torna a sua utilização muito simples e de fácil aprendizado e utilização, quando comparada com suas predecessoras;
- é possível com a ferramenta construir diagramas sobre grafos representativos de categorias a partir de multiconjuntos de objetos e morfismos componentes da categoria. Cálculos que só fazem sentido quando efetuados sobre diagramas podem, desta forma, ser implementados e disponibilizados;
- o rigor formal utilizado na modelagem da interação homem-máquina inviabilizou a construção de grafos e diagramas categoriais mal-formados. Conseqüentemente,

cálculos sobre categorias e diagramas só ocorrem sobre estruturas conceitualmente válidas.

Reitera-se aqui que o grupo de pesquisa em Teoria das Categorias do Instituto de Informática da UFRGS ambiciona que esta ferramenta desenvolvida se torne um agente catalisador na propagação, não apenas de Teoria das Categorias, mas conjuntamente da súmula de conteúdos proposta para o contato inicial de estudantes de qualquer área com a mesma.

A única tarefa pretendida no escopo deste projeto que não pôde ser executada consistia em uma validação integrada da funcionalidade e usabilidade de CaTLeT junto a alunos de graduação. Este laboratório estava previsto para ocorrer no semestre 2001/2, inserido no semestre letivo de alunos regularmente matriculados em turmas da disciplina *Categorias Computacionais*. A greve deflagrada nas instituições públicas brasileiras de ensino técnico e superior no semestre citado inviabilizou temporariamente o atendimento desta meta, pelo menos em tempo de aferir e tabular os resultados para serem anexados a esta dissertação.

7.2 Trabalhos Futuros

CaTLeT foi desenvolvida com cronograma e objetivos definidos. Este cronograma foi posteriormente estendido, de modo que os objetivos originalmente almejados, assim como outros identificados no curso do projeto, fossem atingidos.

O plano de Estudos e Pesquisa (PEP) que pautou a execução deste trabalho dividia os objetivos em dois grupos. O primeiro grupo deveria necessariamente ser contemplado por este trabalho. Havia ainda um segundo grupo de necessidades, tidas como desejáveis. Este segundo grupo, entretanto, seria tratado dentro da eventual sobra de tempo reservado ao projeto, sendo tudo o que não fosse desenvolvido devidamente relacionado como possibilidade para trabalhos futuros. Assumiu-se, desta forma, um compromisso adequado ao tempo disponível, sem que se perdesse a visão inicial do todo a ser buscado.

Os objetivos tidos como imprescindíveis foram plenamente cobertos. Adicionalmente, algumas características apenas tidas como desejáveis também foram projetadas e incorporadas. Muitos dos objetivos enumerados como desejáveis no Plano de Estudos e Pesquisa, entretanto, não foram agregados a CaTLeT. Pode-se apontar os seguintes fatos como justificativas para tal:

- a necessidade de incorporação de algumas características que originalmente não haviam sido previstas, e.g., a possibilidade de manipulação de diagramas sobre categorias;
- a exigência de dedicação do mestrando a suas atividades e compromissos profissionais ao longo de todo o curso por vezes demandou tempo e esforço bastante superior ao originalmente previsto.

Como decorrência, relacionam-se os objetivos complementares originalmente listados no PEP como pontos de partida para futuros trabalhos correlatos:

- prover a construção de categorias sobre categorias, ou seja, categorias cujos objetos sejam outras categorias;
- o atendimento do tópico anterior pode ser visto como a inserção, na ferramenta, do conceito de subobjeto, pois os objetos não serão mais entes atômicos para a

ferramenta. O conceito de subobjeto, dessa forma, também é considerado importante;
- Transformações Naturais;
- Exponenciação;
- suporte à análise de categorias estudadas e conhecidas, tanto finitas quanto infinitas, como Set, Poset, Pfn, Mon e monóides finitos / livremente gerados vistos como categorias.

A possibilidade levantada nas discussões do grupo de apresentar, por meio da construção animada em modo passo a passo, as correções necessárias em um grafo para que o mesmo atenda à definição de categoria, teve sua infraestrutura básica projetada (seção 6.4.2), no entanto a idéia não foi totalmente desenvolvida. Esta funcionalidade claramente representa incremento ao processo de aprendizado, justificando-se plenamente sua incorporação futura a CaTLeT.

Certos cálculos nem sempre possuem como resposta um resultado (e.g. pode não haver produto entre dois ou mais objetos). Uma possibilidade interessante seria dotar a ferramenta da capacidade de sempre fornecer respostas a certos cálculos. Para isso a ferramenta, assim que constatasse que não há resultado para o cálculo solicitado, "completaria" o grafo ou diagrama com objetos e morfismos conforme a necessidade.

O enfoque predominante de CaTLeT foi sobre categorias e seus conceitos. De fato, pouco tempo foi dedicado ao tratamento de funtores. Com certeza muito pode ser imaginado e incorporado a uma ferramenta.

O tratamento de funtores pode ser ampliado para além do que foi efetivamente incorporado a CaTLeT. A possibilidade de definir um funtor entre duas categorias discretas pode beneficiar-se amplamante, e.g., de algum tipo de visualização tridimensional, em que se poderia visualizar as categorias inscritas em planos sob um ângulo perspectivo, e as setas constituintes do funtor entre as mesmas desenhados entre estes planos.

Pode-se enumerar também uma melhoria de caráter puramente tecnológico, que ainda assim constitui recurso de usabilidade de grande relevância. Recentemente, a Sun Microsystems disponibilizou uma package para a linguagem Java destinada a prover tratamento avançado de imagens, o *Java Advanced Imaging* (JAI) [SUN2001]. Entre outras funcionalidades, esta package permite a conversão de imagens geradas a partir da utilização das APIs 2D e 3D da linguagem para formatos gráficos largamente suportados, como BMP, JPEG e GIF. Tal package está disponível a partir da versão 1.3 de Java.

Outra possibilidade bastante interessante para uma ferramenta que se destina a apoiar computacionalmente o processo de ensino / aprendizado em frameworks de EAD voltados a Teoria das Categorias está relacionada à modelagem cognitiva e seleção de estratégias de ensino a serem utilizadas na tentativa de promover auxílio personalizado ao aluno.

CaTLeT pode evoluir para um programa educacional modalidade STI (Sistema Tutor Inteligente). Os STI são programas de computador com propósitos educacionais e que incorporam técnicas de IA, e derivam dos programas CAI (Computer

Assisted/Aided Instruction). STI oferecem vantagens em relação a CAI pelo fato de simularem o processo de pensamento humano. [SCH2000]

Será necessário, para tanto, que a arquitetura da aplicação evolua e incorpore outros módulos. Um módulo, por exemplo, poderia destinar-se ao gerenciamento de perfil dos alunos. Este módulo dotaria a ferramenta de comportamento evolucionário. Cada nova sessão de estudo que se iniciasse assumiria configuração adequada ao aluno corrente.

Bibliografia

[ALU2001] ALUR, Deepak; CRUPI, John; MALKS, Dan. **Core J2EE Patterns – Best Practices and Design Strategies**. Palo Alto: Sun Microsystems Press, 2001. 459 p.

[ASP91] ASPERTI, Andrea; LONGO, Giuseppe. **Categories, Types and Structures - An Introduction to the Working Computer Science, Foundations of Computing**. Massachusetts: MIT Press, 1991.

[BAC95] BACELAR, Carlos. **Introdução à Teoria das Categorias**. Portugal: Departamento de Informática, Escola de Engenharia, Universidade do Minho, 1995. Relatório Técnico.

[BAR90] BARR, Michael; WELLS, Charles. **Category Theory for Computing Science**. New York: Prentice Hall, 1990.

[BOE78] BOEHM, B. et al. **Characteristics of Software Quality**. New York: Elsevier, 1978.

[BRA2000] BRADBURY, J.; ROSEBRUGH, R. **Graphical Database for Category Theory**. Canadá: Mount Allison University, 2000. Disponível em: <http://mathcs.mta.ca/research/rosebrugh/gdct/>. Acesso em: jan. 2000.

[CAT98] CATTELL, R.G.G. et al. **The Object Database Standard**: ODMG 2.0. San Francisco: Morgan Kaufmann, 1998.

[CON86] CONTE, S. D. et al. **Software Engineering Metrics and Models**. California: Benjamin/Cummings, 1986. 460 p.

[COS99] COSTA, Simone André; MACHADO, Júlio Pereira; BLAUTH, Paulo. Teoria das Categorias: Experiência e Proposta de Ensino. In: SIMPÓSIO BRASILEIRO DE INFORMÁTICA NA EDUCAÇÃO, 10., 1999, Curitiba. **Anais...** Curitiba, SBIE, UFPR, 1999. p. 389 - 391.

[DIV99] DIVÉRIO, Tiarajú Asmuz; MENEZES, Paulo Blauth. **Teoria da Computação**: Máquinas Universais e Computabilidade. Porto Alegre: Sagra-Luzzato, 1999. 205 p.

[EIL45] EILENBERG, S.; MAC LANE, S. General Theory of Natural Equivalences. **Trans. Amer. Math. Soc.**, [S.l.], v. 58, p. 231 - 294, 1945.

[FLE95] FLEMING, M.; GUNTHER, R.; ROSEBRUGH, R. **A Database of Categories**. Canadá: Mount Allison University, 1995. Disponível em <http://cs.mta.ca/research/rosebrugh/dbc/>. Acesso em: ago. 1999.

[FOW97] FOWLER, Martin. **UML Distilled**: applying the standard object modeling language. Reading: Addison Wesley, 1997.

[GON98] GONG, Li. **Java Security Architecture (JDK1.2)**. 1998. Disponível em: <http://java.sun.com/products/jdk/1.2/docs/guide/security/spec/security-spec.doc.html>. Acesso em: fev. 2000.

[HAL2000a] HALLOWAY, Stuart. **Serialization in the Real World; Serialization and Class Versioning; Serialization and Secure Data; Serialization and the Complete Class Rewrite**. 29 feb. 2000. Disponível em: <http://developer.java.sun.com/developer/TechTips/2000/tt0229.html>. Acesso em: 29 fev. 2000.

[HAL2000b] HALLOWAY, Stuart. **Improving Serialization Performance with Externalizable**. 25 apr. 2000. Disponível em: <http://developer.java.sun.com/developer/TechTips/2000/tt0425.html>. Acesso em: 27 abr. 2000.

[HIX93] HIX, D.; HARTSON, H.R. **Developing User Interfaces**. New York: John Wiley & Sons, 1993.

[IMS98] IMS Project. Disponível em: <http://www.imsproject.org>. 1998. Acesso em: set. 1999.

[LEO95] LEONEL, Neron Arruda. **Teoria dos Grafos & Análise Combinatória**. Porto Alegre, 1995. Notas de Aula.

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

[MAC98] MACHADO, Júlio Pereira; MENEZES, Paulo Blauth. Sistemas de Gerenciamento para Ensino a Distância. In: CONGRESSO INTERNACIONAL DE EDUCAÇÃO À DISTÂNCIA, 5., 1998, São Paulo. **Anais...** São Paulo: ABED, 1998. Disponível em: <http://www.abed.org.br/artigos2/artigos/24/sgead971.html>. Acesso em: set. 1999.

[MAC2000] MACHADO, Júlio Pereira. **Hyper-Automaton**: hipertextos e cursos na Web usando autômatos finitos com saída. 2000. Dissertação (Mestrado em Ciência da Computação) - Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[MEC2001] BRASIL. Ministério da Educação e Cultura. **Diretrizes Curriculares do Ministério da Educação e Cultura para cursos de Computação e Informática**. Brasília, 2001. Disponível em: <http://www.mec.gov.br/sesu/ftp/curdiretriz/computacao/co_diretriz.rtf>. Acesso em: nov. 2001.

[MEN98] MENEZES, Paulo Fernando Blauth. **Linguagens Formais e Autômatos**. 2.ed. Porto Alegre: Sagra Luzzatto, 1998. 168 p.

[MEN2000] MENEZES, Paulo Fernando Blauth; TOSCANI, Laira Vieira; DIVÉRIO, Tiarajú Asmuz; RIBEIRO, Leila; ZENI, Loiva C.; GONZALEZ, Medianeira. Proposta de Plano Pedagógico para a Matéria Matemática. In: WEI - WORKSHOP SOBRE EDUCAÇÃO EM INFORMÁTICA, 2000, Curitiba. **Anais...**. Curitiba – PR: Ed. Universitária Champagnat, 2000. p. 65 –102.

[MEN2001] MENEZES, Paulo Fernando Blauth; HAEUSLER, E. H. **Teoria das Categorias e Ciência da Computação**. Porto Alegre: Instituto de Informática da UFRGS: Sagra Luzzatto, 2001. 324 p.

[MOU2001] MOURÃO, Tiago. **SimCat**. 2001. Trabalho Final de Graduação (Bacharelado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[OWS97] OWSTON, Ronald D. The World Wide Web: A Technology to Enhance Teaching and Learning? **Educational Researcher**, Washington DC, v. 26, 1997.

[PFE2000] PFEIFF, Fábio V. **Análise de Ferramentas de Apoio ao Ensino / Aprendizado de Teoria das Categorias**. 2000. Trabalho Individual (Mestrado em Ciência da Computação) – Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[PIA92] PIAGET, Jean. **Morphisms and Categories**: comparing and transforming. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 1992.

[PIE91] PIERCE, Benjamin C. **Basic Category Theory for Computer Scientists**. Massachusetts: MIT Press, 1991. 100 p.

[ROS98] ROSEBRUGH, R.; BRADBURY, J. **Category Theory Database Tools**. Canadá: Mount Allison University, 1998. Disponível em: <http://tac.mta.ca/mathcs/javasource/index.html>. Acesso em: nov. 2001.

[SBC97] SOCIEDADE BRASILEIRA DE COMPUTAÇÃO. **Currículo de Referência para Cursos de Graduação em Computação**. Disponível em: <http://www.dcc.unicamp.br/~ranido/gt1-sbc/cr97_bigonha.html>. Acesso em: mar. 2001.

[SCH2000] SCHUCK, Pedro Willibaldo; GIRAFFA, Lúcia Maria Martins. **MATFIN** - Um Assistente Inteligente para Suporte ao Ensino de Matemática Financeira. In: SEMINÁRIOS DE ANDAMENTO, 2000, Porto Alegre. **Anais...**. Porto Alegre: PUCRS, 2000. p. 57 - 67.

[SCO2000] STANCHFIELD, Scott. **Effective Layout Management Short Course**. 2000. Disponível em: <http://developer.java.sun.com/developer/onlineTraining/GUI/AWTLayoutMgr/shortcourse.html>. Acesso em: ago. 2000.

[SUN2000a] SUN MICROSYSTEMS. **Java Tutorial**: a practical guide for programmers. Disponível em: <http://java.sun.com/tutorial>. Acesso em: nov. 2001.

[SUN2000b] SUN MICROSYSTEMS. **Java Platform 1.3 API Specification**. Disponível em: <http://java.sun.com/j2se/1.3/docs/api/index.html>. Acesso em: nov. 2001.

[SUN2001] SUN MICROSYSTEMS. **Java Advanced Imaging API**. Disponível em: <http://java.sun.com/products/java-media/jai/index.html>. Acesso em: jul. 2001.

[WAL91] WALTERS, R. F. C. **Categories and Computer Science**. Cambridge: Cambridge University Press, 1991.

[WEB98] WEBER, Taisy et al. Uma Experiência com Hiperdocumentos e Internet no Suporte a Disciplinas de Computação. In: WORKSHOP SOBRE EDUCAÇÃO EM INFORMÁTICA, 6.; CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 18., 1998, Belo Horizonte. **Anais...** Belo Horizonte: SBC, UFMG, 1998. p.532-545.