UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

UM ESTUDO DA LÓGICA LINEAR COM APLICAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE GRADUAÇÃO

Felipe de Lima Athayde

Santa Maria, RS, Brasil

2010

UM ESTUDO DA LÓGICA LINEAR COM APLICAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

por

Felipe de Lima Athayde

Trabalho de Graduação apresentado ao Curso de Ciência da Computação
da Universidade Federal de Santa Maria (UFSM, RS), como requisito
parcial para a obtenção do grau de
Bacharel em Ciência da Computação

Orientador: Profª Drª Juliana Kaizer Vizzotto

Trabalho de Graduação Nº307
Santa Maria, RS, Brasil

2010

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada,
aprova o Trabalho de Graduação

UM ESTUDO DA LÓGICA LINEAR COM APLICAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

elaborado por
Felipe de Lima Athayde

como requisito parcial para obteno do grau de
Bacharel em Ciência da Computação

COMISSO EXAMINADORA:

Profª Drª Juliana Kaizer Vizzotto
(Presidente/Orientador)

Prof Dr Giovani Rubert Librelotto (UFSM)

Profª Drª Iara Augustin(UFSM)

Santa Maria, 15 de dezembro de 2010.

AGRADECIMENTOS

O presente trabalho é produto de muito esforço e estudo. Sei que há muitos pontos a serem melhorados e o farei no decorrer de minha vida. Mais do que uma produção como requisito para conclusão da graduação em Ciência da Computação, este trabalho é resultado da concepção que tenho das coisas no momento atual de minha vida. Como parte do progresso natural das coisas, minha concepção evoluirá, verei que o que acreditava ser verdadeiro não mais o é ou nunca o foi e novos conhecimentos virão.

Com o término deste trabalho, deixo meus sinceros agradecimentos a todos os meus amigos. Amigos, pois não há expressão mais abrangente e precisa para designar a relação afetiva que me une a todos que participaram da minha vida e pelos quais guardo carinho fraternal.

Nos quatro breves anos que compreendem uma graduação, é incrível o quanto nos é imposto aderir a um ambiente colaborativo, talvez o caráter mais espontâneo e invariável da vida. A força empregada por cada pessoa na nossa vida, com sua intensidade e direção, formam uma resultante quando devidamente somadas. Essas ações conjugadas ditam aonde chegaremos, onde estaremos em tantos anos, o quanto impulsionaremos a vida de outros e, consequentemente, o estado do mundo. Com tantas pessoas envolvidas em minha vida, direta ou indiretamente, muitas foram as forças atuando de modo a favorecer a completude de minha graduação.

Agradeço a Deus, força principal que atua permanentemente em favor do progresso universal.

Agradeço à minha família, pela permanente atenção e auxílio. Devo a todos da minha família o meu desenvolvimento pessoal, meus valores e minha conduta.

Agradeço à secretária do curso, Janice, por todos os esforços despendidos em meu favor. Tal auxílio não apenas contribuiu, mas permitiu que eu vencesse alguns obstáculos que pareciam intransponíveis. Além disso, é admirável o quanto além de suas funções esta pessoa, sempre simpática e sorridente, se colocou para auxiliar os alunos.

Agradeço aos meus colegas de curso pela companhia, auxílio e aprendizado. Alguns destes contribuíram bastante para que este trabalho fosse realizado.

Agradeço aos professores do curso que realmente cumpriram com seu dever. Também aos professores que compuseram minha banca, por todo auxílio no presente trabalho. Agradeço também ao coordenador do curso que demonstrou muita responsabilidade e competência no exercício de suas funções, refletindo o sucesso de suas ações na qualidade de nossa graduação.

Devo a escolha do tema do presente trabalho a uma das mentes mais brilhantes com as quais pude ter contato, através de seus textos, aulas e programas de rádio. É ao contato com essa grande inteligência que devo meus esforços de ascensão e esta enorme mudança de direção na vida. Agradeço ao professor Olavo de Carvalho.

“Somente a consciência individual do agente dá testemunho dos atos sem testemunha, e não há ato mais desprovido de testemunha externa do que o ato de conhecer.”

— Olavo de Carvalho

RESUMO

Trabalho de Graduação
Curso de Ciência da Computação
Universidade Federal de Santa Maria

UM ESTUDO DA LÓGICA LINEAR COM APLICAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Autor: Felipe de Lima Athayde
Orientador: Profª Drª Juliana Kaizer Vizzotto
Local e data da defesa: Santa Maria, 15 de dezembro de 2010.

A ciência da computação se ocupa da automatização do raciocínio para solucionar problemas e auxiliar no exercício de determinadas atividades. Nesta área, as lógicas clássica e intuicionista são muito utilizada no desenvolvimento de algoritmos que atendam às necessidades do mundo real. Porém, há restrições que não são previstas pelos sistemas lógicos clássicos, como a escassez de recursos à disposição neste mundo. Os sistemas lógicos usuais lidam com verdades gerais e imutáveis. A lógica linear, por outro lado, foi desenvolvida para permitir a produção com base nas questões de realismo e preservação de recursos, tão comuns no mundo construtivista em que se vive. Tais conceitos possuem grande receptividade em áreas computacionais como inteligência artificial, sistema de tipos, interpretação abstrata e computação paralela. Este trabalho apresenta um estudo da lógica linear com aplicação em ciência da computação e com algum contraste com outros sistemas lógicos. Primeiro será discutida a lógica linear como linguagem formal e seu sistema de dedução natural. Depois, para demonstrar como a lógica linear pode ser aplicada na ciência da computação, será utilizada uma aplicação baseada em sistemas concorrentes, um dos grandes focos da lógica linear, desenvolvida para ilustrar uma aplicação desenvolvida de forma linear.

Palavras-chave: Lógica linear, ciência da computação, aplicação.

ABSTRACT

Undergraduate Final Work
Undergraduate Program in Computer Science
Universidade Federal de Santa Maria

A LINEAR LOGIC STUDY WITH APPLICATION IN COMPUTER SCIENCE

Author: Felipe de Lima Athayde
Advisor: Prof[a] Dr[a] Juliana Kaizer Vizzotto

Computing science deals with automation of reasoning for solving problems and assisting in certain activities. In this area, both logic and intuitionistic logic plays a fundamental role. They are widely used for developing algorithms to meet the needs of real world. However, there are constraints that classical logic systems can not overcome, considering our world deals with scarcity of resources. The ordinary logical systems deal with general and immutable truths. The linear logic, on the other hand, was developed to capture the realist and resource-based reasoning, so common in the constructivist world we live in. These concepts have great acceptance in computing science areas such as artificial intelligence, type systems, abstract interpretation and parallel computing. This paper presents a study of linear logic with application in computing science with some contrast with other logical systems. First we discuss linear logic as formal language and its natural deduction system. Then a computing application, based on concurrent systems, one of the major focus of linear logic, was developed to demonstrate how one can represent reasoning in a linear fashion.

Keywords: linear logic, computer science, application.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

O ser humano, como um dos mais aptos usuários do intelecto, tem seu comportamento regido por relações lógicas. Ainda mais, tanto no processo cognitivo quanto no de produção de conhecimento, tenta-se utilizar, ainda que inconscientemente, de elementos lógicos que coletados no decorrer da evolução intelectual individual. O estudo de lógica pode ser sumarizado como o estudo do modo correto de raciocinar (GENSLER, 2002).

A utilização com rigor da lógica é adotada principalmente em áreas científicas, onde as construções devem ser sistemáticas de forma a garantir a unidade do saber e garantir a forma correta do raciocínio. Dentre as áreas científicas, a Ciência da Computação surgiu principalmente para auxiliar o ser humano no exercício de atividades já realizadas anteriores ao advento tecnológico e permitir que se avançasse em direção a soluções que não eram possíveis até então.

Na ciência da computação, a lógica formal é fundamental, pois é através dela que se mantém o rigor matemático necessário para dar confiabilidade à área. É através da utilização correta da lógica que se pode desenvolver linguagens para modelar situações encontradas para depois analisá-las formalmente (HUTH; RYAN, 2004). Outras áreas onde a lógica é bastante utilizada é a de interpretação abstrata (ANDREOLI; CASTAGNETTI; PARESCHI, 1993), sistema de tipos (BAILLOT, 2007; BAILLOT; HOFMANN, 2010; XI; ZHU; LI, 2004; SHI; XI, 2009), determinação de complexidade (GROHE, 1998; FREIRE, 2008) e outras. Pode-se consultar (HALPERN; HARPER, 2001) para uma abordagem aprofundada no assunto.

O termo lógica *per se* remete diretamente à lógica clássica, mas existem diversos tipos de lógicas que podem se mostrar convenientes dependendo das necessidades e da circunstância em questão. Dentre os muitos sistemas lógicos que são utilizados em Ciência da Computação, são alguns deles: lógica clássica, lógica proposicional, lógica intuicionista,

lógica de predicados e lógica linear. Porém, o conhecimento de outros tipos de lógica alheios à convencional não costuma ser amplamente explorado em muitos dos maiores cursos de Ciência da Computação no Brasil, como pode ser constatado através da análise da ementa destes cursos (UFRGS, 2010; UFPEL, 2010; PUC-RIO, 2010; UFRJ, 2010; UFSM, 2010; USP, 2010; UNICAMP, 2010).

O sistema de provas da lógica é focado em sistemas formais, como: cálculos de predicado intuicionistas, cálculo de predicado clássico, aritmética. A lógica intuicionista considera o que se pode obter dada uma certa informação, enquanto a Ciência da Computação se preocupa em como produzir tal resultado através do que se é fornecido. Em dado momento, ficou evidente que a linguagem de programação era uma espécie de lógica e a lógica era uma espécie de linguagem de programação. A isto se deu o nome de isomorfismo de Curry-Howard (XAVIER; OLIVEIRA, 2009).

Em um mundo dinâmico, em cima do qual a Ciência da Computação deveria trabalhar, a lógica clássica oferece um estaticismo e trabalha em cima de situações que nem sempre podem ser obtidas no mundo real. A lógica linear, por outro lado, permite abordar os problemas de forma mais dinâmica e coerente, respeitando os limites impostos pelo mundo real e oferecendo um sistema de provas construtivo (LINCOLN, 1992; GIRARD, 1995).

Neste cenário, este trabalho busca estudar lógica linear, como algo que vem se mostrando de grande importância no meio científico, e demonstrar sua aplicabilidade em um programa desenvolvido para fins didáticos. Para este fim, desenvolveu-se uma aplicação que simula um gerenciador de máquinas virtuais, em um sistema que dispõe de recursos finitos. Em dado momento, serão confrontados os sistemas lógicos habituais, clássico e intuicionista, e a lógica linear. Através deste trabalho, será possível compreender em parte a essência da lógica linear e seus operadores quando postos em prática em uma aplicação.

1.1 Justificativas

A Ciência da Computação pode ser vista como a automatização do processamento da informação através da utilização de modelos numéricos. De forma semelhante, utiliza-se do raciocínio para processar as informações absorvidas ou produzidas através da atividade intelectual e é a lógica que permite aproximar tais movimentos intelectuais de sua aplicação computacional. Sendo assim, é evidente a importância da lógica para esta área

científica e é imperativa a fluência dos profissionais dessa área neste sentido, para garantir melhor rendimento e maior confiabilidade nas suas produções.

A lógica linear, com suas características construtivas, oferece uma grande aproximação com o mundo real e seus recursos limitados. Com a expansão de áreas como processamento paralelo, inteligência artificial, otimização de algoritmos e sistema de tipos, a lógica linear tem trazido grande entusiasmo para os cientistas da computação, os quais reconheceram a lógica linear como uma lógica expressiva e poderosa, com conexões profundas com a ciência da computação (LINCOLN, 1992).

Na área de inteligência artificial, a lógica linear oferece soluções bastante intuitivas para problemas como o das Torres de Hanoi e o cenário dos Blocos (POWER; WEBSTER, 2001). Outra grande vantagem da lógica linear, é que utilizando este sistema, não há necessidade de utilizar *garbage collector*, contagem de referências e se pode utilizar com segurança atualização destrutiva de vetores (WADLER, 1990). Na área de processamento paralelo, enquanto um processo está utilizando algum recurso os outros processos devem aguardar até que este recurso esteja disponível novamente. Como uma alternativa à lógica linear, este trabalho visa a correta compreensão da lógica linear, o que pode ser de grande valia para o meio científico nessas e em muitas outras áreas de aplicação.

Apesar da importância da lógica, seu ensino não vem sendo tão enfatizado quanto deveria ser nos cursos de Ciência da Computação no Brasil, como pode-se observar na grade curricular de alguns dos maiores cursos de Ciência da Computação do país. O trabalho direto com lógica em algumas das maiores universidades do país acaba sendo abordado diretamente em poucas matérias (UFRGS, 2010; UFPEL, 2010; PUC-RIO, 2010; UFRJ, 2010; UFSM, 2010), quando nenhuma (USP, 2010; UNICAMP, 2010), com exceção da (UFPE, 2010), que oferece um *curriculum vitæ* bastante amplo no campo da lógica. A abrangência do tema, portanto, fica bastante limitada devido à amplitude do tema e à limitação temporal. O estudo da lógica, portanto, acaba ficando distribuído entre outras cadeiras ofertadas, nenhuma, porém, segundo a ementa das universidades consultadas, abordando a lógica linear. Sendo assim, este trabalho visa expandir os conhecimentos acerca da lógica, com foco na lógica linear, e permitir projeções de trabalhos relacionados.

1.2 Objetivos

1.2.1 Objetivo geral

O propósito deste trabalho é estudar lógica linear e sua aplicação prática com base em um referencial teórico. Será utilizada uma aplicação desenvolvida para demonstrar como pode ser encarado o foco em recursos deste sistema lógico. Desta forma, almeja-se analisar de que forma podem ser formuladas resoluções algorítmicas lineares para problemas computacionais. A partir daí, pode-se analisar se o uso da lógica linear poderia representar alguma vantagem pela aproximação maior com a área em questão. Assim, busca-se aperfeiçoar a área da qual esta ciência derivou e, desta forma, tornar mais acessível a produção científica.

1.2.2 Objetivos específicos

Nessa seção encontram-se os objetivos específicos que serão abordados no decorrer do desenvolvimento deste estudo:

- Conceituação dos elementos fundamentais: razão, lógica e ciência;
- Estudo de Lógica Formal, suas relações fundamentais e dos seus usos na ciências da computação;
- Comparação do sistema linear com os sistemas clássico e intuicionista;
- Exposição dos conceitos referentes à lógica linear, seus operadores e seu sistema de dedução natural;
- Estudo da bibliografia referente à lógica linear na ciência da computação;
- Estudo das aplicações da Lógica Linear na ciência da computação
- Desenvolvimento de uma aplicação focada em recursos para exemplificar a aplicabilidade da lógica linear;
- Explicação linear da aplicação desenvolvida e de como expressar as operações computacionais empregadas.

1.3 Metodologia

Este trabalho começou a ser desenvolvido com a pesquisa e conceituação das questões elementares da lógica e da ciência da computação: raciocínio, lógica e ciência. Os conceitos foram relacionados de forma a compreender a importância de cada um. Em seguida, foi feito um estudo mais aprofundado em cima da lógica, dos sistemas lógicos (clássico, proposicional e intuicionista), dos elementos que constituem seus objetos, dos seus aspectos (clássicos e construtivistas) e como a lógica se aplica na ciência da computação.

Após se estabelecer uma base estável, começou-se o estudo sobre lógica linear, onde diversos artigos foram estudados para melhor compreender corretamente este sistema. No decorrer deste passo, foi sendo produzido o conteúdo explicativo do que é a lógica linear, quais seus operadores, sua gramática, suas operações fundamentais. Concomitantemente, foram sendo comparados os sistemas lógicos clássico e intuicionista com a lógica linear, afim de compreender seus pontos de intersecção e de divergência. Foram citadas algumas áreas de atuação que vem sendo bastante estudadas graças ao advento da lógica linear. Por fim, foi feito um breve estudo em cima do trabalho de um pesquisador que desenvolveu uma implementação da lógica linear, o qual cedeu o código fonte para que pudesse ser apreciado. Tal análise motivou a criação da aplicação desenvolvida a seguir.

Por fim, foi desenvolvida uma aplicação para verificar e exemplificar como pode ser utilizada a lógica linear na prática. Tal desenvolvimento possibilitou uma melhor compreensão da aplicabilidade da lógica linear, pois permitiu expressar as operações computacionais através de operações lineares e da utilização do sistema de dedução natural linear. Os resultados dos experimentos foram descritos, então, e seguiu-se para a conclusão do trabalho.

1.4 Estrutura do trabalho

Referente à estrutura deste trabalho, no começo do capítulo 2 se tratará das conceituações dos elementos fundamentais do trabalho, desde os mais fundamentais, a saber: a razão, lógica e sistemas lógicos, ciência e ciência da computação, até os sistemas lógicos formais. Neste instante, se abordará: sentenças, proposições, argumentos, métodos (dedutivos e indutivos) e falácias. Neste ponto será aprofundado o estudo em cima da lógica proposicional, lógica de predicados e lógica intuicionista. Neste capítulo serão tratadas

questões referentes à origem dos termos, motivações aos seus estudos e breves citações de trabalhos anteriores a respeito deles.

No fim do capítulo 2 será tratada a lógica linear, suas operações, seu sistema de dedução linear, semelhanças e diferenças com as lógicas clássica e intuicionista. Também será descrito brevemente a implementação da lógica linear contida em (POWER; WEBSTER, 2001), uma vez que esta motivou a criação da aplicação desenvolvida para este trabalho.

O capítulo 3 se ocupará da descrição da aplicação desenvolvida para exemplificar a utilização da lógica linear e da descrição das operações lineares empregadas.

Por fim, no capítulo 4, se dará a conclusão do trabalho e projeções de possíveis trabalhos relacionados.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo se destina a definição dos conceitos relevantes abordados neste trabalho. Através da apresentação dessas definições, será possível ter a devida compreensão dos fundamentos utilizados na concepção da lógica, da sua importância para o raciocínio, da relação de ambos para com a Ciência da Computação, do isomorfismo de Curry-Howard, da lógica linear e de algumas aplicações desenvolvidas em cima desta.

2.1 Raciocínio

A palavra raciocínio vem do latim *rationem*, que significa cálculo, medida, regra, conta, que, por sua vez, deriva de *ratio*, que significa determinar, estabelecer, julgar, estimar. O professor Olavo de Carvalho, notável intelectual brasileiro, define que a razão é "em primeiro lugar, a capacidade de se abrir imaginativamente ao campo inteiro da experiência real e virtual como uma totalidade e de contrastar essa totalidade com a dimensão de infinitude que a transcende imensuravelmente" (CARVALHO, 2009).

Tem-se como raciocínio a atividade mental ou o produto desta enquanto dentro da esfera onde reside a razão. A produção intelectual com base na experiência obtida através da observação, experimentação e na capacidade de generalizar com base em critérios probabilísticos e evidências resulta em proposições, as quais unidas à capacidade de estabelecer relações lógico-formais entre si constituem o raciocínio. Olavo de Carvalho vai mais além ao definir como objeto da razão a experiência humana na sua totalidade indistinta, apenas limitada pelo senso da infinitude (CARVALHO, 2009).

Sendo a razão dependente da experiência e do que se conhece, é evidente que a produção de conhecimento através do raciocínio é um processo que pode ser construtivo ou destrutivo, realizado no decorrer do tempo e não uma fonte de verdades universalmente imutáveis.

2.2 Lógica

"A lógica exerce um papel fundamental na ciência da computação, similar ao exercido pelo cálculo na física e na engenharia tradicional. O conhecimento da lógica está se tornando uma necessidade prática para o profissional da computação" (MANNA; WALDINGER, 1985).

A palavra lógica vem do grego *logos*, que significa palavra, pensamento, ideia, argumento, relato, razão lógica ou princípio lógico. Aristóteles (384-322 a.C), ao tentar definir e delimitar os elementos que compõem a argumentação, disse serem quatro as ciências do discurso: a poética, a retórica, a dialética e a analítica (posteriormente denominada lógica). O que lhes dava estatuto de ciência era que enunciavam leis gerais que podiam ser aplicadas a todos os casos semelhantes, independendo de circunstâncias particulares (CARVALHO, 1997).

A retórica e a dialética foram agrupadas como as responsáveis pela arte da discussão e a lógica ficou com atribuição de servir de arte da demonstração científica segundo o encadeamento necessário das razões fundadas em premissas verdadeiras (CARVALHO, 1997). Dessa forma, a lógica tem grande papel ao fornecer subsídio à produção de um bom raciocínio.

À lógica como ciência, pouco importa as motivações e as emoções do indivíduo responsável pela produção racional, as circunstâncias particulares que poderiam servir de subsídio para a aceitação ou refutação de uma produção racional (como fatores religiosos, atributos físicos de um interlocutor ou ainda a natureza de uma ação a ser julgada) etc. Para esta ciência, o que importa é a forma do raciocínio, a coerência das premissas, as relações entre elas e a conclusão obtida destas.

(MORTARI, 2001) apresenta esta definição de forma mais didática, embora, talvez, menos acurada, ao dizer que a lógica se preocupa com a existência de uma boa razão para a crença em algo, ou seja, se a conclusão é consequência do que se sabe.

2.2.1 Auto-evidência

Na definição de um conceito elementar, de axiomas fundamentais de uma ciência, encontram-se verdades que precisam ser aceitas como tal, para que a partir destas se possa produzir. Para saber quais são as fundações de algo, é preciso realizar uma regressão ao fundamental deste objeto, até atingir um ponto primitivo onde o estado atual seja evidente

per se. A este estado se da o nome de auto-evidente. É o ponto a partir de onde a regressão provavelmente implicaria uma regressão ao infinito (REIS, 2006).

No momento que se define os pontos auto-evidentes em uma área, tem-se as ferramentas sólidas necessárias para produzir nela. Naturalmente, a prova de que um determinado ponto fundamental não é correto resultará na prova de que todas construções que se utilizaram deste são inválidas, *non sequiturs*.

2.3 Ciência

A palavra ciência vem do latim *scientia*, que significa conhecimento. Chama-se de ciência a produção de conhecimento sistemática fundada em princípios evidentes e demonstrações. Ainda em Aristóteles é possível encontrar referência aos rigorosos métodos utilizados na produção de conhecimento (ora Aristóteles designava o livro dos Tópicos pelo título de Metódica), onde a dialética seria a primeira formulação do método científico e a lógica o próximo passo da investigação em busca da verdade (CARVALHO, 1997).

A ciência se constitui através dos seguintes passos:

- Primeiramente formula-se uma hipótese sobre determinados fenômenos
- Verifica-se o que era suposto se confirma verdadeiro, através de observações, experimentações e medições
- Articulam-se os dados, obtidos enquanto tentava-se verificar se a hipótese era correta, em um discurso lógico-dedutivo através de métodos de demonstração lógica, "evidenciando, ao menos idealmente, a racionalidade do real" (CARVALHO, 2009).

Como se pode observar, há uma direta relação de dependência entre razão, lógica e ciência, sendo a segunda dependente da primeira e a última dependente das outras duas.

2.3.1 Ciência da Computação

"O objetivo do estudo da lógica em ciência da computação é desenvolver linguagens para modelar situações que se encontra enquanto profissional de computação, de modo que se possa, analisá-las formalmente. Analisar situação significa construir argumentos sobre elas; quer-se fazer isso formalmente, de modo que os argumentos sejam válidos e

possam ser apresentados e justificados rigorosamente, ou executados em uma máquina." (HUTH; RYAN, 2004)

Computação é a manipulação de dados a fim de obter uma determinada solução, de forma que possa ser expressa algoritmicamente. A invenção do computador para automatizar o processo de computação, que originalmente se deu para fins belicosos, se deve em grande parte a John von Neumann (1903-1957), notável matemático húngaro. Neumann criou o projeto lógico de um computador que implementa uma máquina de Turing universal (originalmente descrita em 1937), onde os dados ficam armazenados em memória e cujo processamento é sequencial e determinístico – onde a repetição de uma entrada sempre gera o mesmo resultado.

Assim surgiu a ciência da computação, ocupando-se da produção sistemática de algoritmos para manipulação de informação, enquanto o computador se ocupa da automatização do processamento de informações. Ou seja, a ciência da computação se preocupa com a forma de manipular uma informação e o computador é o modo de executar tal algoritmo.

Nesta área, a utilização com rigor da lógica é fundamental, pois é através da lógica que se pode tratar de diversas áreas, como:

- Determinação de complexidade - determinar se um determinado problema é solúvel por uma determinada linguagem;
- Interpretação abstrata – extrair informação sobre um algoritmo o executando parcialmente, para, por exemplo, depurar ou aperfeiçoar o código;
- Desenvolvimento, estudo e análise dos sistemas de tipos.

É papel da lógica, ainda, permitir a expressão do raciocínio em uma linguagem não-ambígua e auto-evidente, através da lógica formal.

2.4 Sentenças e proposições

No escopo da lógica neste trabalho, uma sentença é uma sequência de palavras cuja ordem é regida pelas regras de uma determinada gramática. Ainda neste escopo, as sentenças devem ser declarativas, onde se pode negar ou afirmar algo (MORTARI, 2001).

Neste trabalho, as proposições e sentenças declarativas serão tratadas como sendo conceitos iguais. Exemplos de sentenças:

- Sócrates é um homem;
- todo homem é mortal;
- alguns gregos são lógicos;
- o cobre é condutor de eletricidade.

Exemplos de sentenças negativas:

- Sócrates não é um homem;
- o cobre não é condutor de eletricidade.

Fica evidente, assim, que não há a preocupação com a verossimilhança na estrutura de uma sentença.

2.4.1 Argumentos

Justificar uma afirmação e dar razões para uma certa conclusão são fundamentais para garantir que se está raciocinando corretamente. Um argumento pode ser visto como um conjunto não-vazio e finito de sentenças relacionadas entre si, onde a conclusão se segue das premissas (MORTARI, 2001).

2.4.2 Dedução

Dado seguinte argumento:

- Sócrates é um homem;
- todo homem é mortal;
- portanto, Sócrates é mortal.

As duas primeiras sentenças são as premissas e a última é a conclusão. Este é um caso de argumento dedutivo, também chamado de silogismo, onde a conclusão se segue das premissas.

2.4.3 Indução

Outra possibilidade de argumentação é através da indução, quando se tenta atingir um gênero a partir do conhecimento das espécies que o constituem. Um exemplo de indução:

- O cobre é condutor de eletricidade, assim como a prata, o ouro, o ferro, o zinco e outros metais,
- Logo, todo metal é condutor de eletricidade.

Fica evidente a falta de solidez de argumentos indutivos, uma vez que não há a garantia da conclusão, que para ser provada equivocada, basta haver um exemplo de metal que não seja condutor de eletricidade. É importante ressaltar que, de forma diferente, a indução matemática consegue atingir o gênero a partir de rigorosos métodos e não se fundamenta da mesma forma que a argumentação indutiva.

2.4.4 Falácias

Falácia é um argumento inválido ou incapaz de provar o que almeja. Nesta seção serão expostos alguns exemplos de erros lógicos que podem incorrer em falácias. Um primeiro exemplo de falácia:

- alguns gregos são lógicos;
- alguns lógicos são chatos;
- por isso, alguns gregos são chatos.

Este é um clássico exemplo de *non sequitur*, onde a conclusão não se dá a partir das premissas, logo, o argumento é inválido. Chama-se esses argumentos de falácias.

Outra possibilidade é quando se tenta atingir uma espécie não pertencente ao gênero devido. Chama-se esta falácia de *Dicto Simpliciter*.

- “Se você matou alguém, deve ir para a cadeia.”

Ora, nem todos os casos onde há um assassinato, necessariamente, a lei condena o indivíduo desta forma.

Quando se tenta induzir algo a partir de uma amostragem insuficiente se incorre em uma falsa indução:

- "O cobre é condutor de eletricidade. Logo, todos metais são condutores de eletricidade."

Outra falácia é tentar compreender o todo a partir de propriedades das partes constituintes deste, como se segue:

- "Todas as peças deste caminhão são leves; logo, o caminhão é leve."

O oposto também é inválido:

- "O caminhão é pesado; Logo, todas as peças deste caminhão são pesadas."

Outro equivoco é julgar que por dois ou mais eventos haverem ocorrido concomitantemente, a concorrência é o razão da conclusão. Chama-se esta falácia de *Cum hoc ergo propter hoc* (junto disto, logo, por causa disso). Como exemplo:

- "Quanto menos bombeiros enfrentarem o fogo, mais forte o fogo será; Logo, os poucos bombeiros causam o aumento do fogo."

De forma análoga, pode-se tentar obter uma conclusão pela sequência de alguns fatos, o que é inválido. Chama-se essa falácia de *Post hoc ergo propter hoc* Exemplo:

- "O Japão se rendeu logo após a utilização das bombas atômicas por parte dos EUA. Portanto, a paz foi alcançada devido à utilização das armas nucleares."

2.5 Lógica formal

Nesta seção serão expostos breves comentários essenciais sobre lógica proposicional, lógica intuicionista e lógica de predicado. Excelentes e mais aprofundadas leituras podem ser encontradas em (AL., 2008) e (REEVES; CLARKE, 2003).

2.5.1 Lógica proposicional

A lógica proposicional se baseia em proposições que se pode argumentar a respeito de sua veracidade ou falsidade. Esse assunto receberá um tratamento conciso e curto, pode-se conferir (HUTH; RYAN, 2004) e (LAGO PEREIRA, 2010) para mais informações sobre lógica proposicional.

Primeiramente, na lógica proposicional, precisa-se organizar as declarações de forma atômica e indecomponíveis, da seguinte forma:

- p: ganhei na loteria;
- q: comprei um bilhete de loteria;
- r: ganhei o bolo de apostas;

Pode-se utilizar as seguintes regras em conjunto com as proposições, dado as proposições supracitadas:

- $\neg$: expressa a negação de uma proposição, exemplo: $\neg$p significa "não ganhei na loteria". Intuitivamente, não há uma diferença entre uma proposição e a dupla negação desta. Logo, !(!p) = p;
- $\vee$: expressa uma disjunção entre proposições, exemplo: p $\vee$ r significam "ganhei na loteria ou ganhei o bolo de apostas";
- $\wedge$: expressa uma conjunção entre proposições, exemplo: q $\wedge$ p significam "comprei um bilhete de loteria e ganhei na loteria";
- $\rightarrow$: expressa uma inferência lógica, exemplo: p $\rightarrow$ q significa "se ganhei na loteria, então comprei um bilhete de loteria";

A partir dessas regras é possível construir proposições mais complexas, da seguinte forma. Dadas as seguintes premissas:

- p: a umidade está elevada;
- q: a temperatura está elevada;
- w: a pressão atmosférica está elevada;

- r: choverá;

Pode-se construir a proposição: “$p \vee (w \wedge q) \rightarrow r$”

Pode-se ler esta sentença como “Se a umidade estiver elevada ou a temperatura e a pressão atmosférica estiverem elevadas, então choverá”.

2.5.2 Lógica intuicionista

Na filosofia da matemática, o construtivismo afirma ser necessário encontrar ou construir um objeto matemático para provar que ele existe. Supor que um objeto não existe e utilizar a partir disso uma contradição não significa encontrar ou construir o objeto, portanto, não é aceito por este sistema.

A lógica proposicional intuicionista, ou lógica construtivista, foi criada por Arend Heyting para prover uma base formal para o intuicionismo de Brouwer. Este sistema lógico se baseia na justificativa das verdades, de forma que se as hipóteses são verdadeiras e justificáveis, a conclusão também o é. Sendo assim, uma disjunção somente será tautologia se ao menos um dos termos for tautologia e uma conjunção somente será tautologia se ambos termos forem tautologia. Portanto, a lei da dupla negação, lei de Pierce e a lei do terceiro excluído não podem ser demonstrados neste sistema lógico.

2.5.3 Lógica de predicados

Segundo (HUTH; RYAN, 2004), a lógica proposicional trata de maneira bastante satisfatória com componentes de frases do tipo não, e, ou e se... então, mas os aspectos lógicos das linguagens naturais são muito mais expressivos e complexos. Não é possível expressas na lógica proposicional a seguinte sentença: “Todo homem é mortal”.

Tal sentença só poderia ser expressa citando um a um todos os homens. Para tentar suprir as limitações da lógica proposicional, a lógica de predicado permite lidar com modificadores como “existe”, “todo”, “entre” e “apenas”. Na sentença acima há uma propriedade, ser mortal, que pode ser expressa na lógica de predicado junto com suas relações lógicas e dependências (HUTH; RYAN, 2004). Para tal, introduziu-se o conceito de predicado.

Através dos predicados, pode-se expressar que João é um homem, através do predicado H(João) e indicar que João é um estudante, através do predicado E(João). De forma análoga, pode-se denotar que Paulo é mais jovem que João, por J(Paulo,João) (HUTH;

RYAN, 2004). Existem dois quantificadores na lógica de predicados:

- $\exists$ - quantificador existencial. Permite estabelecer a existência de um objeto sem identificar esse objeto explicitamente.
- $\forall$ - quantificador universal. Permite estabelecer fatos a respeito de diversos objetos sem ter de enumerá-los.

Através destes operadores podemos expressar, por exemplo, que dados alguns blocos e uma mesa, ou um bloco está sobre a mesa ou está sobre outro bloco. Expressa-se isso da seguinte forma:

$$\forall \text{X[bloco(X)} \rightarrow \exists \text{Y[sobremesa(X,Y)]]}$$

2.5.4 Isomorfismo de Curry-Howard

"O Isomorfismo de Curry-Howard estabelece uma correspondência bijetiva entre sistemas dedutivos da lógica formal e cálculos computacionais de teoria de tipos" (FERNANDES, 2009).

Estabelecido na década de 1940, pelos matemáticos Haskell B. Curry e William A. Howard, o isomorfismo de Curry-Howard permite estabelecer uma correspondência entre a lógica formal e o λ-cálculo simplesmente tipado (FERNANDES, 2009).

(FERNANDES, 2009) apresenta detalhadas deduções acerca das proposições do isomorfismo de Curry-Howard, para mais informações, tal artigo deve ser consultado. De forma breve, é evidente a semelhança entre a dedução para o sequente $\vdash \varphi \rightarrow \psi \rightarrow \psi$:

$$\dfrac{\dfrac{\dfrac{}{\varphi, \psi \vdash \psi}\,(\text{ax})}{\varphi \vdash \psi \rightarrow \psi}\,(\rightarrow\text{I})}{\vdash \varphi \rightarrow \psi \rightarrow \psi}\,(\rightarrow\text{I})$$

E a prova de $\vdash \lambda y : \varphi.\lambda x : \psi.x : \varphi \rightarrow \psi \rightarrow \psi$:

$$\dfrac{\dfrac{\dfrac{}{y : \varphi, x : \psi \vdash x : \psi}\,(\text{var})}{y : \varphi \vdash \lambda x : \psi.x : \varphi \rightarrow \psi}\,(\rightarrow\text{I})}{\vdash \lambda y : \varphi.\lambda x : \psi.x : \varphi \rightarrow \varphi \rightarrow \psi}\,(\rightarrow\text{I})$$

As seguintes correspondências podem ser obtidas da tabela 2.1.

[1] Os tipos habitados do λ-cálculo simplesmente tipados são aqueles que, quando considerados como proposições são tautologias do fragmento positivo da lógica proposicional intuicionista.

Tabela 2.1: Curry-Howard: Correspondências

Fórmulas $\leftrightarrow$ Tipos
Provas $\leftrightarrow$ Termos
β-Redução $\leftrightarrow$ Normalização de provas
Provabilidade $\leftrightarrow$ *Inhabitation*[1]
Teoremas $\leftrightarrow$ Tipos habitados

2.6 Lógica Linear

"Algumas das melhores coisas na vida são gratuitas e outras não. A verdade é gratuita. Tendo-se provado um teorema, você pode usar esta prova tantas vezes quanto desejar, sem custo adicional. Comida, por outro lado, tem um custo. Tendo-se cozinhado um bolo, você pode comê-lo apenas uma vez. Se a lógica tradicional é sobre verdade, então a lógica linear é sobre comida" (WADLER, 1993).

2.6.1 Surgimento e conceituação

Criada em 1987 por Jean-Yves Girard, a Lógica linear é um refinamento dos sistemas clássico e intuicionista, unindo algumas características de ambas (XAVIER; OLIVEIRA, 2009). Seu criador a caracterizou como uma lógica de recursos, algo que vem sendo cada vez mais aceito por cientistas da computação (LINCOLN, 1992). Isso acontece pois as sentenças são tratadas como recursos ou ações e as consequências lógicas são tratadas como transições entre estados (MARTINS, 2003). Neste texto serão realizados alguns confrontos entre sistemas lógicos. Pode-se consultar (PRATT, 2005) para um detalhamento maior, onde a lógica linear é colocada como complemento da lógica clássica.

Na área tecnológica, *a priori*, tem-se de tomar decisões. Estas decisões devem levar em consideração o gerenciamento de recursos, uma vez que este mundo não dispõe de recursos ilimitados e a partilha de recursos é tarefa habitual na área da computação, nos sistemas multiprogramados (MARTINS, 2003). Neste escopo, a lógica linear atua lado-a-lado com o planejamento do sistema, reduzindo este hiato entre o planejamento e a prática.

2.6.2 Operações em lógica linear

Neste ponto, é conveniente a introdução aos operadores da lógica linear. Girard (GIRARD, 1986) faz a seguinte divisão de conectivos:

- os multiplicativos ($\otimes$, $\parr$, $\multimap$) que são versões bilineares do "e", "ou" e do "implica";
- os aditivos ($\oplus$, $\&$) que são versões lineares do "ou" e do "e";
- os exponenciais (!, ?) que permitem preservar a força lógica.
- Das conjunções: o operador $\otimes$ representa uma conjunção multiplicativa onde se

pode utilizar ambas premissas e o operador $\&$ representa uma conjunção aditiva onde se deve escolher qual premissa utilizar, porém não as duas;

- Das disjunções: o operador $⅋$ representa uma disjunção multiplicativa onde se tem a segunda premissa somente se não se tiver a primeira e o operador $\oplus$ representa uma disjunção aditiva onde se tem qualquer uma das premissas, porém não as duas e não há a possibilidade de escolha.

A tabela 2.2 resume os operadores da lógica linear.

Tabela 2.2: Operadores da lógica linear.

$\&$	e aditivo
$\oplus$	ou aditivo
$\otimes$	e multiplicativo
$⅋$	ou multiplicativo
$\top$	*top*[1]
$\bot$	*bottom*[2]
$\multimap$	implicação linear

E a gramática de proposições lineares é:

$$\text{A,B,C} ::= \text{X} \parallel \text{A} \multimap \text{B} \parallel \text{A} \otimes \text{B} \parallel \text{A} \,\&\, \text{B} \parallel \text{A} \oplus \text{B} \parallel \,!\text{A}$$

2.6.3 Lógica clássica e lógica intuicionista x lógica linear

Nesta seção será realizada uma breve comparação entre os sistemas lógicos convencionais, clássico e intuicionista, e a lógica linear. No decorrer do texto serão apresentadas deduções e operações lógicas, porém abstraindo passos, em alguns casos. Para uma descrição extensiva e de grande qualidade, consultar (MELLIèS, 2010). Para a descrição do próprio criador da lógica linear sobre sua semântica e sintaxe, consultar (GIRARD, 1995).

A lógica clássica trata com verdades universais e, por isso, é amplamente utilizada em ramos onde a matemática exerce maior influência. Na lógica clássica, se um fato é usado para concluir outro fato, o primeiro ainda estará disponível (WADLER, 1993). Neste sistema, uma premissa é verdadeira ou falsa e há construções que permitem a duplicação ou eliminação de premissas. A regra de enfraquecimento diz que se pode acrescentar

[1] O operador *top* sinaliza uma premissa que é tautologia.
[2] O operador *bottom* sinaliza uma premissa que é falsa.

premissas sem alterar o grau de expressividade de uma determinada solução. Tal regra pode ser representada como se segue:

$$\frac{\Delta \to \Sigma}{\Delta, A \to \Sigma}$$

A regra de contração permite que se elimine proposições sem alterar a conclusão (MARTINS, 2003; LINCOLN, 1992). Tal regra pode ser representada como se segue:

$$\frac{\Delta, A, A \to \Gamma}{\Delta, A \to \Gamma}$$

Na lógica clássica, considera-se que, dada uma proposição A, ou A é x ou é y, não havendo uma terceira possibilidade. Tal lei se chama "lei do terceiro excluído", *tertium non datur*. Tal regra pode ser representada como se segue:

$$(p \vee \neg p)$$

A regra da dupla negação permite eliminar ou introduzir uma dupla negação. Negar uma proposição duas vezes é o mesmo que afirmá-la. A remoção é como se segue:

$$\frac{\neg\neg A}{A}$$

A introdução é como se segue:

$$\frac{A}{\neg\neg A}$$

Outra lei aceita na lógica clássica é a da não contradição, que diz que uma proposição verdadeira não pode ser falsa e uma proposição falsa não pode ser verdadeira. Portanto, nenhuma proposição pode ser verdadeira e falsa ao mesmo tempo. A esta lei se dá o nome de lei da dupla negação. Tal lei é representada como se segue:

$$\neg(p \wedge \neg p)$$

A lógica intuicionista é um sistema que leva em consideração a prova da existência das premissas usadas em processo construtivo. Se as hipóteses são verdadeiras, também a conclusão o será. Na lógica intuicionista se pode ter outros valores além do verdadeiro e do falso (XAVIER; OLIVEIRA, 2009). É, portanto, evidente a incompatibilidade das regras do terceiro excluído e da dupla negação neste sistema. A Lógica Intuicionista,

ainda, permite o uso das regras de contração e enfraquecimento somente na hipótese, nunca na conclusão (MELLIèS, 2010).

Por exemplo, dadas as proposições, como recursos:

$$D \triangleq \text{"}Um\,dólar\text{"};$$

$$M \triangleq \text{"}Uma\,pizza\text{"};$$

$$C \triangleq \text{"}Uma\,torta\text{"}.$$

E dado o funcionamento da seguinte máquina de vendas:

$$D\,implica\,M;$$

$$D\,implica\,C.$$

Na lógica clássica, pode-se descrever o funcionamento da máquina como "Com um dólar, eu posso comprar uma pizza e uma torta". Ora tal operação pode ser considerada válida no sistema clássico, uma vez que onde se pode duplicar operadores. Tal concepção é incorreta, porém, na lógica linear, onde se pode reconhecer os recursos escassos deste cenário. Tal confusão acontece em ambos sistemas lógicos, clássico e intuicionista, em virtude da confusão entre dois tipos de conjunção: uma significando intuitivamente "Eu tenho ambos" e outra significando "Eu tenho uma escolha" (LINCOLN, 1992).

Na lógica linear pode-se ver esta situação de forma diferente, onde, com um dólar, deve-se escolher entre uma torta e uma pizza. Para possibilitar tais construções, a lógica linear se utiliza de duas operações de conjunção, duas operações de disjunção e um exponencial que permite marcar as proposições que podem ser utilizadas muitas vezes (LINCOLN, 1992). A conjunção multiplicativa $\otimes$ significa que se tem ambos elementos, semelhante à forma de interpretação atribuída à lógica clássica: $D \multimap M \otimes C$ como "Com um dólar, eu posso comprar uma pizza e uma torta". A conjunção aditiva $\&$ significa que se tem o direito de escolher entra um ou outro elemento, porém não ambos, como em: $D \multimap M \& C$ como "Com um dólar, eu posso escolher comprar uma pizza ou uma torta". A disjunção multiplicativa $⅋$ significa que se não houver o primeiro elemento, haverá o segundo. Pode-se ver essa operação de forma mais didática expressando $A \multimap B$ como $A^{\perp} ⅋ B$. A disjunção aditiva $\oplus$ significa que se tem um ou outro elemento, mas não é

nossa escolha qual pegar, como em: $D \multimap M \oplus C$ como "Com um dólar posso comprar uma pizza ou uma torta" (LINCOLN, 1992).

Para completar o sistema lógico, tem-se ainda dois exponenciais: ! e ?. A fórmula !A pode ser vista como a impressão na tela de letras A enquanto se pressiona o botão, gerando qualquer número de As (LINCOLN, 1992). Ao possuir !X de um elemento X, não há necessidade de gerenciar tal recurso pois ele é inesgotável, porém, se não há !X deste elemento, tem-se que gerenciar a sua utilização.

Girard descobriu que a implicação intuicionista $A \rightarrow B$ poderia ser decomposta em dois conetivos separados $!A \multimap B$ e mostrou que com a adição dos exponenciais expostos acima seria possível traduzir as lógicas intuicionista e clássica em lógica linear. Para impedir o uso arbitrário de regras de contração e enfraquecimento, somente se pode utilizar tais regras em sentenças marcadas com os exponenciais ! e ?. Dessa forma se preserva o caráter construtivo deste sistema lógico.

2.6.4 As regras da lógica linear

O sistema de lógica clássico incentiva a utilização de hipóteses. O caráter construtivo da lógica intuicionista não permite tal recurso, uma vez que as premissas devem ser provadas antes que se possa utilizá-las. Deve-se verificar a tabela 2.3 para apreciar as regras.

O sistema de provas utilizado neste trabalho é o de dedução natural. Este sistema serve para verificar a derivabilidade de uma expressão, mostrando-nos uma das possíveis derivações. Na dedução natural, os julgamentos ficam acima da linha e a implicação linear fica abaixo. Utiliza-se julgamentos, pois a lógica linear herdou o caráter construtivista da lógica intuicionista, portanto a prova de suas declarações deve poder ser obtida ou construída. As premissas são representadas por Γ e Δ, cada um contendo premissas diferentes. Escreve-se Γ,Δ para denotar a concatenação destas premissas. $\vdash$ representa um julgamento e a linha que separa a parte de cima e a de baixo representa uma regra (WADLER, 1993).

Se uma concatenação do tipo Γ,Δ aparecer em uma regra, ambas devem conter variáveis diferentes. De forma análoga, no caso de Γ,x, a variável x não deverá estar contida em Γ. Na regra da contração, a notação t[x/y,x/z] representa um termo t onde x substitui todas ocorrências de y e de z (WADLER, 1993). Deve-se observar que há duas formas

de premissas, $\langle A \rangle$ e [A]. Uma premissa da forma $\langle A \rangle$ significa uma representação linear, onde se tem apenas uma unidade do recurso A. A segunda forma, [A], oferece uma representação intuicionista, onde se pode utilizar as regras de contração e enfraquecimento, podendo-se gerar tantos recursos quanto desejado.

Tabela 2.3: Regras da lógica linear.

$$\frac{}{\langle A \rangle \vdash A}\ \langle Id \rangle \qquad \frac{}{[A] \vdash A}\ \text{[Id]} \qquad \frac{\Gamma, \Delta \vdash A}{\Delta, \Gamma \vdash A}\ \text{Exchange}$$

$$\frac{\Gamma, [A], [A] \vdash B}{\Gamma, [A] \vdash B}\ \text{Contração} \qquad \frac{\Gamma \vdash B}{\Gamma, [A] \vdash B}\ \text{Enfraquecimento}$$

$$\frac{\Gamma \vdash A}{[\Gamma] \vdash !A}\ !\text{-I} \qquad \frac{\Gamma \vdash !A \ \ \Delta, [A] \vdash B}{[\Gamma], \Delta \vdash B}\ !\text{-E}$$

$$\frac{\Gamma, \langle A \rangle \vdash B}{\Gamma \vdash A \multimap B}\ \multimap\text{-I} \qquad \frac{\Gamma \vdash A \multimap B \ \ \Delta \vdash A}{\Gamma, \Delta \vdash C}\ \multimap\text{-E}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma, \Delta \vdash A \otimes B}\ \otimes\text{-I} \qquad \frac{\Gamma \vdash A \otimes B \quad \Delta, \langle A \rangle, \langle B \rangle \vdash C}{\Gamma, \Delta \vdash C}\ \otimes\text{-E}$$

$$\frac{\Gamma \vdash A \quad \Gamma \vdash B}{\Gamma \vdash A \& B}\ \&\text{-I} \qquad \frac{\Gamma \vdash A \& B}{\Gamma \vdash A}\ \&\text{-E1} \quad \frac{\Gamma \vdash A \& B}{\Gamma \vdash B}\ \&\text{-E2}$$

$$\frac{\Gamma \vdash A}{\Gamma \vdash A \oplus B}\ \oplus\text{-I1} \qquad \frac{\Gamma \vdash B}{\Gamma \vdash A \oplus B}\ \oplus\text{-I2} \qquad \frac{\Gamma \vdash A \oplus B \ \ \Delta, \langle A \rangle \vdash C \ \ \Delta, \langle B \rangle \vdash C}{\Gamma, \Delta \vdash C}\ \oplus\text{-E}$$

Pode-se encontrar em (WADLER, 1993) um exemplo baseado em premissas relativas a dinheiros e pizzas no seu trabalho, tal exemplo ilustra muito bem o cenário sensível a recursos ao qual a lógica linear se destina e, portanto, será utilizado neste texto também.

Pode-se considerar os termos como recursos, de forma que se pode ler $A \multimap C$ como "consumindo A se produz B". Considerando que A é a proposição "Eu tenho uma nota de 10 reais", B é "Eu tenho uma pizza" e C é "Eu tenho um bolo". Pode-se ler $A \multimap B$ como "consumindo uma nota de 10 reais eu terei uma pizza" e $A \multimap C$ como "consumindo uma nota de 10 reais eu terei um bolo".

Considerando que no mundo real os recursos são limitados, dinheiro, portanto, não é infinito. Gastar uma nota de 10 reais faz com que não se tenha mais esta. No caso, ao se ter uma nota de 10 reais, pode-se comprar ou uma pizza ou um bolo. Isso pode ser expressado através do operador $\&$. A regra segue:

$$\frac{A \vdash B\&C}{A \vdash B}\ \&\text{-E1}\quad \frac{A \vdash B\&C}{A \vdash C}\ \&\text{-E2}$$

Na lógica clássica, a utilização de uma proposição pode ser feita de forma ilimitada, equivalentemente, na lógica linear, o operador $\otimes$ se encarrega de expressar tal situação. A regra segue:

$$\frac{\langle A\rangle \vdash B \quad \langle A\rangle \vdash C}{\langle A\rangle, \langle A\rangle \vdash B \otimes C}\ \otimes\text{-I}$$

A suposição $\langle A\rangle$ pode ser vista como uma unidade de recurso A que se tem à disposição e a suposição [A] pode ser vista como uma fonte ilimitada de recursos A. Utilizando a regra de enfraquecimento se pode gerar recursos *ad libitum* e dessa forma se pode capturar o caráter intuicionista de determinados problemas. Considerando a premissa A como "eu tenho 10 reais", a interpretação linear da sentença $\langle A\rangle$ seria "eu tenho uma nota de 10 reais", enquanto a interpretação intuicionista linear de [A] seria "eu tenho um estoque ilimitado de notas de 10 reais".

$$\frac{}{\langle A\rangle \vdash A}\ \langle Id\rangle \quad \frac{}{[A] \vdash A}\ \text{[Id]}$$

O axioma $\langle A\rangle$ significa que, se eu tiver uma nota de 10 reais no meu bolso, posso retirar dele esta nota, enquanto o axioma [A] significa que, se eu tenho ilimitadas notas de 10 reais no meu bolso, posso retirar dele uma nota.

Assim como Γ,$\langle A\rangle \vdash$B pode ser provado, Γ,[A]$\vdash$B também pode. A prova é como se segue:

$$\frac{\dfrac{\Gamma, \langle A\rangle \vdash B}{\Gamma \vdash A \multimap B}\multimap -I \quad \dfrac{}{[A] \vdash A}\ \text{[Id]}}{\Gamma, [A] \vdash B}\multimap -E$$

É importante notar que [A] é uma suposição intuicionista, enquanto !A é uma suposição linear de mesmo sentido. Ora se é possível retirar um A de [A], é possível retirar um A de !A também. A regra é como se segue:

$$\frac{A \vdash B}{[A] \vdash !B}\ !\text{-I}$$

Adicionando o axioma:

[B]$\vdash$D

onde D significa felicidade, pode-se concluir que B leva a felicidade. Utilizando !-E se tem:

$$\frac{[A] \vdash !B \quad [B] \vdash D}{[A] \vdash D}\ \text{!-E}$$

Dado um estoque ilimitado de notas de 10 reais, pode-se comprar ilimitados pedaços de pizza, os quais levam à felicidade. Há uma equivalência entre a suposição $\langle !A \rangle$ e [A], como a seguinte prova expressa:

$$\frac{\dfrac{\Gamma, \langle A \rangle \vdash B}{\Gamma \vdash A \multimap B} \multimap -I \quad \dfrac{\dfrac{}{[A] \vdash A}\,[\text{Id}]}{[A] \vdash !A}\,\text{!-I}}{\Gamma, [A] \vdash B} \multimap -E$$

Ou no sentido inverso:

$$\frac{\dfrac{}{\langle !A \rangle \vdash !A}\,\langle Id \rangle \qquad \Gamma, [A] \vdash B}{\Gamma, \langle !A \rangle \vdash B}\ \text{!-E}$$

Com todas essas características, a lógica linear possibilita:

- Novas formas de analisar os problemas, como “me de A uma vez e eu lhe concederei B”, com aplicações em inteligência artificial, lógicas de programação refinadas, focadas em estados, análise da lógica clássica, tratamento de exceções, interpretação abstrata, etc;
- Novas formas de representar demonstrações;
- Novas regras para expressar constantes no uso de cópias resultando em um fragmento da lógica linear de computação *polytime*;
- Reduzir o hiato entre o raciocínio computacional e os recursos e estados do mundo real.

2.7 Uma aplicação da lógica linear

Desde a sua criação, aproximadamente em 1987, a lógica linear teve diversas tentativas de implementação. Dentre os diversos trabalhos realizados, James Power, pesquisador do departamento de Ciência da Computação da universidade *National University of Ireland*, desenvolveu uma codificação do sistema da lógica linear baseada em estados (POWER; WEBSTER, 2001) e a cedeu gentilmente para observação. Outras implementações importantes são:

- Lolli (HODAS, 1992): um refinamento de λProlog para operar com lógica linear;
- LLP (HODAS et al., 1998): uma linguagem de programação lógica baseada na lógica linear intuicionista. É um superconjunto de Lolli.
- LilliMon (AL., 2005): uma extensão da linguagem Lolli, focada em programação concorrente monádica.

Nesta seção será brevemente descrita a implementação desenvolvida em (POWER; WEBSTER, 2001), pois ela baseou a aplicação desenvolvida para este trabalho. O motivo para tal é que esta implementação é mais recente que as outras implementações listadas acima e se mostrou bastante didática.

2.7.1 COQ - Um sistema de Gerenciamento de Provas Formais

O Coq é um sistema de gerenciamento de provas formais distribuído sob licença *GNU Lesser General Public License* versão 2 (GNU LESSER GENERAL PUBLIC LICENSE, 2010). Este sistema trabalha com uma linguagem matemática de alto nível chamada Gallina (THE COQ PROOF ASSISTANT, 2010). Através desta linguagem própria e de um conjunto de regras já programadas, é possível:

- definir funções e predicados que podem ser avaliados eficientemente;
- declarar teoremas matemáticos e especificações de software;
- desenvolver intuitivamente provas formais desses teoremas;
- uniformizar a escrita;
- verificar a tipagem correta;

- implementar uma lógica construtiva de alta ordem, facilitando a descrição de objetos lógicos;
- Contrastar aplicações desenvolvidas usando lógica clássica e aplicações em outras lógicas - o Coq tem duas hierarquias de tipo: *Set* de tipos construtivos e *Prop* para a lógica clássica;
- o suporte a definições indutivas, possibilitando em construções diferentes das padrões da lógica clássica.

Um dos mais notáveis recursos, para o escopo deste trabalho, reside na possibilidade de se poder ensinar, algoritmicamente, como resolver problemas lógicos ao programa. Uma vez que o programa aprenda a realizar uma dedução, ele pode repeti-la em outras ocasiões e até tentar desenvolver provas automaticamente(PINTO; FRADE, 1998). Muitos trabalhos foram desenvolvidos em conjunto com as regras que acompanham o programa. James Power desenvolveu uma codificação da lógica linear utilizando Coq (POWER; WEBSTER, 2001) e é justamente considerando o trabalho dele que esta seção se orientará.

Através do contato com James Power, um dos autores das codificação da lógica linear para Coq, pode-se obter os códigos fontes para a implementação da lógica linear usando Coq. Em seguida, será explicado brevemente o experimento realizado. A prova dos experimentos não será descrita neste trabalho por fugir do escopo e somente será descrito o primeiro experimento, para tal se deve conferir (POWER; WEBSTER, 2001).

James Power teve sucesso em sua tentativa de codificar a lógica linear utilizando Coq e suas operações nativas. Esta codificação permitiu expressar problemas baseados em recursos linearmente. Tal abordagem é bastante didática e inspirou a programação da aplicação desenvolvida neste trabalho para explicar a forma linear de raciocínio computacional.

2.7.2 Cenário dos blocos

Este é um cenário clássico da área de planejamento da inteligência artificial. Neste cenário, tem-se um conjunto de blocos sobre uma mesa e o objetivo é construir um ou mais conjuntos de blocos vertical. Só se pode mover um bloco por vez e ele deve ser colocado em cima da mesa ou acima de outro bloco. Portanto, nenhum bloco que possua blocos no topo pode ser movido.

O objetivo do teste deste cenário no escopo da lógica linear é testar como isso poderia ser resolvido utilizando um sistema baseado em estados, onde os objetivos e transições podem ser representados (POWER; WEBSTER, 2001). Neste exemplo, existe um braço robótico que pode pegar os blocos de cima e movê-los.

Primeiramente se deve definir os predicados. O seguinte código define os predicados em Coq - a sintaxe utilizada é da aplicação Coq:

- Variable emcima : Bloco -> Bloco -> ILinProp.
- Variable mesa : Bloco -> ILinProp.
- Variable limpar : Bloco -> ILinProp.
- Variable segura : Bloco -> ILinProp.
- Variable vazio : ILinProp.

Aqui estão definidos os cinco predicados na sintaxe do Coq. Primeiro se declara o predicado e depois o seu tipo. O primeiro, por exemplo, representa um predicado de nome *emcima* que possui dois elementos do tipo Bloco e cuja relação lógica forma um tipo ILinProp, representando uma proposição linear.

A relação emcima entre blocos pode ser vista como *(emcima x y)*, onde x está sobre o bloco y. Os predicados unários: *(mesa x)* indica que o bloco x está sobre ela e *(vazio x)* indica que não há nada sobre x. Dois predicados relativos ao estado do robô: *(segura x)* indica que o robô segura o bloco x e *(vazio x)* indica que o robô não está segurando bloco algum.

O próximo passo é definir as ações do sistema, com base no estado atual. Para isso se deve definir qual o estado atual, quais recursos se tem e para qual estado se deve ir. O processo funciona como uma dedução linear. As ações criadas em (POWER; WEBSTER, 2001) são as seguintes:

- Ação *Pegar*: diz que se o braço estiver em um estado *vazio* (sem estar segurando nada) e o bloco x não tiver nada em cima de si, o braço irá pegá-lo. Há duas possibilidades, ou x estava sobre a mesa ou estava sobre outro bloco y. No segundo caso, precisa-se informar que y agora não possui bloco algum sobre si (*limpar*);

- Ação *Botar*: diz que se o braço estiver em um estado *segurando*, segurando um bloco x, pode-se levá-lo a um estado *vazio*. Com nada sobre x, ou x estará sobre a mesa ou sobre outro bloco.

Para a codificação dessas ações na semântica do Coq, o operador $\otimes$ é representado por **, o operador & é representado por && e a inferência linear $\multimap$ é representada por -o. As ações Pegar e Botar tem a seguinte forma - a sintaxe utilizada é da aplicação Coq:

```
Axiom Pegar :
    (x,y:Bloco)
    ('(vazio ** (limpar x))
|- (segura x) ** (((mesa x) -o um) && ((emcima x y) -o (limpar x y)))).
Axiom Botar :
    (x,y:Bloco)
    ('(vazio ** (limpar x))
|- (segura x) ** (((mesa x) -o um[1]) && ((emcima x y) -o (limpar x y)))).
```

2.7.3 Da importância desta implementação

Esta implementação trouxe uma abordagem fortemente focada em estados e recursos (POWER; WEBSTER, 2001), além de ser bastante didática. A praticidade e agilidade no desenvolvimento da aplicação motivaram o presente trabalho. Com base nesta implementação optou-se por desenvolver uma aplicação também baseada em estados e recursos.

A seção seguinte se ocupará da aplicação desenvolvida para este trabalho e da sua experimentação.

[1] A unidade 1 (um) é a identidade do operador $\otimes$ e denota uma inferência linear que não produz nada.

3 A APLICAÇÃO - UM SIMULADOR DE MÁQUINAS VIRTUAIS

Tendo os conceitos da lógica linear expostos, foi desenvolvida uma aplicação para melhor exemplificar a aplicação da lógica linear e suas implicações. Baseado no trabalho de James Power (POWER; WEBSTER, 2001), desenvolveu-se trabalho uma codificação do sistema da lógica linear para este. Decidiu-se pela criação de uma aplicação também baseada em estados e preservação de recursos. A aplicação desenvolvida é um simulador de um gerenciador de máquinas virtuais em um sistema de recursos limitados.

3.1 Descrição da aplicação

O sistema em questão é um simulador de um gerenciador de máquinas virtuais. O usuário pode escolher quanto de cada um dos recursos do sistema ele quer utilizar em cada máquina virtual, limitado pelas capacidades do sistema. A virtualização consiste na emulação de recursos computacionais de forma que o hardware original é omitido.

Definida a quantidade de recursos a alocar por máquina virtual, o usuário pode executar tantas máquinas quanto desejar, porém o sistema não deve permitir que se tente utilizar mais recursos do que o disponível, uma vez que os recursos são finitos.

Escolheu-se uma aplicação que se utiliza de concorrência, por esta ser uma das áreas que mais podem se beneficiar da lógica linear. O motivo para tal é que a lógica linear oferece um controle mais adequado dos recursos disponíveis no sistema e possibilita reduzir a necessidade de sincronização. Tais tópicos serão tratados na seção 3.2.

3.1.1 Os recursos do sistema

Neste sistema utilizado como exemplo, os recursos disponíveis são:

- 8 Gb de memória;

- Até 100 % do processador;
- 2000 Gb de espaço em disco;
- 5 impressoras;
- 5 Gravadores;
- ilimitadas conexões.

Seja A um recurso qualquer. Enquanto houver recurso A disponível, as seguinte regras permitem que este seja utilizado:

$$\frac{}{\langle A\rangle \vdash A}\ \langle Id\rangle \qquad \frac{}{[A] \vdash A}\ \text{[Id]}$$

Através do monitor do sistema, (Figura 3.1) na interface do programa, pode-se verificar tais recursos.

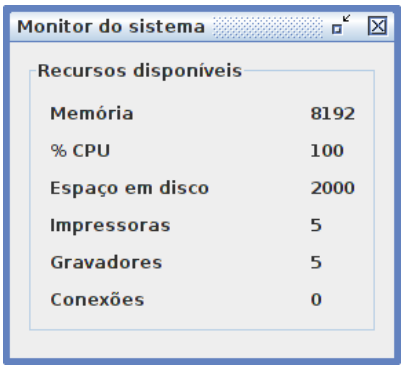

Figura 3.1: Tela do monitor do sistema.

Note que o recurso conexões está marcado com o um recurso intuicionista, da forma *[Conexão]* e, portanto, não há uma quantidade de recursos delimitada. Pode-se utilizar a regra da introdução do exponencial ! da seguinte forma:

$$\frac{\Gamma \vdash Conexão}{[\Gamma] \vdash !Conexão}\ \text{!-I}$$

3.1.2 O gerenciador das máquinas virtuais

É através do gerenciador de máquinas virtuais que se pode definir novas máquinas virtuais, inicializá-las de acordo com os recursos disponíveis no sistema ou interrompê-las para gerar novos recursos para o sistema. A interface de gerenciamento de máquinas virtuais do programa é ilustrada na figura (3.2).

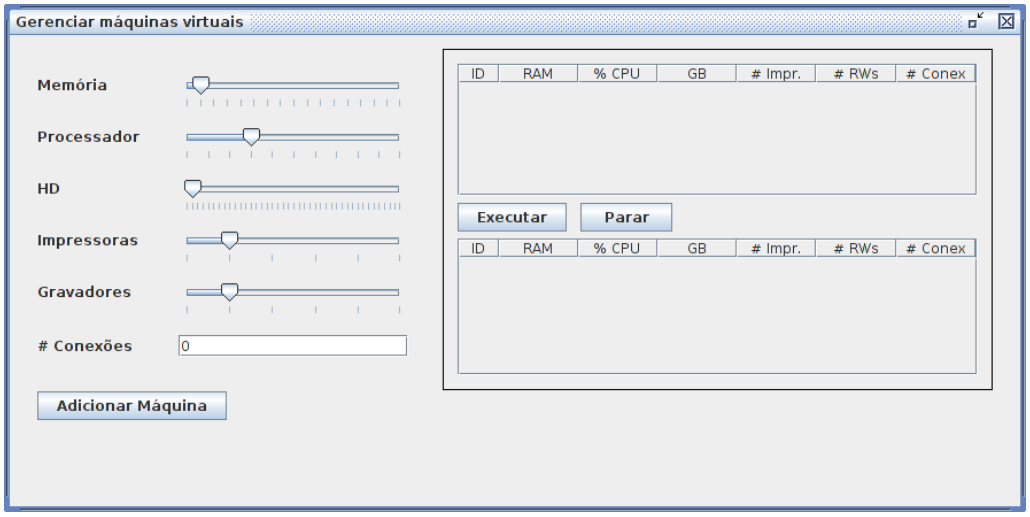

Figura 3.2: Tela do gerenciador de máquinas virtuais.

À esquerda é possível definir as características de uma máquina virtual. Neste momento, não há limite para a definição de uma máquina. Após definir os dados da máquina, basta clicar em *Adicionar máquina* para adicioná-la à tabela superior, onde se acomodam as máquinas disponíveis. Dentre as máquinas disponíveis, o usuário pode escolher quais deseja inicializar utilizando o botão *Executar*. Enquanto uma máquina está rodando, é possível suspender o seu uso através do botão *Parar*.

No momento em que uma máquina solicita permissão para rodar, ocorre a verificação dos recursos. Se todos os recursos demandados pela máquina estiverem disponíveis, a máquina os consumirá e o sistema terá à sua disposição uma nova máquina virtual como recurso. Os recursos do sistema serão automaticamente atualizados e podem ser conferidos no monitor do sistema. No momento que uma máquina tem sua execução interrompida, esta é consumida e os recursos que ela possuía passam a integrar o sistema novamente.

Seja φ o conjunto dos recursos necessários para um conjunto de máquinas Δ rodar. Se houver φ disponível, pode-se consumi-los para produzir as máquinas de Δ. A inserção do operador linear é representada da seguinte forma:

$$\frac{\langle\varphi\rangle \vdash \Delta}{\vdash \langle\varphi\rangle \multimap \Delta} \multimap -I$$

De forma análoga, pode-se interromper as máquinas Δ, consumindo-nas, para produzir φ. A representação, também pela inserção da inferência linear, é a seguinte:

$$\frac{\langle\Delta\rangle \vdash \varphi}{\vdash \langle\Delta\rangle \multimap \varphi} \multimap -I$$

3.2 Uma interpretação linear do sistema

Como foi explicado na seção 2.6, a lógica linear vê os objetos do raciocínio lógico, as premissas, como recursos. Em um sistema linear, ao utilizar determinados recursos, eles são consumidos e não estão mais à disposição. A aplicação desenvolvida é um simulador de máquinas virtuais que trabalha com concorrência computacional, onde várias *threads* estarão sendo executadas simultaneamente. Em computação concorrente é comum que haja disputas por recursos, uma vez que dois processos não podem utilizar um mesmo recurso concomitantemente. É nestas situações que a lógica linear encontrou mais adeptos, como em (SHI; XI, 2009; CERVESATO; SCEDROV, 2009; BEFFARA, 2004; KAHRAMANOGULLARı, 2009; AL., 2005; NETO, 1996). Girard descreve a lógica linear como uma primeira tentativa de resolver o problema da sincronização no paralelismo em um nível lógico, através do fragmento multiplicativo da lógica linear (GIRARD, 1986). Para um maior aprofundamento no assunto, devem ser consultados os trabalhos citados previamente neste parágrafo.

Como a lógica linear permite uma única referência a objetos que não estejam marcados com o exponencial !, pode-se utilizar uma abordagem de atualização destrutiva *in loco* (ALAGI; CSöRNYEI; VARGA, 2010), ou um valores de tipo linear (WADLER, 1990), que somente possuem uma referência, eliminando a necessidade de um sistema de *garbage collector*.

Girard, em (GIRARD, 1986) citou com bastante entusiasmo as possibilidades de se empregar a lógica linear para programação concorrente, devido às vantagens advindas da adição dos operadores multiplicativos, pois estes permitem uma comunicação sem a necessidade de sincronização (ora são os operadores aditivos que requerem sincronização, uma vez que dependem de uma escolha). Isto torna os programas mais previsíveis e o sucesso da comunicação dependente do fato que se pode ver os programas como provas de algo.

Outra vantagem da visão linear de um sistema é que há ferramentas que permitem expressar a manipulação de estados, tão importante em boa parte dos sistemas computacionais atuais (REDDY, 1993; GIRARD, 1986). Na computação concorrente, a ideia é que todos os processos estarão atuando em um mesmo estado, cada um podendo realizar alterações locais. Para um maior aprofundamento no assunto, Cervesato e Scedrov elaboraram um trabalho sobre concorrência baseada em processos e estados (CERVESATO;

SCEDROV, 2009).

Um exemplo de passagem de um estado para outro ocorre ao inicializar uma máquina virtual. Neste momento, recursos que antes estavam disponíveis são consumidos para gerar outros recursos, neste caso, a máquina virtual. Assim que essa máquina for desativada, consome-se ela, como recurso, e o produto é o mesmo utilizado para inicializá-la. O sistema volta ao estado inicial.

Cada máquina virtual deve ter, no mínimo, memória RAM, alguma porcentagem do processador, algum espaço em disco, uma impressora ou um gravador e permissão para estabelecer tantas conexões quanto desejar. Dessa forma, pode-se demonstrar os operadores da lógica linear e expressar operações das lógicas clássica e intuicionista, através do uso do exponencial !.

Pode-se reconhecer claramente alguns estados do programa desenvolvido. São estes:

- Estado inicial do programa. Nenhuma máquina virtual rodando;
- Uma máquina virtual consome recursos e começa a rodar. O programa sofre alterações no seu estado inicial e passa a dispor de menos recursos;
- Várias máquinas virtuais rodando. O programa esgota seus recursos e se encontra em um estado onde nenhuma outra máquina pode rodar;
- Uma ou mais máquinas interrompem a execução. O produto delas é o que foi consumido para fazê-las rodar. O sistema passa a dispor de recursos para inicializar outras máquinas.

3.2.1 As operações da aplicação

O sistema conta com os seguintes recursos, em seu estado inicial:

- A: Memória: 16 u., cada uma representando 512 Megabytes;
- B: Processador: 10 u., cada uma representando 10% da CPU;
- C: Espaço em disco: 40 u., cada uma representando 50 Gigabytes;
- D: Impressoras: 5 u., cada uma representando 1 impressora;
- E: Gravadores: 5 u., cada uma representando 1 gravador;

- F: Conexões: infinitas u., cada uma representando 1 conexão.

A escolha destes recursos foi feita de forma arbitrária e pode ser alterada na aplicação, de modo a respeitar outras configurações desejadas. Para os fins deste trabalho, tais valores são suficientes, uma vez que não se deseja realizar uma comparação de desempenho.

Através desses recursos, pode-se montar máquinas com diversas especificações, cada uma com uma quantidade de recursos diferentes. Por exemplo, uma máquina M com 1024 Megabytes, que terá à sua disposição 30 $\%$ da CPU, 100 Gigabytes de espaço em disco, 1 impressora e permissão para estabelecer 5 conexões resultaria na seguinte inferência linear:

$$A \otimes A \otimes B \otimes B \otimes B \otimes C \otimes C \otimes D \otimes F \otimes F \otimes F \otimes F \otimes F \multimap M$$

Isso pode ser lido como: Consumindo 2 As, 3 Bs, 2 Cs, 1 D e 5 Fs se produz 1 M. A máquina gerou uma alteração local no estado da máquina e isso leva o sistema pra outro estado, com os seguintes recursos:

- A: Memória: 14 u., cada uma representando 512 Megabytes;
- B: Processador: 7 u., cada uma representando 10% da CPU;
- C: Espaço em disco: 38 u., cada uma representando 50 Gigabytes;
- D: Impressoras: 4 u., cada uma representando 1 impressora;
- E: Gravadores: 5 u., cada uma representando 1 gravador;
- F: Conexões: infinitas u., cada uma representando 1 conexão.

No momento que a máquina M for encerrada o resultado será o seguinte:

$$M \multimap A \otimes A \otimes B \otimes B \otimes B \otimes C \otimes C \otimes D \otimes F \otimes F \otimes F \otimes F \otimes F$$

Neste caso o sistema volta para um estado igual ao inicial, algo que poderia ser diferente em um caso diverso.

3.3 Utilizando a aplicação

Nesta seção serão descritos experimentos realizados com a aplicação e serão explicadas possíveis interpretações lineares.

O sistema parte do estado inicial, onde todos os recursos ainda estão disponíveis, como se pode constatar analisando o monitor do sistema na figura 3.3.

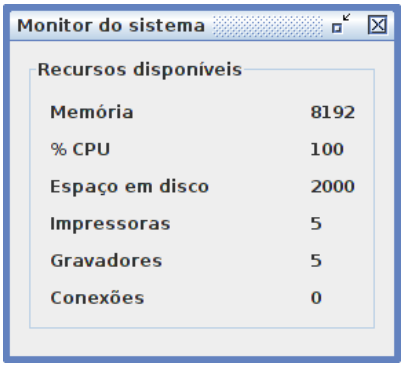

Figura 3.3: Estado inicial do sistema, todos recursos disponíveis.

Para este primeiro teste, duas máquinas foram criadas, conforme ilustrado na figura 3.4.

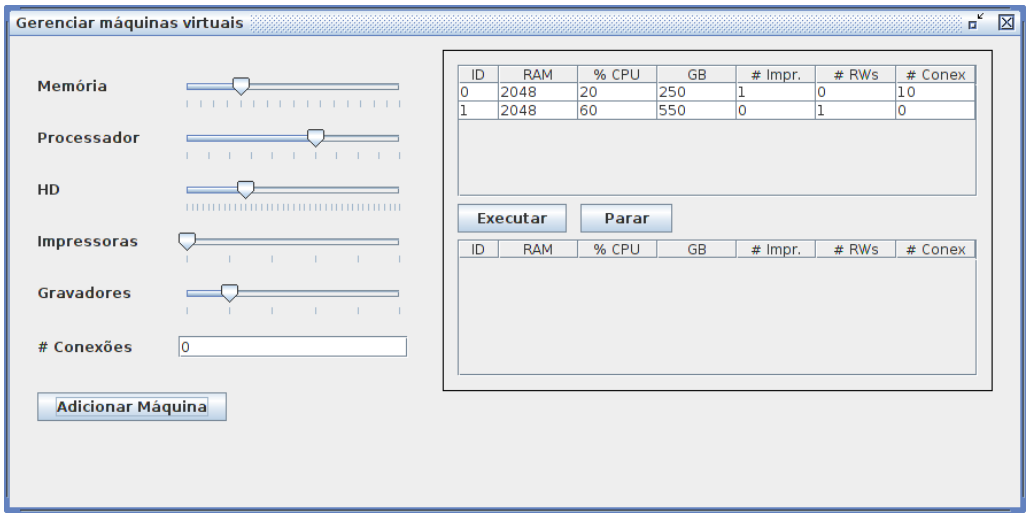

Figura 3.4: Duas máquinas criadas, nenhuma em execução.

Para este primeiro passo, nenhum recurso é utilizado. O que consta na tabela superior é apenas a descrição do tipo, introduzindo neste momento a sintaxe que será utilizada até o fim deste capítulo:

$$(A\&(A(\otimes A)^p)) \otimes (B\&(B(\otimes B)^p)) \otimes (C\&(C(\otimes C)^p)) \otimes (!F \otimes (D\&E)) \multimap MáquinaX$$

Onde as variáveis p podem assumir qualquer valor independentemente. Ou seja, o processo para descrever a produção de uma máquina X pode envolver o consumo de 1 ou mais unidades de memória, 1 ou mais unidades de processador, 1 ou mais unidades de espaço em disco, tantas conexões quanto desejado com ou uma impressora ou gravador, tudo à escolha do usuário. Esta fórmula, além de outras que não serão expostas neste texto, representa parte do comportamento da aplicação e é essencialmente o raciocínio linear aplicado, desenvolvido para este trabalho.

Após inicializar a máquina ID 1, tem-se os dados contidos na figura 3.5.

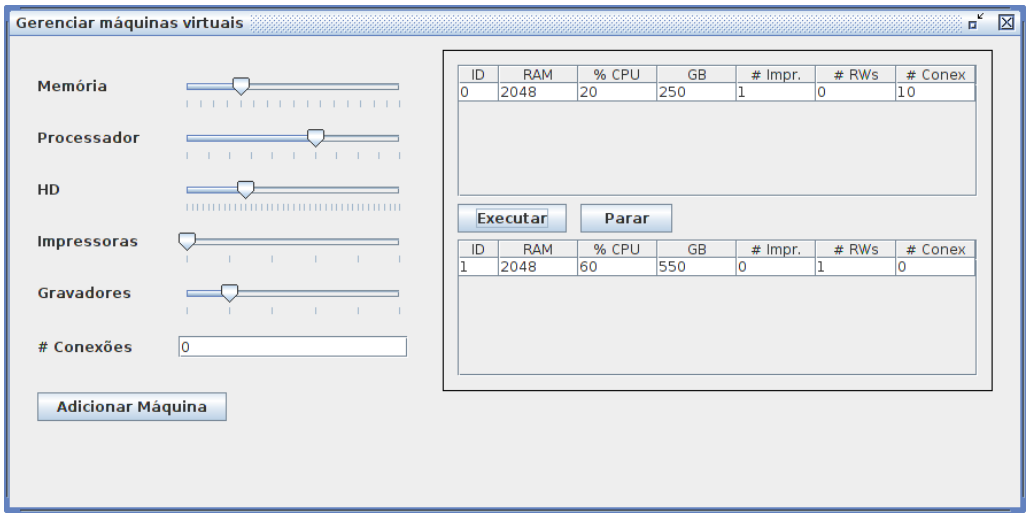

Figura 3.5: Uma máquina disponível e uma em execução.

Alteração local que pode ser expressa através da implicação linear:

$$(A(\otimes A)^3) \otimes (B(\otimes B)^5) \otimes (C(\otimes C)^{10}) \otimes E) \multimap Máquina\, ID\, 1$$

Outro modo de representar esta fórmula, desta vez na forma de uma dedução natural:

$$\frac{A^1 6, B^1 0, C^4 0, D^5, E^5, [F]}{A^4 \otimes B^6 \otimes C^1 1 \otimes E^1 \multimap Máquina\, ID\, 1}$$

A inicialização da máquina de ID 1 significa que recursos foram consumidos para gerá-la. O monitor do sistema neste instante confirma os recursos gastos, conforme a figura 3.6.

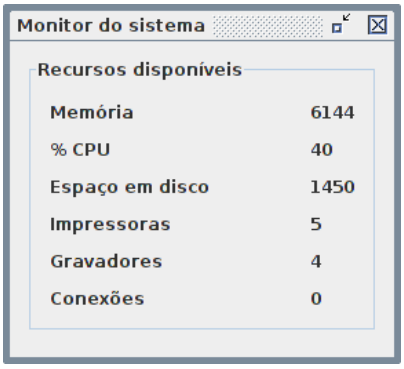

Figura 3.6: Estado do monitor do sistema após inicializar a máquina de ID 1.

Para utilizar mais recursos, criou-se e se inicializou uma terceira máquina. Após criar uma terceira máquina e inicializá-la, obtém-se o estado mostrado na figura 3.7.

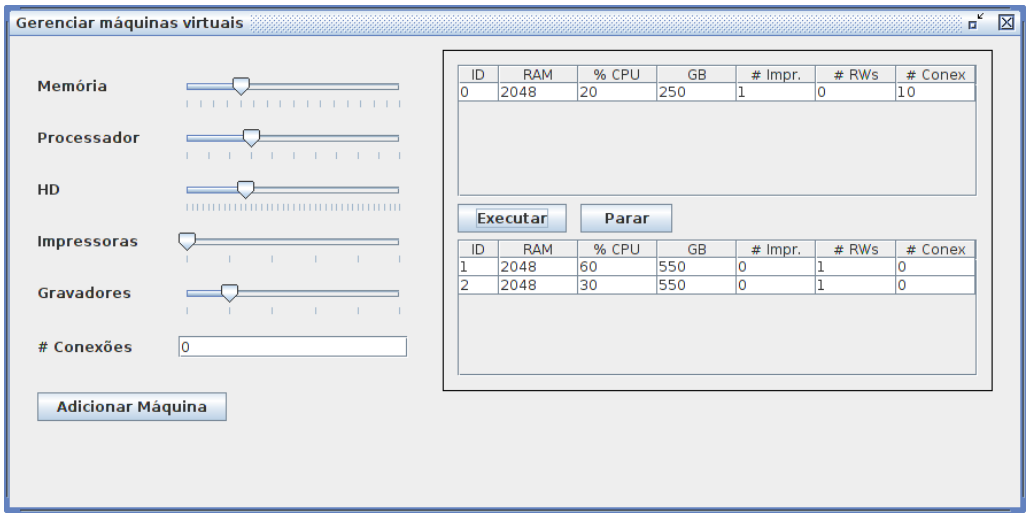

Figura 3.7: Uma máquina disponível e duas em execução (IDs 1 e 2).

De forma análoga, há uma alteração local que pode ser expressa através da implicação linear:

$$(A(\otimes A)^3) \otimes (B(\otimes B)^2) \otimes (C(\otimes C)^{10}) \otimes E) \multimap Máquina\, ID\, 2$$

Pode-se confirmar, através da análise do monitor do sistema neste instante, que os recursos estão praticamente esgotados, não há uma fatia de tempo de processador suficiente para o processo ID 0 rodar. A única máquina restante na tabela de máquinas disponíveis não pode ser executada, uma vez que depende do consumo de recursos não disponíveis no sistema. O monitor do sistema neste instante tem seus dados expostos na figura 3.8.

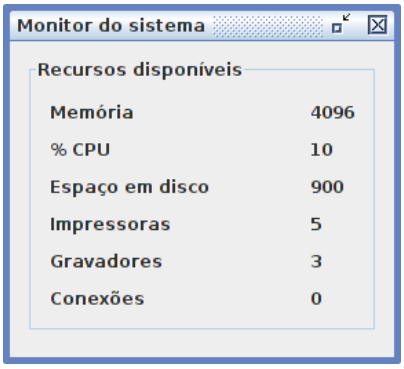

Figura 3.8: Estado do monitor do sistema enquanto há duas máquinas em execução (IDs 1 e 2).

Após fechar a máquina de ID 1, consume-se esta e se produzem os recursos consumidos para inicializá-la. O estado atual pode ser conferido na captura da tela neste instante na figura 3.9.

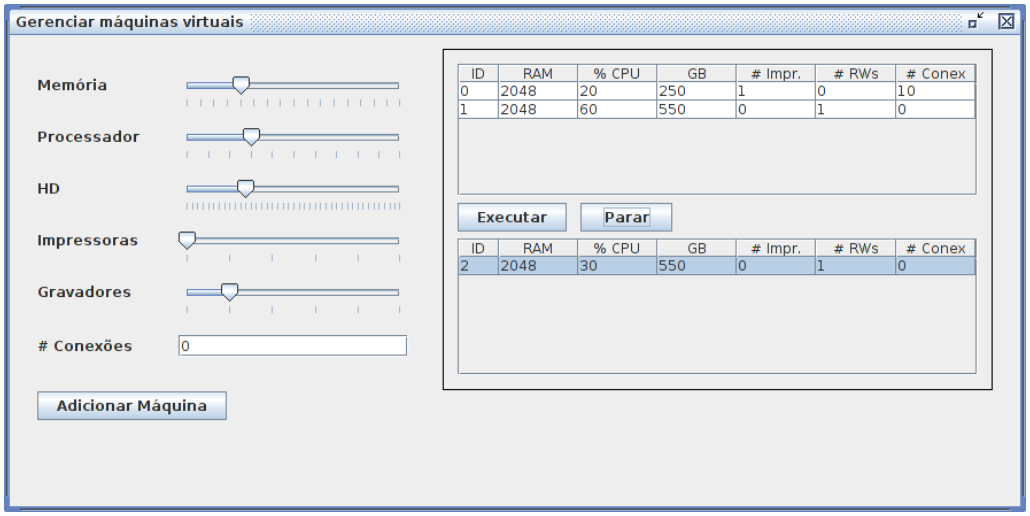

Figura 3.9: Duas máquinas disponíveis e uma em execução.

Operação que pode ser representada linearmente na forma:

$$Máquina\,ID\,1 \multimap (A(\otimes A)^3) \otimes (B(\otimes B)^5) \otimes (C(\otimes C)^{10}) \otimes E)$$

E na forma de uma dedução natural:

$$\frac{A^8, B^1, C^1 8, D^5, E^3, [F], Máquina\,ID\,1, Máquina\,ID\,2}{Máquina\,ID\,1 \multimap A^4 \otimes B^6 \otimes C^1 1 \otimes E^1}$$

A confirmação dos recursos do sistema pode ser obtida da análise do monitor do sistema neste instante, exposto na Figura 3.10.

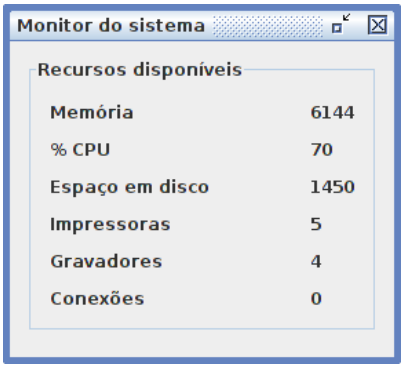

Figura 3.10: Estado do monitor do sistema após finalizar uma das duas máquinas que estavam em execução, restando somente a ID 2.

3.3.1 Análise dos resultados

Enquanto se utiliza os sistemas convencionais no desenvolvimento de uma aplicação, deve-se definir as regras que determinaram o seu comportamento e delimitar o seu funcionamento. Na lógica linear este procedimento é mais adequado ao mundo real. Basta definir as regras de inferência linear e todo o comportamento da aplicação estará atuando de forma consciente dos recursos e os atualizando automaticamente. Ao inicializar uma máquina virtual, os recursos necessários são utilizados na sua produção, não há uma referência para o que eles eram antes da produção, portanto não há necessidade de realizar procedimentos a fim de sanear a aplicação. Tudo isto resulta em uma maior segurança e praticidade.

O fragmento multiplicativo da logica linear não permite que decisões sejam feitas por alguém, tornando desnecessárias sincronizações. Sem a necessidade de sincronização, o desenvolvimento de aplicações concorrentes se torna mais fácil.

Com base em todas estas vantagens, percebe-se que, através da utilização da lógica linear, foi possível desenvolver uma aplicação concorrente de forma simples, sustentável e próxima ao mundo real. Este sistema lógico oferece uma boa alternativa aos convencionais sem perder o poder de expressividade. Compreendê-lo corretamente pode tornar o processo de desenvolvimento mais simples e a aplicação mais segura.

4 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho discutiu-se a lógica linear, suas características, operadores, seu sistema de dedução natural, algumas comparações com outros sistemas lógicos e como se pode modelar uma aplicação de forma linear. A criação deste sistema lógico trouxe grande entusiasmo para a área da ciência da computação e a partir deste trabalho pode-se conhecer alguns dos motivos para tal. Ilustrou-se brevemente como que estes elementos podem ser unidos, formando uma aplicação de modo linear. Desta abordagem, pôde-se obter diversas vantagens, como a possibilidade de focar no desenvolvimento da aplicação sem se preocupar com sincronização, uma vez que ao consumir recursos na produção de algo, não restam referências aos recursos iniciais e, portanto, não há como serem utilizados novamente de forma indevida. Isto pode representar uma aplicação mais segura e agilizar o processo de desenvolvimento, por não depender de um sistema de *garbage collector*. Outra vantagem é a proximidade do raciocínio ao processo de desenvolvimento da aplicação. Define-se o que se quer feito e não o contrário. Desta forma, reduz-se o esforço despendido em delimitar o comportamento que não se deseja da aplicação e se ocupa com o que a aplicação deve fazer. Espera-se que o presente trabalho possa servir de base para trabalhos vindouros, que visem focar num ou noutro aspecto que não pôde ser abordado em detalhes, uma vez que neste trabalho teve-se que partir de pontos bastante elementares.

Pode-se dizer que a lógica serve como um filtro para o raciocínio, permitindo que se garanta o bom raciocinar. A lógica linear lida com recursos, os quais tem um caráter limitado quantitativamente. Uma vez consumido, um recurso não mais existe. O profissional da computação está tão acostumados a utilizar a lógica clássica que nem se dá conta das barreiras que ela lhe impõe. Para expressar algoritmicamente situações onde há utilização de recursos, primeiro se estabelece as proposições e depois se passa a delimitá-las, lapidá-las de forma a torná-las da forma que se deseja. Capturando somente o que se quer

do problema a ser computado. Assim o é no caso dos recursos e assim o é no mundo real. Pôde-se observar que a lógica linear, por outro lado, oferece uma aproximação ao o mundo real bastante adequada ao programador.

A lógica linear oferece uma aplicação mais realista, baseada em estados e consciente dos recursos limitados. Tudo contribui para que não haja preocupação com sincronização e saneamento do que não é mais utilizado no programa, permitindo um maior controle da aplicação pelo programador. As facilidades de utilizar-se de um raciocínio linear, o qual a lógica linear possibilita, podem ser observadas na análise da aplicação desenvolvida no presente trabalho. Nesta, teve-se apenas que definir quais eram os recursos disponíveis e quais as provas das inferências lineares que descreviam o comportamento da aplicação. Os estados descritos são mera consequência evidente da aplicação das inferências neste ou naquele caso. Ao consumir boa parte dos recursos do sistema, é natural que se estará em um estado onde máquina virtual alguma possa vir a ser inicializada. Analogamente, ao terminar a execução de uma dada máquina virtual, é evidente que os recursos antes consumidos se farão novamente disponíveis. É assim no mundo real e é a este que a lógica linear tenta aproximar-se.

Por fim, deve-se ver a lógica linear não como uma tentativa de substituir os outros sistemas lógicos, mas complementá-los. A utilidade de algo está no seu emprego. Ao utilizar a lógica linear com os exponenciais (! e ?), pode-se torná-la tão expressiva quanto as lógicas clássica e intuicionista, permitindo tratar de todos os problemas que já eram resolvidos com os outros sistemas. Da análise dos trabalhos pesquisados foi possível observar que este sistema, ainda quando não utilizado diretamente, em uma linguagem de programação, tem seus conceitos já estão difundidos implicitamente em diversas áreas - como no paralelismo e sistema de tipos.

Tendo os conceitos da lógica linear expostos e uma aplicação desenvolvida para verificar a aplicabilidade deste sistema linear, pode-se expor algumas sugestões de trabalhos futuros. Como uma das possibilidades, a utilização da lógica linear em sistemas de tipos pode ser bastante vantajosa. Outra necessidade que se pode notar é a falta de uma linguagens de programação recente utilizando lógica linear, para favorecer a produção linear. Por fim, há a carência de um estudo comparativo de aplicações concorrentes utilizando o raciocínio linear em contraste com outras abordagens, a fim de observar qual oferece maior desempenho.

REFERÊNCIAS

AL., E. G. P. et. **Lógica Matemática, teoria da prova e teoria de funções**. Disponível em: http://www.mat.ufmg.br/∽elaine/papers/projeto_pesquisa.pdf. Acesso em: outubro de 2010.

AL., P. L. et. Monadic concurrent linear logic programming. **PPDP'05**, Lisboa, 2005.

ALAGI, G.; CSöRNYEI, Z.; VARGA, K. P. Parallel programming techniques and linear type systems. **8th Joint Conference on Mathematic and Computer Science**, Komárno, 2010.

ANDREOLI, J.-M.; CASTAGNETTI, T.; PARESCHI, R. Abstract Interpretation of Linear Logic Programming. **ILPS'93**, Vancouver, 1993.

BAILLOT, P. From Proof-Nets to Linear Logic Type Systems for Polynomial Time Computing. **TLCA'07**, Paris, 2007.

BAILLOT, P.; HOFMANN, M. Type Inference in Intuitionistic Linear Logic. **PPDP'10**, Munique, 2010.

BEFFARA, E. A Concurrent Model for Linear Logic. **Eletronic Notes in Theoretical Computer Science**, Paris, 2004.

CARVALHO, O. de. **Arthur Schopenhauer - Como Vencer um Debate sem Precisar Ter Razão**. 1.ed. Rio de Janeiro: TOPBOOKS, 1997.

CARVALHO, O. de. A ciência contra a razão. **Diário do Comércio**, São Paulo, 2009.

CERVESATO, I.; SCEDROV, A. Relating State-Based and Process-Based Concurrency through Linear Logic. **OSD/ONR CIP/SW URI**, Annandale, 2009.

FERNANDES, F. L. **O Isomorfismo de Curry-Howard via Teoria de Categorias**. Disponível em: http://www.mat.ufmg.br/∽pgmat/teses/Diss166.pdf. Acesso em: outubro de 2010.

FREIRE, C. M. Lógicas modais e complexidade descritiva. **SBC**, Casadinho, 2008.

GENSLER, H. J. **Introduction to Logic**. 1.ed. London: Routledge, 2002.

GIRARD, J.-Y. Linear Logic. **Theoretical Computer Science volume 50**, S.l., 1986.

GIRARD, J.-Y. Linear Logic, its syntax and semantics. **Cambridge University Press**, S.l., 1995.

GNU Lesser General Public License. Disponível em: http://www.gnu.org/licenses/lgpl.html. Acesso em: outubro de 2010.

GROHE, M. Finite Variable Logics In Descriptive Complexity Theory. **Bulletin of Symbolic Logic**, S.l., 1998.

HALPERN, Y. Y.; HARPER, R. On the Unusual Effectiveness of Logic in Computer Science. **Workshop on the Unusual Effectiveness of Logic**, Anaheim, CA, 2001.

HODAS, J. S. Lolli: an extension of λprolog with linear logic context management. **Workshop on the λProlog Programming Language**, Filadélfia, 1992.

HODAS, J. S.; WATKINS, K.; TAMURA, N.; KANG, K.-S. Efficient Implementation of a Linear Logic Programming Language. **International Conference and Symposium on Logic Programming 1998**, Claremont, 1998.

HUTH, M.; RYAN, M. **Lógica em Ciência da Computação - Modelagem e Argumentação sobre Sistemas**. 2.ed. Rio de Janeiro: LTC Editora S.A., 2004.

KAHRAMANOGULLARı, O. On Linear Logic Planning and Concurrency. **Information and Computation**, London, 2009.

LAGO PEREIRA, S. do. **Lógica Proposicional**. Disponível em: http://www.ime.usp.br/∽slago/ia-2.pdf. Acesso em: outubro de 2010.

LINCOLN, P. Linear Logic. **SIGACT**, S.l., 1992.

MANNA, Z.; WALDINGER, R. **The Logical Basis for Computer Programming**. 1.ed. Massachusetts: Addison-Wesley Professional, 1985.

MARTINS, L. R. Dedução Natural e Normalização Fraca para Lógica Linear Completa. **Mestrado em Ciência da Computação - UFC**, Fortaleza, 2003.

MELLIèS, P.-A. **Categorical Semantics of Linear Logic**. Disponível em: http://www.pps.jussieu.fr/∽mellies/papers/panorama.pdf. Acesso em: novembro de 2010.

MORTARI, C. A. **Introdução à lógica**. 1.ed. São Paulo: Editora UNESP, 2001.

NETO, A. G. S. S. Using Logic for Concurrency: a critical study. **Dissertação de Mestrado - UFPE**, Pernambuco, 1996.

PINTO, J. S.; FRADE, M. J. Introdução ao Sistema Coq de Assitência à Prova. **DI/CCTC - Publicações Pedagógicas**, [S.l.], 1998.

POWER, J.; WEBSTER, C. Working with Linear Logic in Coq. **12th International Conference on Theorem Proving in Higher Order Logics**, Nice, 2001.

PRATT, V. Linear Logic complements Classical Logic. **Linear Logic '96**, Tokyo, 2005.

PUC-RIO. Disponível em: http://www.inf.puc-rio.br/. Acesso em: outubro de 2010.

REDDY, U. S. **A Linear Logic Model of State**. Disponível em: http://www.cs.bham.ac.uk/∽udr/papers/state.full.ps.gz. Acesso em: novembro de 2010.

REEVES, S.; CLARKE, M. **Logic for Computer Science**. 1.ed. London: Addison-Wesley Publishers Ltd., 2003.

REIS, M. C. G. dos. **Aristóteles - De Anima**. 1.ed. São Paulo: Editora 34, 2006.

SHI, R.; XI, H. A Linear Type System for Multicore Programming. **SBLP'09**, Gramado, Brasil, 2009.

THE Coq Proof Assistant. Disponível em: http://coq.inria.fr/. Acesso em: outubro de 2010.

UFPE. Disponível em: http://www2.cin.ufpe.br/site/index.php. Acesso em: outubro de 2010.

UFPEL. Disponível em: http://inf.ufpel.edu.br/. Acesso em: outubro de 2010.

UFRGS. Disponível em: http://www.inf.ufrgs.br/. Acesso em: outubro de 2010.

UFRJ. Disponível em: http://www.dcc.ufrj.br/quadro-curricular.html. Acesso em: outubro de 2010.

UFSM. Disponível em: http://www.inf.ufsm.br/. Acesso em: outubro de 2010.

UNICAMP. Disponível em: http://www.ic.unicamp.br/. Acesso em: outubro de 2010.

USP. Disponível em: http://www.icmc.usp.br/∽scc/. Acesso em: outubro de 2010.

WADLER, P. Linear types can change the world! **IFIP TC2**, Israel, 1990.

WADLER, P. A taste of linear logic. **Mathematical Foundations of Computer Science**, Glasgou, 1993.

XAVIER, F. A.; OLIVEIRA, M. O. M. de Menezes e. **Lógica Linear e Aplicações em Ciência da Computação**. Disponível em: aquilesburlamaqui.wdfiles.com/local–files/logica-aplicada-a-computacao/texto_linear.pdf. Acesso em: outubro de 2010.

XI, H.; ZHU, D.; LI, Y. Applied Type System with Stateful Views. **BUCS-2005-03**, Boston, 2004.