UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

HELMUT WILLY KNOBLAUCH

QUADRO COMPARATIVO DOS MÉTODOS DOS TABLEAUX E DAS RESOLUÇÕES

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Professor Dr. Jorge Muniz Barreto

Florianópolis, dezembro de 2001

QUADRO COMPARATIVO DOS MÉTODOS DOS TABLEAUX E DAS RESOLUÇÕES

Helmut Willy Knoblauch

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação, área de concentração Sistemas de Conhecimento, e aprovada em sua forma final pelo Programa de Pós-graduação em Ciência da Computação da Universidade Federal de Santa Catarina.

Prof. Fernando A. O. Gauthier, Dr.
Coordenador do Programa

Banca Examinadora:

Prof. Jorge Muniz Barreto, D.Sc.A, INE-UFSC, Orientador

Prof. Mauro Roisemberg, Dr., INE-UFSC

Prof^{a}. Paulo Sergio da Silva Borges, Dr., INE-UFSC

Prof. Walter Celso de Lima, Dr., UDESC

Florianópolis, 13 de dezembro de 2001.

É preciso pensar para acertar,
calar para resistir, agir para vencer,
porque sucesso está um pouco além
de onde as pessoas comuns desistem.
(Anônimo)

À
minha família: esposa Miriam, Mônica,
Priscila,Marcos, Magda e ao genro
Nei Eduardo Knob

Fica registrado este reconhecimento especial:
A Deus, a quem devo não só a existência, mas tudo que tenho e sou;
Ao Prof. Dr. Jorge Muniz Barreto, discípulo do SCHREYER,
incansável pesquisador, solidário, enriquecedor de vidas,
amante da natureza e amigo, como orientador;
Aos professores doutores do curso de Ciência em Computação
por ampliarem a minha visão do mundo computacional vasto
e ao mesmo tempo, fazendo a especificação de cada área
e apontando as possibilidades de pesquisa;
Aos colegas do curso de mestrado de Ciência em Computação
pelo companheirismo, apoio e ajuda;
À Universidade Federal de Santa Catarina e á Universidade
do Contestado pela promoção do curso de Ciência em Computação.
À mestre Nádia Grezzana Mascelani e aos funcionários da
Pós-Graduação pela gentileza e carinho de atendimento;
Ao Coordenador do Curso de Bacharelado de Informática,
Moacir Kichel bem como à Diretora Acadêmica Elisete Maria Pedott
pelo encaminhamento e incentivo;
À banca examinadora pela avaliação, conselhos finais e elogios.

SUMÁRIO

Lista de figuras

QUADRO COMPARATIVO DOS MÉTODOS DOS TABLEAUX E DAS RESOLUÇÕES

Candidato: Helmut Willy Knoblauch

Orientador: Professor Dr. Jorge Muniz Barreto

Resumo

Expõe que ambos os métodos, tanto dos Tableaux como das Resoluções fazem parte daquilo que se chama prova automática de teoremas. O método da Resolução trabalha com prova por refutação: "se a negação de um teorema é falsa, então ele será verdadeiro." Este método é baseado em uma regra de inferência única, chamada regra de resolução, e utiliza intensivamente um algoritmo de casamento de padrões chamado algoritmo de unificação.

Faz-se algumas comparações entre os dois métodos, pois, o algoritmo dos Tableaux não necessita de transformações para qualquer forma normal. Revela que a principal vantagem do método dos Tableaux, em relação, ao método tradicional e conhecido de Resolução é fato deste evitar utilização de cláusulas nas suas formas.

QUADRO COMPARATIVO DOS MÉTODOS DOS TABLEAUX E DAS RESOLUÇÕES

Candidato: Helmut Willy Knoblauch

Orientador: Professor Dr. Jorge Muniz Barreto

Abstract

It exposes that both methods, so much of Tableaux as of the Resolutions they are part of that that calls her automatic test of theorems. The method of the Resolution works with test for refutation: " if the denial of a theorem is false, then he will be true ". This method is based on a rule of only inference, call resolution rule, and it uses an algorithm intensity of marriage of patterns called unification algorithm.

It make some comparative between both methods, because, the algorithm of Tableaux doesn't need transformations for any normal form. Reveals that the main advantage of the method of Tableaux, in relationship, to the traditional and well-known method of Resolution is fact of this to avoid use of clause forms .

INTRODUÇÃO

Quando se fala em robôs inteligentes, em controle genético de quase todas as doenças, em ciência baseada no microchip, no satélite e na fibra óptica, em integração do computador com o cérebro humano, nas máquinas moleculares, na inteligência artificial e na nanotecnologia, quando buscam implantar clones de neurônios humanos em sistemas robóticos, o que resta para a inteligência humana?

Mentes criativas como Marvin Minsky, Rodney Brooks, Douglas Lenat, tentam produzir um robô ao qual se possa pedir: "Você tem alguma idéia de como resolver isto"? Na verdade estão querendo produzir uma máquina que simule comportamento inteligente.

Está certo que uma equipe busca a construção deste robô através de um caminho conhecido por "topdown" (de cima para baixo), tentando criar o Cog (de "cognition"), cuja palavras iniciais possam ser: "Cogito, ergo sum" ("Penso, logo existo", que são palavras imortais do filósofo Renè Descartes); a outra equipe por sua vez usa a estratégia chamada "bottom-up" (de baixo para cima) tentando implantar sistemas lógicos.

Pensando em tudo isto, questiona-se: o que é inteligência? Acumular conhecimentos? Capacidade de trabalho? É inteligência fazer trabalhos rotineiros ou mecânicos? Quantas facetas possui a inteligência humana? Qual a parte que vai caber à inteligência artificial?

A Inteligência Artificial, talvez, possa ser definida como a capacidade de um sistema em adaptar seu comportamento (tomar decisões) a fim de alcançar seus objetivos em ambientes mutáveis, mas cabe ainda ao ser humano a criatividade. E dentro desta criatividade a lógica desempenha o papel fundamental. Como disse alguém: "A lógica, observada em [Barreto 1997] como estudo dos mecanismos do pensamento, possui um papel importante na Inteligência Artificial, sendo discutida desde os primórdios desta como um método de representação de conhecimento e modelagem de raciocínio."

Se acredita que a lógica é imprescindível na vida de qualquer especialista em programação. Pode ser que para o mercado de trabalho seja interessante apenas a

programação em si, ou seja a codificação, a validação e verificação e também a manutenção do software ou do programa, mas por detrás disto está a análise, especificação do problema e o projeto. Antes de começar a programar é indispensável não só ter o problema, mas também a solução do mesmo.

O que é fascinante é poder criar programas, programas com implementação cada vez mais simples e mais eficiente.

Qualquer instituição de informática precisa, sem dúvida alguma, trabalhar com paradigma O paradigma é que orienta e proporciona condições para que uma implementação seja bem sucedida.

É evidente, com justa razão, que a universidade se empenhe em enfatizar a bagagem intelectual e atitudes que vão ajudar os indivíduos nos próximos 50 anos, porque o profissional deve estar pronto a enfrentar quaisquer oscilações do mercado e não estar preso à uma tecnologia em particular, porque ela também é mutável, ainda mais, no processo de aprimoramento rápido no qual nós nos encontramos.

Pelo que tudo indica a automatização, utilizando sistemas especialistas, e a modelagem do raciocínio são as grandes linhas da informática neste novo milênio.

Portanto, visto que a automatização é um assunto cada vez mais presente no nosso mundo atual e, pelo fato, dela estar dependente da modelagem de raciocínio, porque são inter-relacionados, percebe-se que automatização precisa da modelagem do raciocínio, daí a a importância da lógica.

O objetivo ao longo desta exposição é traçar um comparativo entre dois métodos do raciocínio lógico: o método dos Tableaux e o método das Resoluções.

2. LÓGICA E LÓGICA COMPUTACIONAL

2.1. INTRODUÇÃO À LÓGICA

Poder apresentar uma boa argumentação é um dos principais requisitos exigidos para um bom advogado, matemático, detetive, escritor e para outros profissionais, principalmente, para um bom programador computacional.

Mas não precisamos ser nenhum desses profissionais para apresentar uma boa argumentação. Na vida diária fazemos afirmações e negações baseadas num certo encadeamento lógico. Por exemplo, Paulo só tira notas boas. Então ele vai passar de ano. Este argumento é válido.

Há argumentação válida e não válida. Por exemplo, uma argumentação válida: Todos os animais são mortais. O canário é um animal. O canário é mortal. Agora vejamos uma argumentação não válida: Todos os seres humanos são mortais. Alguns animais são mortais. Todos os animais são mortais. A conclusão é verdadeira, pois o sabemos por intuição. Há verdades que não precisam ser comprovadas, mas esta conclusão não é conseqüência das premissas, portanto, a argumentação não é válida.

Veja esta colocação de Lewis Carrol: Se você tivesse que comprar um relógio, qual escolheria: o que atrasa 1 minuto por dia, ou um outro que está completamente parado com as molas quebradas e cujos ponteiros marcam 4 horas? Qual é a sua decisão?

Pois uma resposta lógica para o problema pode surpreender. Lewis Carrol, que foi quem primeiro formulou esta questão, argumenta que o relógio parado é o mais eficiente. Acompanhe sua argumentação: "O relógio que atrasa 1 minuto por dia vai dar a hora exata a cada 2 anos, enquanto a hora marcada no relógio quebrado vai coincidir com a hora certa duas vezes ao dia".

Lewis Carrol é o pseudônimo usado por Charles Lutwidge Dogdson (1832-1898), professor de Lógica e Matemática da Universidade de Oxford, autor de vários livros, entre eles Symbolic Logic – The game of Logic [Carrol, 1881 e 1891]. Porém sua maior obra foi o clássico Alice no País das Maravilhas.

Algumas vezes depara-se com paradoxos. Por exemplo: Toda regra tem exceção. A proposição anterior é uma regra, logo tem exceção. Então, existem regras sem exceção. Proposições como essas são chamadas de paradoxos. Paradoxo é tudo aquilo que numa primeira abordagem parece verdadeiro, mas é falso, ou que parece falso mas é verdadeiro. Paradoxos sempre despertaram a curiosidade de pessoas das mais variadas culturas.

Há ainda aqueles que não são nem verdadeiros, nem falsos, como o do barbeiro que segue. Um dos mais famosos paradoxos da matemática é o Bertrand Russel (1872-1970), matemático e filósofo britânico. Diz o paradoxo: O barbeiro de Sevilha faz a barba de todos os sevilhanos, e apenas desses, que não se barbeiam a si próprios. O barbeiro de Sevilha pode barbear-se a si próprio? Suponha que sim. Então ele faz a barba de si próprio. Mas o barbeiro só faz a barba dos que não se barbeiam a si próprios.

Suponha, então, que ele não faz sua própria barba. Mas todos os que não se barbeiam a si próprios fazem a barba com o barbeiro de Sevilha, então ele faz a própria barba.

Este paradoxo é exemplo de um enunciado circular, que ora nega, ora confirma uma afirmação. Com o paradoxo do barbeiro, Bertrand Russel fez muitos matemáticos arrancarem os próprios cabelos e barbas, pois colocou em xeque toda uma teoria matemática que se produzia na época.

Epimênides um outro grande lógico afirmou: Todos os cretenses são mentirosos. Epimênides era cretense, pois, nascera na ilha de Creta. Teria ele dito a verdade? Epimênides de Cnossos, em grego, Epimenídes, poeta, filósofoe legislador grego, originário de Creta (século VI a C). Sua vida é semilendária, e atribuem-se-lhe muitas obras apócrifas; esteve em Atenas no tempo de Sólon. 'E considerado um dos fundadores do orfismo (Doutrina dos mistérios órficos: culto secreto de origem helênicas e que reapareceu nos séculos VII e VI a C. sob forma filosófica, pregando preceitos mais puros de moral e esperança na imortalidade feliz.

Enigmas e passatempos lógicos sempre despertaram a curiosidade de muitos. No livro cujo nome é: "Como se chama este livro?", de Raymond Smullyan, publicado em 1978, é uma coletânea de problemas que ativam nosso pensamento lógico.

O que é a lógica? Fala-se que há quatro modos da razão se referir à ordem. Há uma ordem, que a razão não estabelece mas unicamente constata: trata-se do domínio da Filosofia da Natureza e da Metafísica. Há outra ordem, que a razão, pela reflexão, estabelece nos atos da vontade: a Moral. Há uma terceira ordem, que a razão estabelece nas coisas externas que produz, relativo a Filosofia da Arte. A quarta ordem, que a razão, estabelece nos seus próprios atos: a Lógica. Portanto, a Lógica é a ciência da ordem a ser estabelecida nas operações da razão. Existe a Lógica natural e a Lógica científica. Esta, aprendida principalmente nos livros, é aperfeiçoamento daquela [Costa 1994].

Mas para que fim (causa final) estabelecer ordem nas operações da razão? Para alcançar a verdade.

É útil notar-se que a razão e a própria inteligência enquanto de uma verdade conhecida passa a outra desconhecida. A inteligência de modo intuitivo apanha as coisas, enquanto que a razão raciocina sobre elas. A inteligência por assim dizer é um ponto; a razão é o movimento de um ponto a outro. Na realidade é uma só faculdade com duas operações e com dois nomes distintos.

A operação pela qual a inteligência (que então se chama razão) passa de uma verdade para outra, recebe o nome de raciocínio. Exemplo: Todo catarinense é brasileiro. Ora, João é catarinense. Logo, João é brasileiro. As duas primeiras frases (chamadas premissas) constituem o antecedente; a conclusão, a terceira frase, é o conseqüente. A conseqüência está no nexo entre o antecedente e o conseqüente.

Por conseguinte, no raciocínio acima, de uma verdade: " Todo catarinense é brasileiro", mediante a engrenagem do mesmo raciocínio, passou a razão a esta outra verdade: "Logo, João é brasileiro. '

O coração, o centro da Lógica, é a operação própria da razão (não digo da inteligência), isto é, a operação pela qual o homem de uma coisa conhecida passa a outra desconhecida. E como esta operação se chama raciocínio, este é, propriamente o centro de toda a Lógica [Copi 1978].

É difícil dar uma definição precisa de Lógica. "Logic is the systematic study of the structure of propositions and of the general conditions of valid inference by a method which abstracts from the content or matter of the propositions and deals only with their logical form" Encyclopaedia Brittanica

A lógica desempenha importante papel como teoria matemática. Ela é básica como método de representação de conhecimento. É sistema formal simples que apresenta uma teoria semântica interessante do ponto de vista da representação do conhecimento. Ainda hoje grande parte da pesquisa em inteligência artificial está ligada direta ou indiretamente à Lógica.

Em resumo, a Lógica é a ciência ou a arte da ordem nas operações da razão para que o homem, facilmente e sem erro através de sua consciência, atinja a verdade ou aquilo que é real.

2.2. NOTA HISTÓRIA

A Lógica Clássica resultou de séculos de reflexões sobre o problema lógico, remontando aos primórdios do pensamento filosófico grego (particularmente a Parmênides de Eléia, que formulou, pela primeira vez, o princípio de identidade, e a seu discípulo Zenão, que, inaugurando a dialética como argumentação combativa ou erística [polêmica contra os pitagóricos], ao mesmo tempo foi o primeiro a recorrer ao raciocínio por absurdo). Quem foram os pitagóricos? "O quadrado da hipotenusa é igual à soma dos quadrados dos catetos, ou: $a^2 = b^2 + c^2$. Quem passou o primeiro grau e não ouviu este teorema de Pitágoras? 2500 anos nos separam do autor deste cálculo. Um teorema ainda usado após 25 séculos depois de sua morte. Quem foi Pitágoras?

Um grego, matemático, um astrônomo, mas Pitágoras foi mais que isso: conhecia também música, moral, filosofia, geografia e medicina.

Pitágoras viveu há 2500 anos (580-497 a. C.) e não deixou obras escritas. O que se sabe de sua biografia e de suas idéias é uma mistura de lenda e história real. A lenda começa antes mesmo de Pitágoras nascer, por volta de 580 a.C., a sacerdotisa do deus Apolo disse a um casal que vivia na ilha de Samos, no mar Egeu: "Tereis um filho de grande beleza e extraordinária inteligência; será um dos homens mais sábios de todos os tempos". No mesmo ano, o casal teve um filho. Era Pitágoras.

O jovem assombrava os mestres das melhores escolas de Samos: não conseguiam responder as perguntas do moço de 16 anos. Nessas condições, só havia uma coisa a fazer: despachá-lo a Mileto, para que estudasse com Tales - o maior sábio da época, provavelmente o primeiro grego a se dedicar cientificamente aos números.

Quando adulto, Pitágoras resolveu somar ao seu conhecimento, além dos números, idéias sobre a ciência e a religião de outros povos. Acreditando que era preciso ver para crer, arrumou as malas e disse "até logo" a seus patrícios: foi à Síria, depois à Arábia, à Caldéia, à Pérsia, à Índia e, como última escala, ao Egito, onde passou mais de 20 anos e se fez até sacerdote para melhor conhecer os mistérios da religião egípcia. Dizem que quando Cambises conquistou o Egito, Pitágoras foi levado em cativeiro para a Babilônia. Curioso como era, o grego aproveitou a chance para descobrir em que pé andavam as ciências naquele país.

Depois dos 50 anos, seu desejo era voltar a Samos e abrir uma escola. Mas Samos tinha mudado e o ditador Polícrates, que governava a ilha, não queria saber nem de escolas nem de templos. Aí Pitágoras seguiu adiante, à Crotona, no sul da Itália, onde as melhores famílias da cidade lhe confiaram prazerosamente a educação de seus filhos. E Pitágoras pôde, por fim, fundar sua escola, onde passou a ensinar aritmética, geometria, música e astronomia. E, permeando essas disciplinas, aulas de religião e moral. Mais que uma escola, Pitágoras conseguira criar uma comunidade religiosa, filosófica e política. Os alunos que formava saíam para ocupar altos cargos do governo local; cientes de sua sabedoria torciam o nariz antes as massas ignorantes e apoiavam o partido aristocrático. Resultado: as massas retrucaram pela violência e - segundo dizem uns - incendiaram a escola, prenderam o professor e o mataram. Outros são mais otimistas: contam que Pitágoras foi só exilado para Metaponto, mais ao norte, na Lucânia, onde morreu, esquecido mas em paz, com mais de 80 anos de idade.

Pitágoras imaginava os números como pontos, que determinam formas. E o Universo, o que é, senão um conjunto de átomos, cuja disposição dá forma à matéria?

De qualquer modo, Pitágoras não se contentava em dizer frases; demonstrou que era necessário provar e verificar geometricamente um enunciado matemático, ou seja, expressá-lo como teorema. E formulou vários, além daquele mais conhecido. Por exemplo: a soma dos ângulos internos de um triângulo é igual a soma de dois ângulos retos ($a+b+c=180°$); a superfície de um quadrado é igual a multiplicação de um lado por si mesmo. Donde a expressão "elevar ao quadrado": $2x2=2^2$; o volume de um cubo é igual à sua aresta multiplicada três vezes por si mesma: $2x2x2=2^3$, o que originou a expressão "elevar ao cubo".

Pitágoras também mostrou que música e matemática são parentes: o comprimento e a tensão das cordas de uma lira, por exemplo, podem ser convertidos em expressões matemáticas.

O gênio de Samos era um homem religioso, acreditava na transmigração da alma: quando um homem morre, sua alma passa para outro ou para um animal. Só pela vida "pura" a alma poderia libertar-se do corpo e viver no céu. E vida pura significava, para Pitágoras, austeridade, coragem, piedade, obediência, lealdade. Dizia a seus alunos: "Honra os deuses sobre todas as coisas. Honra teu pai e tua mãe. Acostuma-te a dominar a fome, o sono, a preguiça e a cólera". Mas acreditava igualmente numa série de superstições: não comer carne por causa da reencarnação, não comer favas, não atiçar o fogo com ferro, não erguer algo caído do chão.

Melhor meio de purificar a alma, ensinava Pitágoras, era a música. O Universo - afirmava - era uma escala, ou um número musical, cuja própria existência se devia à sua harmonia.

Como astrônomo, seu principal mérito foi conceber o Universo em movimento. Como teórico de medicina, achava que o corpo humano era constituído basicamente por uma harmonia: homem doente era sinal de harmonia rompida. Como filósofo, deu origem a uma corrente que se desenvolveu durante os séculos seguintes, inspirando - entre os principais pensadores gregos - inclusive o famoso Platão.

Pitágoras, em gr. Pythagóras, filósofo grego do séc VI a C (Samos primeira metade do séc. VI a.C. - Metaponto, inÍcio do séc. v a.C.). Teria emigrado para a Sicília, para ali fundar espécie de ordens filosóficas e políticas. Provocou um movimento religioso e científico, o pitagorismo: sua metafísica ensina a metempsicose e a purificação da alma pelo conhecimento; uma doutrina esotérica fala da realidade por trás do mundo visível, que é de essência matemática essas teorias seriam conforme a lenda, de origem egípcia. Pitágoras não deixou nenhum escrito. Alguns de seus discípulos, os "matemáticos", estabeleceram numerosos teoremas, postos em ordem por Euclides, no III séc. a.C. Demonstraram, em particular, a incomensurabilidade da diagonal em relação ao lado do quadrado, descobrindo assim um limite intransponível ao uso dos números racionais. Estudaram a estrutura dos números e das progressões aritméticas, procurando definir os números "perfeitos", números iguais à soma de seus divisores. Atribuiu-se a Pitágoras o teorema do quadrado da hipotenusa, e o

estabelecimento da tábua de multiplicação e um estudo das escalas (escala de Pitágoras).

O pitagorismo considera como essencial o papel do número na natureza. Por seu matematicismo sistemático contribuiu para a formação do racionalismo ocidental, e, por sua mística dos números, ligou-se á magia e às filosofias esotéricas. Grande parte de nossas informações sobre Pitágoras e seus discípulos é de segunda mão e de natureza lendária; é também duvidoso se se deve a Filolao um sistema heliocêntrico do Universo ou se astrônomos posteriores invocaram para tanto a autoridade de um discípulo de Pitágoras. Com o neopitagorismo dos sécs. III e IV d.C., Pitágoras nada tem mais em comum que o nome.

Grande importância, nessa fase preparatória da lógica clássica, tiveram também os sofistas (pela análise a que submetem a linguagem), Sócrates (considerado o iniciador da filosofia dos conceitos) e Platão (em cuja dialética de índole marcadamente matemática se dá à criação do procedimento da divisão dos gêneros em espécies, que vão surgindo, dicotomicamente, através de séries paralelas [p. ex. números racionais, divididos em frações e números inteiros, estes por sua vez divididos em pares e ímpares etc.]. Aristóteles. no Organon [tradução em inglês pela editora Encicplopédia Britânica no Great Books], sistematizou a lógica clássica, estudando os elementos do discurso (conceitos, juízos, raciocínios) e desenvolvendo uma lógica na qual o raciocínio dedutivo se baseava no silogismo. Sua lógica permaneceu, todavia, comprometida com os pressupostos gerais de sua filosofia: a) os conceitos constituídos por abstração, a partir do plano empírico (o que confere ao plano lógico "inerência" ao real imediato, tendendo a justificar seu status); b) as essências da realidade entendidas como fixas e situadas numa hierarquia imutável, na qual a mudança só interfere acidental e parceladamente, isto é, dentro do âmbito de alteração possível de cada essência, sem quebra da rígida distribuição dos seres em "lugares naturais" (reflexo da noção biologista das espécies fixas que Aristóteles opunha ao biologismo evolucionista, de inevitáveis repercussões políticas, de um Anaximandro ou um Empédocles); c) a construção do plano lógico a partir somente da linguagem corrente, cuja logicidade inerente é aceita, apenas depurada, sem maior apelo a crítica dessa linguagem através da aceitação (como em Platão) da superioridade, para a ciência, da linguagem constituída pelos "artefatos" matemáticos. Ainda na Antiguidade grega, a escola socrática de

Mégara apresentou grandes lógicos, como Eubúlides de Mileto, Diodoro Cronos e Filon de Larissa, críticos do aristotelismo que denunciaram, através de paradoxos (ex. o de Epimênides), dificuldades internas da lógica. Herdeiros da tradição aristotélica (por via de Teofrasto) e da megárica, os estóicos criaram, no período helenístico, uma lógica original, a primeira lógica proposicional. Na Idade Média, a escolástica fez amplo uso do método silogístico de exposição. Para a lógica, o período criador começou com Abelardo (1079-1142), que estabeleceu os primeiros fundamentos da lógica das conseqüências. Os lógicos medievais aperfeiçoaram a silogística, abordaram importantes problemas de dupla quantificação ao de classe vazia e mesmo de lógica das relações. A lógica medieval distingue-se da antiga porque extrai da língua então falada (o latim) leis e regras que a regem, levando em consideração todas as funções semânticas e sintáticas dos signos, apresentando-se, em conseqüência, quase inteiramente, como uma metalinguagem. A formalização apresentada pela lógica medieval foi julgada excessiva e estéril, mais tarde, por Ramus, Descartes e Bacon. Dentro da tradição cartesiana, o grupo de Port-Royal, fazendo desaparecer as sutilezas lógicas dos últimos escolásticos, tentaria salvar a lógica do descrédito em que tombara. Atualmente, as teorias lógicas se diversificam em várias correntes, sendo as principais:

1) a corrente que permanece fiel à tradição lógica clássica, bivalente, normativa e estreitamente ligada a uma metafísica essencialista (ex. corrente neotomista);

2) as múltiplas correntes da lógica simbólica, que consideram a lógica uma ciência positiva, rejeitando-lhe caráter normativo, recusando qualquer ingerência da metafísica e recorrendo metodicamente a expressão simbólica e formal, sendo seus meios de expressão retirados da linguagem matemática;

3) a corrente da lógica husserliana, na qual reaparece, dentro dos quadros da fenomenologia, a vinculação da lógica a metafísica;

4) a lógica dialética, fundamentada na noção de contradição, desenvolvida por Hegel dentro dos quadros do idealismo e reformulada, com pressupostos materialistas, pelo marxismo. (Seus antecedentes históricos mais remotos tem sido freqüentemente situados na filosofia de Heráclito de Éfeso, para quem o logos era luta e tensão entre opostos nos diferentes planos da realidade). Questão geral que interessa aos lógicos contemporâneos e a do limite das possibilidades de formalização. Por exemplo, procura-se verificar se existe um modelo formalizável, embora complexo, capaz de

expressar as "contradições reais" afirmadas pela dialética marxista. Ou então (p. ex., Chaim Perelman) procura-se distinguir a Lógica do tipo científico (feita com proposições inteiramente formalizáveis) da dialógica ou teoria da argumentação (referente às proposições que, de direito, não são inteiramente formalizáveis).

A tradição histórica da lógica simbólica remonta aos estóicos, mas foi realmente formulada por Leibniz, que pretendeu criar uma espécie de álgebra geral, capaz de ser compreensível por todos independentemente da língua natural de cada um. Pode-se admitir que a lógica simbólica se inicia com o trabalho de George Boole, The Laws of Thought (As leis do pensamento) (1854). Entre as pesquisas que se seguiram, as mais importantes são as dos alemães Gottlob Frege e Georg Cantor, este último, criador da teoria dos conjuntos, e dos ingleses Alfred North Whitehead e Bertrand Russel, autores dos Principia mathematica (1910 a 1913). Das críticas a esta ultima obra surgiram as lógicas plurivalentes de J. Lukasiewicz, as lógicas modais de C. I. Lewis e as teorias de Ludwig Wittgenstein. Uma divergência em torno dos fundamentos filosóficos da obra de Whitehead e Russel fez que fôssem desenvolvidas as escolas formalista e intuicionista, na matemática, a primeira criada por David Hilbert e a segunda por L. E. J. Brouwer.

Em 1927, Hilbert propôs um programa através do qual se buscariam os fundamentos da matemática sobre princípios lógicos tão simples, que teriam, a respeito de sua validade, o acordo dos intuicionistas e dos formalistas. Em 1931, Kurt Gödel provou, num trabalho considerado ponto culminante das pesquisas lógicas desde o Principia mathematica (Uber formal unentscheidbare Satze der Principia mathematica und germandter Systeme (Sobre sentenças formais não-decidíveis dos Principia mathematica e sistemas relacionados), que o programa de Hilbert era irrealizável. As pesquisas seguintes ao resultado de Gödel dirigiram-se para uma formalização da idéia intuitiva de computação através das noções de máquina de Turing e dos teoremas de Alonzo Church (1936) [Church 1936}.

A Lógica, portanto, possui longa história de mais de vinte e três séculos. Aristóteles, na Grécia Antiga, sistematizou e codificou os fundamentos da lógica. "Silogismo é um discurso no qual, tendo-se afirmado algumas coisas, algo além destas coisas se tornam necessariamente verdade". Aristóteles, Primeira Analítica, Livro I, 24ª. Em 1847, George Boole propôs uma linguagem formal que permite a realização de

inferências. Lógica Moderna (1879), Gottlob Frege publicou a 1a. versão do "Cálculo de Predicados". Russel e o Positivismo - Lógica como base para todas as outras ciências. David Hilbert, Guiseppe Peano, Georg Cantor, Ernst Zermelo, Leopold Lowenheim e Thoralf Skolem.

2. 3. UM SISTEMA LÓGICO COMO SISTEMA FORMAL

Pode-se afirmar que *Lógica* é a ciência e a arte do raciocínio correto, que abrange uma área de pesquisa vastíssima e possui inúmeras ramificações. Para entendermos isto melhor, precisamos fazer uma digressão, descrevendo um amplo contexto, dentro do qual localizaremos finalmente o papel e escopo da Lógica.

Há dois conceitos básicos: *Realidade* e *Consciência*.

Nunca houve unanimidade, nem hoje em dia, nem no passado, quanto ao que seria Realidade. Podemos apreender algo do significado do termo "Consciência" a partir de nós próprios, já que somos seres que sentem, pensam e percebem aquilo que parece estar fora de nós. Quanto à Realidade, existem inúmeras perguntas sobre ela formuladas, por filósofos e indagadores de todos os tempos. Ela depende ou não de nossa percepção da mesma? Em que medida ela corresponde ao que parecemos perceber da mesma?

Com certeza, para cada um de nós, algo parece existir que provoca este encadeamento de sentimentos, pensamentos e percepções. O que é real seria mesmo este mundo aparente sugerido pelos nossos sentidos e razões, com espaço, tempo, matéria, sol, planetas, estrelas, etc.? Existem filósofos que respondem afirmativamente, outros assumem uma posição oposta.

Poderíamos, sem entrar no mérito estrito desta discussão, classificar o que é *real* de duas formas distintas:

- no sentido restrito ou *cético*, algo é real com respeito a uma dada consciência se aquilo sempre se manifesta, em todas as circunstâncias; assim sendo, sonhos, por exemplo, não seriam reais, pois os mesmos são contraditados assim que acordamos;

- no sentido amplo ou *crédulo*, algo é real (com respeito a uma dada consciência) se aquilo se manifesta pelo menos uma vez como uma experiência da consciência, seja como uma experiência sensorial genuína, seja como produto da imaginação.

Entre estes dois extremos de Realidade, podem haver algumas gradações intermediárias. Por exemplo, para quase todos nós, este aparente mundo em que vivemos, abrangendo o Planeta Terra e o espaço que o circunda, juntamente com a sociedade humana que nela habita, e todas as vivências daí decorrentes, manifesta-se constantemente em nossa consciência, pelo menos no período em que parecemos estar acordados. Mas há alguma garantia de que assim será para sempre, ou tudo isto é apenas mais um sonho, além de todo aquele material já por nós vivenciado e classificado como "sonho"?

Aquilo que busca perceber a Realidade e que possui ciência de sua própria existência é chamado de *Consciência*. A mesma concebe, em seu íntimo, um reflexo da Realidade, o qual pode refletir, de modo mais ou menos fiel, a própria Realidade.

A consciência humana parece ser provida dos seguintes vetores, os quais concorrem para o seu funcionamento:

- *Razão* – é a faculdade que diz respeito à formação e evolução da estrutura básica de *concepção da Realidade*, erigida pela Consciência. – Distúrbios em seu funcionamento estão ligados a algumas doenças mentais.
- *Emoções* e *Sentimentos* – dizem respeito à coloração que a Consciência empresta à sua concepção da Realidade.
- *Intuição* – é a faculdade da Consciência responsável por uma compreensão global da Realidade, a mais fidedigna possível; é a atmosfera dentro da qual funcionam a Razão, as Emoções e os Sentidos; tem sido vivenciada por muitos homens e mulheres quando os mesmos sentem, subitamente, amplas formas de compreensão, que parecem corresponder a seus mais profundos anseios. – Parece estar, ainda, subdesenvolvida, na maioria dos seres humanos.
- *Instinto* – diz respeito a certas pré-concepções da Realidade assumidas pela Consciência, aparentemente existentes antes da intervenção da Razão, essenciais para a sobrevivência em cada ciclo de existência; à grosso modo, este corresponde, no computador, à memória ROM.

- *Libido* – é a força motriz essencial da sexualidade humana, que objetiva uma vivência de completamento através de um outro *Ser*; foi extensivamente estudada, no Ocidente, a partir do trabalho de Freud.
- *Sentidos* – compreendem os cinco sentidos clássicos, de visão, audição, tato, olfato e paladar, os sentidos de movimento e posição do corpo, bem como inúmeros outros ainda não devidamente catalogados e estudados. – São a porta através da qual a consciência humana adquire informações básicas da Realidade que parece cercá-la; fornecem uma boa parte do material básico a partir do qual a Consciência molda a sua concepção da Realidade.
- *Crenças* – constituem um método de expansão da concepção de Realidade, não baseado nos sentidos e nas formas mais seguras da Razão; as mesmas podem ser *racionais*, quando refletem formas *indutivas* da Razão, ou *não racionais*, quando não as refletem. [Buschbaum, 2001]

A Razão funciona em duas fases que funcionam de uma forma mais ou menos sinérgica:

- Representação da Realidade em uma estrutura concebível pela consciência pensante;
- Raciocínio inferencial do relativamente conhecido para o desconhecido, segundo esta representação.

Estes dois processos são designados respectivamente de *razão formativa* e *razão operativa*.

A Lógica, como ciência, estuda as manifestações da razão operativa em todos os contextos lingüísticos possíveis, nas linguagens escrita e falada. O estudo de tais manifestações tais como ocorrem na vida quotidiana, sem nenhuma idealização, é feito por um ramo desta ciência dito *Lógica Informal*. Um estudo idealizado de tais manifestações, lançando mão de *linguagens formais* e métodos matemáticos, é feito pelo ramo dito *Lógica Formal*. O objeto da *Lógica Formal para Computação* está no estudo das rotinas racionais da Lógica Formal expressáveis em *algoritmos*.

O funcionamento da razão operativa está essencialmente ligado à obtenção de frases condicionalmente verdadeiras a partir de frases hipoteticamente verdadeiras, daí a compreensão e uso de pelo menos uma *linguagem* parece ser imprescindível.
Para nossas finalidades, podemos classificar as linguagens em dois tipos básicos:

- Linguagens naturais ou informais – correspondem às línguas usadas habitualmente pelos seres humanos para a sua comunicação quotidiana, tais como o português, inglês, etc. A sua descrição necessita, em geral, de livros com centenas de páginas, pois as regras de suas gramáticas são complexas e inumeráveis.
- Linguagens artificiais ou formais – as suas regras gramaticais são simples e em pequeno número; algumas, especialmente aquelas em geral utilizadas em Lógica, detém um poder expressivo comparável ao das linguagens naturais.

Para um estudo das regularidades presentes nas diversas formas de raciocínio, bem como dos possíveis algoritmos correspondentes, as linguagens formais são as mais adequadas, pois a sua expressão computacional é exeqüível.

Uma linguagem artificial é definida em duas etapas:

1. escolha de um conjunto não vazio de sinais (em geral, gráficos), a serem usados na construção de suas *expressões significativas*;
2. enunciação de uma (em geral pequena) coleção de regras (isto é, uma *gramática*) destacando, entre as expressões da linguagem, quais são significativas.

As expressões significativas de uma linguagem formal utilizada em Lógica são de um dos dois tipos básicos:

termos – são nomes de objetos do *universo de discurso*[1];

fórmulas – são afirmações ou asserções, na linguagem formal considerada, acerca de objetos do universo de discurso.

Definimos um *sistema lógico* em duas fases:

- fornecemos uma *gramática*, a qual especifica a coleção de linguagens formais deste sistema;
- fornecemos uma *teoria*, a qual especifica, dentre as fórmulas do sistema, quais devem ser consideradas absolutamente verdadeiras, e , para cada contexto de fórmulas supostas hipoteticamente verdadeiras, quais são verdadeiras neste contexto.

Quanto à profundidade de raciocínio envolvida, a Lógica pode atuar em alguns níveis, a saber:

- Lógica proposicional – a mesma estuda relações simples entre fórmulas, as quais correspondem a expressões da língua portuguesa tais como "se... então", "e", "ou", "não é verdade que", etc.
- Lógica quantificacional – além das já citadas relações simples, estuda o comportamento de certas variações de referências feitas no interior de fórmulas; tais variações correspondem, em língua portuguesa a expressões do tipo "para todo", "para algum", "nenhum", etc.
- Lógica equacional – além do que é feito nos dois níveis acima, lida com a equivalência de referências; corresponde a expressões do tipo "é igual a", "é idêntico a", etc.
- Teoria das descrições – lida também com especificações, ambíguas ou não, de objetos do universo de discurso; corresponde a expressões do tipo "um objeto x possuindo a propriedade P(x)", "o objeto x possuindo a propriedade P(x)", etc.
- Teoria dos conjuntos – é a base, o substrato comum em que se baseia toda a matemática tradicional; estuda as propriedades comuns a coleções em geral.

O alfabeto de uma *linguagem lógica* pode utilizar os seguintes tipos de sinais:

- nomes de objetos definidos ou *constantes*, os quais ajudam a constituir os *termos iniciais*;
- nomes de objetos indefinidos ou *variáveis*, os quais, juntamente com as constantes, constituem a coleção dos termos iniciais;
- sinais que formam termos a partir de termos, os *sinais funcionais*; se um sinal funcional usa n termos para formar um termo, dizemos que n é uma *aridade deste sinal*;
- sinais que formam fórmulas a partir de termos, os *sinais predicativos*; se um sinal predicativo usa n termos para formar uma fórmula, dizemos que n é uma *aridade deste sinal*;
- sinais que formam fórmulas a partir de fórmulas, os *conectivos*; se um conectivo usa n fórmulas para formar uma fórmula, dizemos que n é a *aridade deste sinal*; os conectivos lógicos são em geral de aridades 1 ou 2;
- sinais que formam termos a partir de uma variável e uma fórmula, os *qualificadores*;
- sinais que formam fórmulas a partir de uma variável e uma fórmula, os *quantificadores*.

Notação: considere, a partir de agora, as seguintes referências, para as listas de letras descritas abaixo, seguidas ou não de plicas ou subíndices:

- a, b, c – referem-se a constantes;
- x, y, z, w – referem-se a variáveis;
- f, g ,h – referem-se a sinais funcionais;
- p, q, r – referem-se a sinais predicativos;
- t, u – referem-se a termos;
- P, Q, R, S – referem-se a fórmulas;
- Γ, ϑ – referem-se a coleções de fórmulas.

Os conectivos comumente usados em lógica são:

- "$\rightarrow$" – "$P \rightarrow Q$" pode ser lido de uma das seguintes formas:
  - se P, então Q;
  - P implica Q;
  - P acarreta Q;
  - Q se P;
  - P só se Q;
  - P é condição suficiente para Q;
  - Q é condição necessária para P.
- "$\wedge$" – "$P \wedge Q$" é lido como "P e Q".
- "$\vee$" – "$P \vee Q$" é lido como "P ou Q".
- "$\neg$" – "$\neg P$" é lido como "não P" ou "não é verdade que P".
- "$\leftrightarrow$" – "$P \leftrightarrow Q$" é lido como "P equivale a Q" ou "P se, e somente se, Q".

Os quantificadores comumente utilizados são:

- "$\forall$" – dito o *quantificador universal*; "$\forall x$ P" é lido como "qualquer que seja x, P", ou "para todo x, P", ou ainda "para cada x, P";
- "$\exists$" – dito o *quantificador existencial*; "$\exists x$ P" é lido como "existe x tal que P" ou "para algum x, P".

Existem dois sinais predicativos especialmente importantes, respectivamente para a Lógica Equacional e Teoria dos Conjuntos, a saber:

- "$=$" – "$t = u$" é lido como "t é igual a u" ou "t é idêntico a u";
- "$\in$" – "$t \in A$" é lido como "t pertence à coleção A".

Na Lógica Equacional, para cada número natural n, é possível definir ainda três quantificadores existenciais, representando expressões dos tipos "existem no máximo n objetos x tais que P(x)", "existem exatamente n objetos x tais que P(x)" e "existem pelo menos n objetos x tais que P(x)". A fórmula "$\exists x$ P" corresponde à expressão "existe pelo menos um objeto x tal que P(x)". Um caso importante dá-se quando n = 1, sendo que expressões do tipo "existe um único objeto x tal que P(x)" são comumente simbolizadas por "$\exists! x$ P".

Definição: O alfabeto de uma *linguagem quantificacional* possui os seguintes sinais, separados em coleções disjuntas dois a dois:

- uma coleção de *constantes*, eventualmente vazia;
- uma coleção infinita enumerável de *variáveis*;
- para cada $n \geq 1$, uma coleção, eventualmente vazia, de *sinais funcionais de aridade* n, sendo que o mesmo sinal funcional pode ter mais de uma aridade;
- para cada $n \geq 0$, uma coleção, eventualmente vazia, de *sinais predicativos de aridade* n, sendo que o mesmo sinal predicativo pode ter mais de uma aridade; é obrigatório haver pelo menos um sinal predicativo no alfabeto[2];
- os conectivos "$\rightarrow$", "$\wedge$", "$\vee$" e "$\neg$";

  Quanto a "$\neg$" ser conectivo cabe aqui a observação, já que aele não conecta dois elementos da linguagem.
- os quantificadores "$\forall$" e "$\exists$"[3];
- os sinais de pontuação "(", ")" e ",";
- se tal alfabeto fundamenta uma lógica da descrição, o mesmo deve possuir um qualificador;
- se tal alfabeto fundamenta uma lógica equacional, o mesmo deve possuir o sinal predicativo "$=$", de aridade 2;
- se tal alfabeto fundamento uma teoria dos conjuntos, o mesmo deve possuir o sinal predicativo "$\in$", de aridade 2

Definição: *Termos* e *fórmulas* de uma linguagem quantificacional são especificados pelas cláusulas abaixo:

- toda constante é um termo;
- toda variável é um termo;
- se f é um sinal funcional de aridade n e $t_1,\ldots,t_n$ são termos, então $f(t_1,\ldots,t_n)$ é um termo, dito *termo molecular*;
- se x é uma variável, Υ é um qualificador e P é uma fórmula, então $\Upsilon x\ P$ é um termo, dito *descrição*, cuja *matriz* é P;
- se p é um sinal funcional de aridade n e $t_1,\ldots,t_n$ são termos, então $p(t_1,\ldots,t_n)$ é uma fórmula, dita *fórmula atômica*;
- se P é uma fórmula, então $\neg P$ é uma fórmula, dita *negação*;
- se P e Q são fórmulas, então:
  - $(P \rightarrow Q)$ é uma fórmula, dita *implicação*, cujo *antecedente* é P, e cujo *conseqüente* é Q;
  - $(P \wedge Q)$ é uma fórmula, dita *conjunção*, cujos *conjuntores* são P e Q;
  - $(P \vee Q)$ é uma fórmula, dita *disjunção*, cujos *disjuntores* são P e Q.
- se x é uma variável e P é uma fórmula, então:
  - $\forall x\ P$ é uma fórmula, dita *fórmula universal*, cuja *matriz* é P;
  - $\exists x\ P$ é uma fórmula, dita *fórmula existencial*, cuja *matriz* é P.

Teorema da Legibilidade Única: Cada termo e cada fórmula só pode ser lido de uma única forma, isto é:

- as coleções de constantes, variáveis, termos moleculares, descrições, fórmulas atômicas, negações, implicações, conjunções, disjunções, fórmulas universais e fórmulas existenciais são duas a duas disjuntas;
- se $f(t_1,\ldots,t_n)$ e $g(u_1,\ldots,u_p)$ são termos moleculares idênticos, então $f = g$, $n = p$ e, para cada $i \in \{1,\ldots,n\}$, $t_i = u_i$;
- se $\Upsilon x\ P$ e $\Upsilon y\ Q$ são descrições idênticas, então $x = y$ e $P = Q$;
- se $p(t_1,\ldots,t_n)$ e $q(u_1,\ldots,u_p)$ são fórmulas atômicas idênticas, então $p = q$, $n = p$ e, para cada $i \in \{1,\ldots,n\}$, $t_i = u_i$;
- se $\neg P$ e $\neg Q$ são negações idênticas, então $P = Q$;
- se $(P \rightarrow Q)$ e $(R \rightarrow S)$ são implicações idênticas, então $P = R$ e $Q = S$;

- se $(P \wedge Q)$ e $(R \wedge S)$ são conjunções idênticas, então $P = R$ e $Q = S$;
- se $(P \vee Q)$ e $(R \vee S)$ são disjunções idênticas, então $P = R$ e $Q = S$;
- se $\forall x\ P$ e $\forall y\ Q$ são fórmulas universais idênticas, então $x = y$ e $P = Q$;
- se $\exists x\ P$ e $\exists y\ Q$ são fórmulas existenciais idênticas, então $x = y$ e $P = Q$.

Doravante, a menos que seja dito explicitamente o contrário, só consideraremos linguagens quantificacionais sem qualificadores.

Um sistema lógico como sistema formal, consiste em um conjunto de fórmulas e um conjunto de regras de inferência. As fórmulas são sentenças pertencentes a uma linguagem formal cuja sintaxe é dada. A parte de lógica que estuda os valores de verdade é chamada teoria de modelos. Uma regra de inferência é uma regra sintática que quando aplicada repetidamente a uma ou mais fórmulas verdadeiras gera apenas novas fórmulas verdadeiras.

A seqüência de fórmulas geradas através da aplicação de regras de inferência sobre um conjunto de inicial de fórmulas é chamada de prova. A parte de lógica que estuda as provas é chamada teoria de provas.

Gödel e Herbrand na década de 30 mostraram que toda e qualquer fórmula verdadeira pode ser provada.

Church e Turing em 1936 mostraram que não existe um método geral capaz de decidir, em um número finito de passos, se uma fórmula é verdadeira.

Um dos primeiros aplicações da Lógica foi a Prova Automática de Teoremas, a partir da segunda metade da década de 60.

A partir de Kowalsky (1973) lógica passou a ser estudada com método computacional para a solução de problemas[Kowaksky, 1983].

O método explora o fato de expressões lógicas poderem ser colocadas em formas canônicas (apenas com operadores "e", "ou" e "não").

Teoria da Resolução de Robinson - 1965. Transforma a expressão a ser provada para a forma normal conjuntiva ou forma clausal. Existe uma regra de inferência única, chamada regra da resolução. Utiliza um algoritmo de casamento de padrões chamado algoritmo de unificação.

Base para a Linguagem Prolog.

Recentemente Lógicas Não-Padrão ou Não-Clássicas tem sido cada vez mais utilizadas, não somente em IA. Ex: Lógica Temporal tem sido utilizada em estudos de programas concorrentes.

Em IA estas lógicas vem sendo usadas para tratamento de imprecisão, informações incompletas e evolução com o tempo em que evolui o problema tratado por IA.

O Cálculo das Proposições se interessa pelas SENTENÇAS DECLARATIVAS, as PROPOSIÇÕES, que podem ser Verdadeiras ou Falsas.

No âmbito da IA, a lógica permite a representação de conhecimento e o processo de raciocínio para um sistema inteligente.

Como uma linguagem para representação de conhecimento no computador, ela deve ser definida em dois aspectos, A SINTAXE e a SEMÂNTICA.

A SINTAXE de uma linguagem descreve as possíveis configurações que podem constituir sentenças.

A SEMÂNTICA determina os fatos do mundo aos quais as sentenças referem., ou seja, ou sistema "acredita" na sentença correspondente.

Sintaxe das Proposições

<fórmula>::= <fórmula-atômica> | <fórmula-complexa>

<fórmula-atômica>::= Verdadeiro | Falso | P | Q

| R | ...

<fórmula-complexa>::= (<fórmula>)

| <fórmula> <conectivo> <fórmula >

| ¬ <fórmula>

<conectivo>::= ∧ | ∨ | → | ↔

Hoje é segunda ou terça-feira.

Hoje não é terça-feira.

Logo, Hoje é segunda-feira.

S V T, ¬ T ∴ S

A semântica é definida especificando a interpretação dos símbolos da proposição e especificando o significado dos conectivos lógicos.

Uma fórmula tem uma interpretação a qual define a semântica da linguagem. A interpretação pode ser considerada um mapeamento do conjunto das fórmulas para um

conjunto de valores de verdade, que na Lógica dicotômica é o conjunto {verdadeiro,falso} ou {V,F}.

Tabelas Verdade fornecem um teste rigoroso e completo para a validade ou invalidade de formas de argumento do cálculo das proposições. Quando existe um algoritmo que determina se as formas de argumento expressáveis em um sistema formal são válidas ou não, esse sistema é dito DECIDÍVEL. Desta forma, elas garantem a decidibilidade da lógica proposicional.

Uma forma de argumento é válida se todas as suas instâncias são válidas.Uma instância de argumento é válido se sua conclusão for verdade se suas premissas o forem.

Se a forma for válida, então qualquer instância dela será igualmente válida. Assim a Tabela-Verdade serve para estabelecer a validade de argumentos específicos.
Tabelas-Verdade podem ser usadas, não apenas para definir a semântica do conectivos, mas também para testar a validade de sentenças. Exemplo: A Rainha ou a Princesa comparecerá à cerimônia. A Princesa não comparecerá. Logo, a Rainha comparecerá.
$R \vee P, \neg P \Rightarrow R$

Regras de Inferência - Sejam as fórmulas f1, f2, ..., fn ($n \geq 1$) e C. Então, toda seqüência finita de fórmulas, conseqüência de regras de inferência tem como conseqüência final C, chama-se PROVA.

Um ARGUMENTO é uma seqüência de enunciados no qual um deles é a CONCLUSÃO e os demais são as PREMISSAS que servem para provar ou, pelo menos, fornecer algumas evidências para a conclusão.

Evita o trabalho tedioso de ficar construindo Tabelas-Verdade.

$\alpha \vdash \beta$ significa que β pode ser derivado de α através do processo de inferência, onde α e β são fórmulas bem formadas [Mauro, 2001].

Regras de Inferência, Exemplos:

"Se há jogo na Ressacada, então viajar de avião é difícil."

"Se eles chegarem no horário no aeroporto, então a viagem de avião não será difícil."

"Eles, chegaram no horário."

"Logo, não houve jogo na Ressacada."

Equivalência - É um bicondicional que é um teorema.

Árvores de Refutação - São uma outra maneira de garantir a decidibilidade da Lógica Proposicional.

2.4 - REGRAS PARA ÁRVORE DE REFUTAÇÃO

2.4.1. Inicia-se colocando-se as PREMISSAS e a NEGAÇÃO DA CONCLUSÃO.

2.4.2. Aplica-se repetidamente uma das regras a seguir:

2.4.2.1. Negação (¬): Se um ramo aberto contém uma fórmula e sua negação, coloca-se um "X" no final do ramo, de modo a representar um ramo fechado.

(um ramo termina se ele se fecha ou se as fórmulas que ele contém são apenas fórmulas-atômicas ou suas negações, tal que mais nenhuma regra se aplica às suas fórmulas. Desta forma tem-se um ramo fechado, que é indicado por um X, enquanto o ramo aberto não é representado por um X.)

2.4.2.2. Negação Negada (¬ ¬): Se um ramo aberto contém uma fórmula não ticada da forma ¬ ¬ Ø, tica-se ¬ ¬ Ø e escreve-se Ø no final de cada ramo aberto que contém ¬ ¬ Ø ticada.

2.4.2.3. Conjunção (∧): Se um ramo aberto contém uma fórmula não ticada da forma Ø ∧ ß, tica-se, Ø∧ß e escreve-se Ø e ß no final de cada ramo aberto que contém Ø ∧ ß ticada.

2.4.2.4. Conjunção Negada (¬ ∧): Se um ramo aberto contém uma fórmula não ticada da forma ¬ (Ø∧ß), tica-se, ¬ (Ø∧ß) e BIFURCA-SE o o final de cada ramo aberto que contém ¬ (Ø ∧ ß) ticada, no final do primeiro ramo se esreve ¬ Ø e no final do segundo ramo se escreve ¬ ß.

2.4.2.5. Disjunção (v): Se um ramo aberto contém uma fórmula não ticada da forma Øvß, tica-se, Øvß e BIFURCA-SE o o final de cada ramo aberto que contém Ø v ß ticada, no final do primeiro ramo se esreve Ø e no final do segundo ramo se escreve ß.

2.4.2.6. Condicional (→): Se um ramo aberto contém uma fórmula não ticada da forma Ø → ß, tica-se, Ø → ß e BIFURCA-SE o final de cada ramo aberto que contém Ø→ ß ticada, no final do primeiro ramo se escreve ¬ Ø e no final do segundo ramo se escreve ß.

2.4.2.7. Disjunção Negada ($\neg$ v): Se um ramo aberto contém uma fórmula não ticada da forma $\neg$ (ØvB), tica-se, $\neg$ (ØvB) e ESCREVE-SE $\neg$ Ø e $\neg$ B no final de cada ramo aberto que contém $\neg$ (ØvB) ticada.

2.4.2.8. Condicional Negado ($\neg \rightarrow$): Se um ramo aberto contém uma fórmula não ticada da forma $\neg$(Ø$\rightarrow$B), tica-se, $\neg$ (Ø $\rightarrow$ B) e ESCREVE-SE Ø e $\neg$ B no final de cada ramo aberto que contém $\neg$ (Ø$\rightarrow$B) ticada.

2.4.2.9. Bicondicional ($\leftrightarrow$): Se um ramo aberto contém uma fórmula não ticada da forma Ø $\leftrightarrow$ B, tica-se, Ø $\leftrightarrow$ B e BIFURCA-SE o o final de cada ramo aberto que contém Ø $\leftrightarrow$ B ticada, no final do primeiro ramo se escreve Ø e B e no final do segundo ramo se escreve $\neg$ Ø e $\neg$ B.

2.4.2.10. Bicondicional Negado ($\neg \leftrightarrow$): Se um ramo aberto contém uma fórmula não ticada da forma $\neg$ (Ø $\leftrightarrow$ B), tica-se, $\neg$ (Ø $\leftrightarrow$ B) e BIFURCA-SE o o final de cada ramo aberto que contém $\neg$ (Ø $\leftrightarrow$ B) ticada, no final do primeiro ramo se escreve Ø e $\neg$B e no final do segundo ramo se escreve $\neg$ Ø e B.[Mauro 2001].

2.5 - LINGUAGEM DE PRIMEIRA ORDEM NA LÓGICA CLÁSSICA

2.5.1 Definição: Por um alfabeto A entende-se uma coleção não vazia de sinais gráficos. Uma samblagem de um dado alfabeto e uma seqüência finita de símbolos do alfabeto. A denota o conjunto de todas as samblagens de A.

2.5.2 Definição: Um alfabeto de primeira ordem é a união das seguintes coleções de sinais, disjuntas duas a duas:

uma coleção infinita enumerável de sinais ditos variáveis, as quais são as mesmas em todo alfabeto de primeira ordem;

a coleção {$\neg$, $\wedge$, $\vee$, $\rightarrow$, } de sinais ditos conetivos lógicos;

a coleção {$\forall$, $\exists$} de sinais ditos quantificadores;

a coleção {")", "(" ","} de sinais ditos de pontuação;

uma coleção (eventualmente vazia) de sinais ditos constantes;

um alfabeto de primeira ordem com igualdade contém também a coleção {=};

uma coleção (possivelmente vazia) de sinais ditos funcionais; a cada sinal funcional associamos uma coleção de números positivos, ditos as suas aridades;

uma coleção (não vazia) de sinais ditos predicativos; a cada sinal predicativo associamos uma coleção de números naturais, ditos as suas aridades.

2.5.3 Convenção: Consideramos que a função sintática de cada soma utilizada em dois alfabetos distintos é invariante. Em particular, temos que:

todo sinal considerado constante em um dado alfabeto deve continuar a sê-lo em qualquer outro alfabeto considerado no mesmo contexto;

todo sinal considerado funcional (predicativo) em um dado alfabeto deve continuar a sê-lo em qualquer outro alfabeto considerado no mesmo contexto, de modo que a sua coleção de aridades é invariante para todos os alfabetos considerados.

2.5.4 Convenção: De agora em diante, a não ser que seja dito expressamente o contrário, adotaremos as seguintes convenções notacionais:

as letras x, y, z, w, seguidas ou não de plicas ou índices denotam variáveis de um alfabeto de primeira ordem;

as letras a, b, c, seguidas ou não de plicas ou índices, denotam constantes de um alfabeto de primeira ordem;

a letra Σ, seguida ou não de plicas ou índices, denota um alfabeto de primeira ordem;

as letras p, q, r, seguidas ou não de plicas ou ínidces, denotam sinais predicativos de um alfabeto de primeira ordem;

as letras f, g, h, seguidas ou não de plicas ou índices , denotam sinais funcionais de um alfabeto de primeira ordem.

2.5.5 Definição: As seguintes cláusulas especificam o que entendemos por Σ-termos:

toda variável é um E-termo~Σ-termo;

toda constante de Σ é um Σ-termo;

se $t_1, ..., t_n$ são Σ-termos e se f é um sinal de funcional de Σ, então $f(t_1,...,t_n)$ é um Σ-termo.

2.5.6 Definição: As seguintes cláusulas especificam o que entendemos por Σ-fórmulas:

se t1, ... , tn são Σ-termos e r é um sinal de predicativo de Σ, então r(t1,...,tn) é uma Σ-fórmulas;

se P é uma Σ-fórmula, então $\neg$ P é uma Σ-fórmulas;

se P e Q são Σ-fórmulas, então $(P \wedge Q)$, $(P \vee Q)$ e $(P \rightarrow Q)$ são Σ-fórmulas;

se P e uma Σ-fórmulas e x é uma variável, então $\exists$ x P são Σ-fórmulas.

2.5.7. Definição: Uma linguagem de primeira ordem é a coleção de todas as Σ-formulas de um dado alfabeto Σ; dizemos também neste caso que tal linguagem é a linguagem de primeira ordem gerada por Σ. Se t é um Σ-termo e L e a linguagem gerada por Σ, dizemos que t é um termo em L.

5.8 Convenção: Sempre que for irrelevante a indicação do alfabeto Σ especifico, usaremos respectivamente as palavras "termos" e "fórmulas" no lugar de "Σ-termos" e "Σ-fórmulas.

2.5.9 Convenção: Adotamos as seguintes conveções sintáticas para todos os sinais abaixo, seguidos ou não de plicas ou índices:

- t, u denotam termos;
- P, Q, R, S denotam fórmulas;
- Γ, ϑ denotam coleções de fórmulas;
- L denota uma linguagem de primeira ordem.

2.5.10 Definição: Um designador é um termo ou uma fórmula.

2.5.11 Definição: Pelas cláusulas abaixo especificamos quando um termo é subtermo de outro termo:

- t é subtermo de t;
- t_i é subtermo de $f(t_1, ..., t_n)$, onde $1 < i < n$;

- se t_1 é subtermo de t_2 e t_2 é subtermo de t_3, então t_1 é subtermo de t_3.

2.5.12 Definição: "$P \longleftrightarrow Q$" é uma abreviatura para "$(P \rightarrow Q) \wedge (Q \rightarrow P)$".

2.5.13 Definição: Pelas cláusulas abaixo especificamos quando uma dada fórmula é subfórmula de outra dada fórmula:

- P é subfórmula de P;
- P é subfórmula de $\neg P$;
- P e Q são subfórmulas de $(P \rightarrow Q)$, $(P \vee Q)$ e $(P \wedge Q)$
- P é subfórmula de $\forall x\, P$ e $\exists x\, P$;
- se P é subfórmula de Q e Q e subfórmula de R, então P é subfórmula de R.

2.5.14 Convenção: Adotaremos as seguintes convenções para escrita informal de termos e fórmulas:

- notamos $f(t_1, t_2)$ por $(t_1\ f\ t_2)$;
- notamos $p(t_1, t_2)$ por $(t_1\ p\ t_2)$;
- quando $(t_1\ f\ t_2)$ não está escrito como subtermo de outro termo podemos prescindir do par exterior de parênteses;
- quando $(t_1\ p\ t_2)$ não está escrito como subfórmula de outra fórmula podemos prescindir do par exterior de parênteses;
- mesmo quando $(t_1\ p\ t_2)$ está escrito como subfórmula de outra fórmula, podemos prescindir do par exterior de parênteses, se isto não prejudicar a clareza da escrita da fórmula envolvida;
- quando $(P \rightarrow Q)$, $(P \wedge Q)$ e $(P \vee Q)$ não estão escritos como subfórmulas de outra fórmula podemos prescindir do par exterior de parênteses;
- a seguinte lista fornece a ordem de prioridade para separação em subfórmulas: $\Leftrightarrow, \rightarrow, \{\wedge \vee\}$; por exemplo, "$P \Leftrightarrow Q \vee R \rightarrow S$" representa "$P \Leftrightarrow ((Q \vee R) \rightarrow S)$";
- quando o mesmo conetivo de aridade 2 se suceder em uma fórmula, a parentização implícita se dá da direita para esquerda; por exemplo, "$P \rightarrow Q \rightarrow P$" representa "$P \rightarrow (Q \rightarrow P)$".

2.5.15 Definição: Especificamos abaixo quando uma determinada variável é livre em uma fórmula:

- x é livre em $r(t_1,.. , t_n)$ se x ocorre em um dos termos $t_1,...t_n$;
- x é livre em $\neg$ P se x é livre em P;
- x é livre em $(P \rightarrow Q)$,$(P \wedge Q)$ e $(P \vee Q)$ se x é livre em P ou x é livre em Q;
- x não é livre em $\forall xP$ e $\exists xP$;
- x é livre em $\forall yP$ e $\exists yP$ se x é distinto de y e x é livre em P.

2.5.16 Definição: Se D é um designador, dizemos que x é livre em D se uma das seguintes condições for satisfeita:

- D é um termo e x ocorre em D;
- D é uma fórmula e x ocorre livre em D.

Se S é uma coleção de designadores, dizemos que x é livre em S se x é livre em pelo menos um dos elementos de S.

5.17 Definição: Termos fechados são termos nos quais não ocorrem variáveis.

2.5.18 Definição: Sentenças são fórmulas sem variáveis livres

2.5.19 Definição: Especificamos abaixo o resultado da substituição de uma variável por um dado termo em um termo ou uma fórmula (consideramos aqui x distinto de y):

- c(x/t) = c;
- x(x/t) = t;
- y(x/t) = y;
- f(t1, ..., tn)(x/t) = f(t1(x/t), ..., tn(x/t));
- p(t1, ..., tn)(x/t) = p(t1(x/t), ..., tn(x/t));
- $(\neg P)(x/t) = \neg P(x/t)$;
- $(P \rightarrow Q)(x/t) = P(x/t) \rightarrow Q(x/t)$;
- $(P \wedge Q)(x/t) = P(x/t) \wedge Q(x/t)$;
- $(P \vee Q)(x/t) = P(x/t) \vee Q(x/t)$;
- $(\forall x\ P)(x/tl = \forall x\ P$;
- $(\exists x\ P)(x/tl = \exists x\ P$;

- $(\forall y\ P)(x/t) =$

  - $\forall y\ P$, se x não é livre em P;

  - $\forall y\ P(x/t)$, se x é livre em P e y não ocorre em t;

  - $\forall z\ P(y/z)(x/t)$, se x é livre em P e y ocorre em t, onde z é a primeira variável, distinta de x e y, que não é livre em P e não ocorre em t;

- $(\exists y\ P)(x/t) =$

  - $\exists y\ P$, se x não é livre em P;

  - $\exists y\ P)(x/t)$, se x é livre em P e y não ocorre em t;

  - $\exists z\ P)(y/z)(x/t)$, se x é livre em P e y ocorre em t, onde z é a primeira variável, distinta de x e y, que não é livre em P e não ocorre em t.

2.5.20 Definição: Dizemos que um termo t (uma formula P) está no escopo de uma variável x em uma formula Q se existe uma subfórmula de Q de uma das formas $x\ \forall x\ R$ ou $\exists x\ R$, tal que t(P) ocorre em R.

2.5.21 Definição: Especificamos abaixo quando duas fórmulas são congruentes:

· P é congruente a P;

· se P é congruente a P', então $\neg P$ é congruente a $\neg P'$;

. se P é congruente a P' e Q é congruente a Q', então $P \rightarrow Q$ ($P \wedge Q$, $P \vee Q$) é congruente a $P' \rightarrow Q'$ ($P' \wedge Q'$, $P' \vee Q'$);

· se P é congruente a P', então $\forall x\ P$ ($\exists x\ P$) é congruente a $\forall x\ P'$ ($\exists x\ P'$);

· se y não é livre em P e P é congruente a P'(x/y), então $\forall x\ P$ ($\exists x\ p$) é congruente a $\forall y$ P'(x/y) ($\exists y$ P'(x/y)).

2.6 - SEMÂNTICA CLÁSSICA

Atribuímos significados à fórmulas de uma dada linguagem escolhendo uma coleção de valores veritativos, os quais são distintos juízos que podem ser feitos acerca do grau de verdade e falsidade de uma dada fórmula. Os valores veritativos que

atribuem graus de veracidade a uma dada fórmula são também chamados de valores distinguidos.

Em qualquer lógica o conceito semântico fundamental e o de valoração, a qual é uma função que associa fórmulas a valores veritativos, possuindo determinadas propriedades, variando de lógica para lógica. Definimos assim uma semântica para uma dada lógica, especificando para cada linguagem L desta lógica qual é o conjunto associado de valorações para L.

2.6.1 Predefinição: Dada uma lógica consideramos conhecida a coleção correspondente de valores veritativos e, para cada linguagem desta lógica, a coleção correspondente de valorações.

2.6.2 Definição: Dizemos que V e uma valoração em uma dada lógica se existe uma linguagem L, desta lógica, tal que V e uma valoração para L.

Considere nos parágrafos seguintes uma lógica fixa, e seja L uma linguagem para esta lógica.

2.6.3 Definição: Dizemos que uma valoração V para L satisfaz uma fórmula P de L se V(P) é um valor distinguido. V satisfaz uma coleção de formulas de L se esta satisfizer todas as fórmulas desta coleção.

Definição: Uma valoração V satisfaz P (Γ) se existir uma linguagem L tal que V é uma valoração para L. P (Γ) é uma fórmula de L (uma coleção de fórmulas de L) e V satisfaz P (Γ).

2.6.5 Definição: Se $P \in L$ ($\Gamma \subseteq L$) e V é uma valoração para L, dizemos que V é uma valoração para P (Γ).

2.6.6 Definição: Considere $\Gamma \cup \{P\} \subseteq L$. Nas cláusulas abaixo especificamos conceitos semânticos básicos para fórmulas e para coleções de fórmulas em uma dada lógica fixa:

- P (Γ) é satisfatível em L se existe uma valoração para L que satisfaz P (Γ);
- P (Γ) é satisfatível se existe uma valoração para P (Γ) que satisfaz P (Γ);
- P (Γ) é insatisfatível em L se toda valoração para L não satisfaz P (Γ);
- P (Γ) é insatisfatível se toda valoração para P (Γ) não satisfaz P (Γ);
- P (Γ) é válido em L se toda valoração para L satisfaz P (Γ);

- P (Γ) é válido se toda valoração para P (Γ) satisfaz P (Γ);
- P (Γ) é inválido em L se existe uma valoração para L que não satisfaz P (Γ);
- P (Γ) é inválido se existe valoração para P (Γ) não satisfaz P (Γ);
- P é conseqüência semântica de Γ em L, e notamos isto por $\Gamma \models$ (P), se toda valoração para L que satisfaz Γ satisfaz P;
- P é conseqüência .semântica de Γ se toda valoração para $\Gamma \cup \{P\}$ que satisfaz Γ satisfaz P.

2.6.7 Definição: Em qualquer um dos níveis proposicional, quantificacional ou equacional da lógica clássica, a coleção de valores veritativos correspondente e a mais simples possível, contendo apenas dois valores veritativos: v, representando "verdadeiro", e f, representando "falso". Dotamos o conjunto {v, f} de uma relação de ordem, considerando f estritamente menor que v.

2.6.8 Escólio: O leitor pode se convencer facilmente que este conjunto, munido da relação da ordem que acabamos de definir, é uma álgebra booleana, na qual qualquer subconjunto da mesma possui um elemento mínimo e possui um elemento máximo. Temos também, nesta álgebra que o complemento de v, notado por v', e f, e o complemento de f, notado por Γ, é v.

2.6.9 Definição: Uma valoração proposicional para L é uma função V atendendo às seguintes condições:

- V e uma função de L em {v, f};
- $V(\neg P) = V(P)'$;
- $V(P \rightarrow Q) = \max\{V(P)', V(Q)\}$;
- $V(P \wedge Q) = \min\{V(P), V(Q)\}$;
- $V(P \vee Q) = \max\{V(P), V(Q)\}$.

2.6.10 Teorema: Na lógica proposicional clássica os conceitos de satisfatibilidade, insatisfatibilidade, validade, invalidade e conseqüência semântica são invariantes com respeito às mudanças de linguagem.

Para definirmos uma semântica da lógica clássica a nivel quantificacional devemos atribuir significados a constantes, variáveis, sinais funcionais e sinais predicativos (através do conceito de interpretação), obtendo duas funções semânticas básicas: a primeira, dita denotação, associa termos a indivíduos do universo de discurso da interpretação, e a segunda, dita valoração quantificacional, associa fórmulas a valores veritativos.

2.6.11 Definição: Uma L-interpretação (para a lógica quantificacional clássica) é um termo $I = < \Delta, \sigma, s >$ atendendo às seguintes condições:

- Δ é um conjunto não vazio, dito universo de discurso;
- σ é uma função, dita a L-atribuição de sinais de I, satisfazendo as seguintes cláusulas:
  - para cada constante c em L, $\sigma(c) \in \Delta$;
  - para cada sinal funcional f em L e cada aridade n de f, $\sigma(f,n)$ é uma função de Δ" em Δ;
  - para cada sinal predicativo p em L e cada aridade n de p, $\sigma(p,n)$ está contido em Δ";
  - s é uma função dita a Δ-atribuição de variáveis de I, da coleção de variáveis para Δ;

Se I e uma L-interpretação para a lógica equacional clássica, então $\sigma(=, 2)$ é a coleção $\{(x, x)/ x \in \Delta\}$. Uma interpretação é uma L-interpretação para alguma linguagem de primeira ordem L; dizemos neste caso que L é a linguagem da interpretação. Se $P \in L$ ($\Gamma \subseteq L$) e L é a linguagem de uma interpretação I, dizemos que I é uma interpretação para P (Γ).

2.6.12 Definição: Dada uma Δ-atribuição s de variáveis e um elemento d de Δ, definimos uma nova Δ-atribuição de variáveis, notada por s(x/d), por $s(x/d) =$

$$\begin{cases} d, \text{ se } x = y, \\ s(y), \text{ se } x \neq y \end{cases}$$

Se $I = <\Delta, \sigma, s >$ é uma L-interpretação, então I(x/d) é a L-interpretação especificada pelo terno $I = <\Delta, \sigma, s(x/d)>$.

2.6.13 Definição: Dada uma L-interpretação $I = < \Delta, \sigma, s >$, definimos duas funções I_t e I_f, ditas respectivamente a L-denotação e a L-valoração quantificacional geradas por I pelas seguintes cláusulas:

- I_t é uma função da coleção de termos em L para Δ;
- $I_t(c) = \sigma(c)$;
- $I_t(x) = s(x)$;
- $I_t(f(t_1, \ldots, t_n)) = \sigma(f,n)(I_t(t_1), \ldots, I_t(t_n))$;
- I_f é uma L-valoração proposicional de L em {v, f};
- $I_f(p(t_1, \ldots, t_n)) = v$ sss $< I_t(t_1),\ldots, I_t(t_n)> \in \sigma (p,n)$;
- $I_f(\forall xP) = Min\{ I(x/d)_f(P)/d \in \Delta\}$;
- $I_f(\exists xP) = Max\{ I(x/d)_f(P)/d \in \Delta\}$.

2.6.14 Escólio: Se I é uma L-interpretação equacional para a lógica clássica, então I_f é dita a L-valoração equacional gerada por I, e obedece à seguinte cláusula adicional:

2.6.15 Definição: Dizemos que uma L-interpretação satisfaz uma fórmula P de L se $I_f(P) = v$. Uma L-interpretação satisfaz uma coleção Γ de fórmulas de L se esta satisfizer todas as fórmulas desta coleção. Uma interpretação I satisfaz P (Γ) se esta é uma L-interpretação para P (Γ) que satisfaz P (Γ).

2.6.16 Notação: Usamos as seguintes siglas:

- LCP = lógica clássica proposicional;
- LCQ = lógica clássica quantificacional
- LCE = lógica clássica equacional.

2.6.17 Teorema: Se $\Gamma \cup \{P\}$ é uma coleção de fórmulas não contendo quantificadores, então:

- $\Gamma \models_{LCP} P$ sss $\Gamma \models_{LCQ} P$;

- $\Gamma \models P$ sss $\Gamma \models P$;
  LCQ LCE

2.6.18 Teorema: Se $\Gamma \cup \{P\}$ é uma coleção de fórmulas não contendo o sinal de "=", então:

- $\Gamma \models P$ sss $\Gamma \models P$;
  LCQ LCE

Os dois teoremas anteriores dizem que a lógica clássica quantificacional equacional são extensões conservativas da lógica clássica proposicional, e que a lógica clássica equacional e uma extensão conservativa da lógica clássica quantificacional. Devido a este fato estabelecemos a convenção seguinte:

2.6.19 Convenção: De agora em diante, quando expusermos fatos semânticos, ficara implícito que estamos nos referindo a lógica clássica, sem nos preocuparmos se estamos em um dos níveis proposicional, quantificacional ou equacional, a não ser que seja necessário.

2.6.20 Definição: Dizemos que duas interpretações $I = <\Delta, \sigma, s >$ e $I' = <\Delta, \sigma', s' >$ concordam em uma fórmula P (Γ) se as seguintes condições forem satisfeitas:

· I e I' são interpretações para P (r);
· para toda constante c ocorrendo em P (r), a(c) = 6'(c);
· para todo sinal funcional f ocorrendo em P (r) e p: toda aridade de f, a(f,n) = 6'(f,n);
· para todo sinal pr p o P (T') e p~ ide ~ ie p,
a(P,n) = a°(P,n);
> para toda variavel .

2.6.21 Lema da Concordância de Interpretações: Se duas interpretações I e I' concordam em P (Γ), então I satisfaz P (Γ) se, e somente se, I' satisfaz P (Γ).

Uma conseqüência imediata do Teorema 2.10 e do Lema da Concordância é o teorema a seguir, o qual cita uma série de propriedades semânticas invariantes com respeito às mudanças de linguagem nos níveis proposicional, quantificacional e equacional da lógica clássica.

2.6.22 Teorema: Os conceitos de satisfatibilidade, insatisfatibilidade, validade, invalidade e de conseqüência semântica são invariantes com respeito as mudanças de linguagem, em qualquer um dos níveis proposicional, quantificacional e equacional da lógica clássica. Para exemplificar o que entendemos por isto, detalhamos abaixo essa idéia para o conceito de satisfatibilidade. As seguintes proposições são equivalentes: .

· P (Γ) é satisfatível em alguma linguagem L;
· P (Γ) é satisfatível em toda linguagem L, tal que $P \in L$ ($\Gamma \subseteq L$);
· P (Γ) é satisfatível.

2.7 – REGRAS E TEOREMAS DA LÓGICA CLÁSSICA

Nesta seção definiremos uma relação alternativa, a relação de conseqüência sintática para a lógica clássica, baseada em processos mecânicos de manipulação simbólica, a qual se revela equivalente à relação de conseqüência semântica da mesma lógica. Além disso, enumeramos as regras e teoremas da lógica clássica que consideramos de maior importância.

Existem algumas alternativas que poderiam servir para definirmos esta relação de conseqüência: cálculos axiomáticos ou sistemas de Hilbert, por exemplo em [Enderton 1972], sistemas de dedução natural, por exemplo em [Prawitz 1965] e cálculos de seqüentes, por exemplo em [Ebbinghaus & outros 1989]. Escolhemos aqui a terceira opção, devido a sua simplicidade.

2.7.1 Definição: Um seqüente é uma lista de fórmulas (de alguma linguagem de primeira ordem). Um esquema de seqüentes é uma coleção de seqüentes. Uma regra de seqüentes e uma coleção de n-tuplas de seqüentes, onde n e um número natural maior ou

igual a dois. Um postulado de seqüentes é um esquema de seqüentes ou uma regra de seqüentes. Um cálculo de seqüentes é uma coleção de postulados de seqüentes possuindo pelo menos um esquema de seqüentes.

2.7.2 Definição: Seja C um cálculo de seqüentes. Um axioma seqüencial em C e um elemento de um esquema de C. Uma aplicação de regra em C e um elemento de uma regra de seqüentes de C. Se <Si, -.., S", S> e uma aplica~ao de uma regra de seqüentes de C, notamos usualmente <SI, ..., S", S> por Sl, "', ~ ; dizemos neste caso que S Si, ..., S" são as hipóteses da aplica~ao e que S e a conclusão da aplicação. Uma demonstração em C e uma lista de seqüentes, tal que cada elemento desta lista e um axioma seqüencial em C ou e uma conclusão de uma aplicação de uma regra de seqüentes de C, tal que todas as hipóteses dessa aplicação precedem esta conclusão na lista.

2.7.3 Definição: Dizemos que P é conseqüência sintática de Γ em C e notaremos isto por $\Gamma \vdash_{C} P$. se (Γ P) o último elemento de uma demonstração em C.

2.7.4 DefInição: A seguir damos os postulados que formam o cálculo de seqüentes para lógica clássica proposicional, quantificacional e equacional.

Os axiomas e regras de seqüentes primitivos para o cálculo de seqüentes clássico proposicional são os seguintes:

Esquema da Reflexividade: Se $P \in \Gamma$, então "Γ P"é um axioma seqüencial da reflexividade;

Regra da Transitividade: $\dfrac{\text{" } \Gamma P1,\ldots, \Gamma Pn, \{P1,\ldots,Pn\ P\}\ Q \text{ "}}{\Gamma\ Q}$ é uma aplicação da regra da transitividade;

· Regra da Monotonicidade: se $\Gamma \subseteq \Gamma'$, então $\dfrac{\text{" } \Gamma\ P \text{ "}}{\Gamma'\ P}$ é uma aplicação da regra da monotonicidade;

Regra da Dedução: $\dfrac{\text{“ } \Gamma\ P\ \ Q\text{”}}{\Gamma\ P \rightarrow Q}$ é uma aplicação da regra da dedução;

Esquema Modus Ponens: "P P $\rightarrow$ Q Q" é um axioma seqüencial do esquema modus ponens;

· Esquema do $\wedge$-introdução: "P Q P $\wedge$ Q" é um axioma seqüencial do esquema do $\wedge$-introdução;

quemas do $\wedge$-eliminação:

· "P $\wedge$ Q P" é um axioma seqüencial do primeiro esquema do $\wedge$-eliminação;

· "P $\wedge$ Q Q" é um axioma seqüencial do segundo esquema do $\wedge$-eliminação;

· Esquemas do $\vee$-introdução:

· "P P $\vee$ Q" é um axioma seqüencial do primeiro esquema do $\vee$-introdução,

"Q P $\vee$ Q" é um axioma seqüencial do segundo esquema do $\vee$-introdução;

· Regra da Prova por Casos: "Γ P $\vee$ Q, Γ P R, Γ Q R" é uma aplicação da

Γ R

regra da prova por casos;

Regras da Redução ao Absurdo:

"Γ P Q, Γ P $\neg$Q " é uma aplicação da primeira regra da redução ao absurdo;

Γ $\neg$P

"Γ $\neg$P Q, Γ $\neg$ P $\neg$Q" é uma aplicação da segunda regra da redução ao absurdo;

Γ P

Os axiomas e regras de seqüentes primitivos para o cálculo de seqüentes clássico quantificacional são todos os que foram dados acima para o cálculo de seqüentes proposicional, mais os seguintes:

Regra da Generalização: Se x não é livre em Γ, então "Γ P" é uma aplicação da

Γ $\forall$xP

regra da generalização;

Esquema do $\forall$-eliminação: $\forall$x P P(x/t)" é um axioma seqüencial do esquema do $\forall$-eliminação;

Esquema do ∃-introdução: "P (x/t) ∃x P" é um axioma seqüencial do esquema do ∃-introdução;

- Regra do ∃-eliminação: Se y não é livre em $\Gamma \cup \{Q\}$, então :

" $\Gamma \exists xP.\ \Gamma P(x/y)\ Q$ " é uma aplicação da regra do ∃-eliminação.

ΓQ

Os axiomas e regras de seqüentes primitivos para o cálculo de seqüentes clásssicos são todos os que foram dados acima, mais os seguintes:

· Esquema da Reflexividade da Igualdade: "t = t" é um axioma seqüencial do esquema da reflexividade da igualdade.

Esquema da Substituição em Sinais Funcionais:

"$t_1 = t_1'$ ... $t_n = t_n'$ $F(t_1, ..., t_n) = f(t_1', ..., t_n')$ é uma axioma seqüencial do esquema da substituição em sinais funcionais.

Esquema da Substituição em Sinais Predicativos:

"$t_1 = t_1'$ ... $t_n = t_n'$ $p(t_1, ..., t_n) \rightarrow p(t_1', ..., t_n')$ na axioma seqüencial do esquema da substituição em sinais predicativos.

2.7.5 Notação: Usamos as seguintes siglas:

· CCP = cálculo de seqüentes clássico proposicional;

· CCQ = cálculo de seqüentes clássico quantificacional;

· CCE = cálculo de seqüentes clássico equacional;

2.7.6 Teorema: Se $\Gamma \cup \{P\}$ é uma coleção de fórmulas não contendo quantificadores, então:

- $\Gamma \vdash_{CCP} P$ sss $\Gamma \vdash_{CCQ} P$;

- $\Gamma \vdash_{CCP} P$ sss $\Gamma \vdash_{CCE} P$;

2.7.7 Teorema: Se $\Gamma \cup \{P\}$ é uma coleção de fórmulas não contendo o sinal de "=", então:

- $\Gamma \vdash P$ sss $\Gamma \vdash P$.
  CCQ CCE

Os dois teoremas anteriores dizem que os cálculos clássico quantificacional e equacional são extensões conservativas do cálculo clássico proposicional, e que o cálculo clássico equacional é uma extensão conservativa do cálculo clássico quantificacional.

Devido a este fato e pelos Teoremas 6. 17 e 6. 18, exporemos, de agora em diante, fatos adicionais sobre a lógica clássica sem nos preocuparmos em geral se estamos em um dos níveis proposicional, quantificacional ou equacional.

2.7.8 Convenção: Quando estamos trabalhando exclusivamente com a lógica clássica, para dizer que P é conseqüência sintática de Γ no cálculo de seqüentes para a lógica clássica. notaremos isto por $\Gamma \vdash P$.

A seguir damos uma lista das principais regras e esquema do cálculo de Seqüentes para a lógica clássica.

2.7.9 Esquemas e Regras Estruturais:

Regras Estruturais:

- Reflexividade: Se $P \in \Gamma$, então $\Gamma \vdash P$;
- Transitividade: Se $\begin{cases} \Gamma & P_1 \\ & \\ \Gamma & P_n \end{cases}$ e $P_1,\ldots,P_n \vdash Q$, então $\Gamma \vdash Q$;
- Monotonicidade: Se $\Gamma \vdash P$ e $\Gamma \subseteq \Gamma'$, então $\Gamma' \vdash P$;
- Compacidade: Se $\Gamma \vdash P$, então existe Γ' finito tal que $\Gamma' \vdash P$.

2.7.10 Esquemas e Regras de Introdução e Eliminação de Conectivos:

Regras de Introdução e Eliminação para Conectivos:

- Regra da Dedução: Se $\Gamma, P \vdash Q$, então $\Gamma \vdash P \rightarrow Q$;

- Modus Ponens: $P, P \rightarrow Q \vdash Q$;
- $\wedge$-Introdução: $P, Q \vdash P \wedge Q$;
- $\wedge$-Eliminação: $\begin{cases} (i)\ P \wedge Q \vdash P; \\ (ii)\ P \wedge Q \vdash Q; \end{cases}$
- $\vee$-Introdução: $\begin{cases} (i)\ P \vdash P \vee Q; \\ (ii)\ Q \vdash P \vee Q; \end{cases}$
- Regra da Prova por Casos: $P \vee Q, P \rightarrow R, Q \rightarrow R \vdash R$;
- Regras da Redução ao Absurdo:

(i) Se $\begin{cases} \Gamma, P \vdash Q \\ \Gamma, P \vdash \neg Q \end{cases}$, então $\Gamma \vdash \neg P$;

(ii) Se $\begin{cases} \Gamma, \neg P \vdash Q \\ \Gamma, \neg P \vdash \neg Q \end{cases}$, então $\Gamma \vdash P$.

2.7.11 Esquemas e Regras complementares para Conectivos:

Regras Básicas para Conectivos:

- Reflexividade da Implicação: $\vdash P \rightarrow P$;
- Não Contradição: $\begin{cases} (i)\ P, \neg P \vdash Q; \\ (ii) \vdash \neg(P \wedge \neg P); \end{cases}$
- Terceiro Excluído: $\vdash P \vee \neg P$;
- Conseqüente da Implicação: $Q \vdash P \rightarrow Q$;
- Antecedente da Implicação: $\neg P \vdash P \rightarrow Q$;
- Silogismo Hipotético: $P \rightarrow Q, Q \rightarrow R \vdash P \rightarrow R$;
- **Dupla Negação**: $\begin{cases} (i)\ \neg\neg P \vdash P; \\ (ii)\ P \vdash \neg\neg P; \end{cases}$
- Modus Tollens: $\neg Q, P \rightarrow Q \vdash \neg P$;
- Contraposição: $\begin{cases} (i)\ P \rightarrow Q \vdash \neg Q \rightarrow \neg P; \\ (ii)\ \neg Q \rightarrow \neg P \vdash P \rightarrow Q; \end{cases}$
- Silogismo Disjuntivo: $\begin{cases} (i)\ \neg P, P \vee Q \vdash Q; \\ (ii)\ \neg Q, P \vee Q \vdash P; \end{cases}$

Negação da Implicação:
$$\begin{cases} \text{(i) } \neg(P \to Q) \vdash P \wedge \neg Q; \\ \text{(ii) } P \wedge \neg Q \vdash \neg(P \to Q); \end{cases}$$

- Negação da Conjunção:
$$\begin{cases} \text{(i) } \neg(P \wedge Q) \vdash P \to \neg Q; \\ \text{(ii) } P \to \neg Q \vdash \neg(P \wedge Q); \\ \text{(iii) } \neg(P \wedge Q) \vdash Q \to \neg P; \\ \text{(iv) } Q \to \neg P \vdash \neg(P \wedge Q); \end{cases}$$
-

- De Morgan:
$$\begin{cases} \text{(i) } \neg(P \vee Q) \vdash \neg P \wedge \neg Q; \\ \text{(ii) } \neg P \wedge \neg Q \vdash \neg(P \vee Q); \\ \text{(iii) } \neg(P \wedge Q) \vdash \neg P \vee \neg Q; \\ \text{(iv) } \neg P \vee \neg Q \vdash \neg(P \wedge Q); \end{cases}$$
- Implicação Material:
$$\begin{cases} \text{(i) } P \to Q \vdash \neg P \vee Q; \\ \text{(ii) } \neg P \vee Q \vdash P \to Q; \end{cases}$$
- Redução da Disjunção:
$$\begin{cases} \text{(i) } P \vee Q \vdash \neg P \to Q; \\ \text{(ii) } \neg P \to Q \vdash P \vee Q; \\ \text{(iii) } P \vee Q \vdash \neg Q \to P; \\ \text{(iv) } \neg Q \to P \vdash P \vee Q; \end{cases}$$

- $\leftrightarrow$-Eliminação:
$$\begin{cases} \text{(i) } P, P \leftrightarrow Q \vdash Q; \\ \text{(ii) } Q, P \leftrightarrow Q \vdash P; \end{cases}$$
- Comutatividade da Equivalência: $\vdash (P \leftrightarrow Q) \leftrightarrow (Q \leftrightarrow P)$;
- Transitividade da Equivalência: $P \leftrightarrow Q, Q \leftrightarrow R \vdash Q \leftrightarrow R$;
- Lema da Substituição para Conectivos:
$$\begin{cases} \text{(i) } P_1 \leftrightarrow P_2 \vdash \neg P_1 \leftrightarrow \neg P_2; \\ \text{(ii) se } \# \in \{\to, \wedge, \vee\}, \text{ então:} \\ \quad \blacksquare \ P_1 \leftrightarrow P_2 \vdash P_1 \# Q \leftrightarrow P_2 \# Q; \\ \quad \blacksquare \ Q_1 \leftrightarrow Q_2 \vdash P \# Q_1 \leftrightarrow P \# Q_2. \end{cases}$$

2.7.14 Escólio: As regras da dupla negação, contraposição, negação da implicação, negação da conjunção, De Morgan, implicação material e redução da disjunção podem ser reescritas como equivalências.

2.7.14.1 Esquemas Complementares para Conectivos:

Regras Complementares para Conectivos:

- *Idempotência:* $\begin{cases} \text{(i)} \vdash P \wedge P \leftrightarrow P; \\ \text{(ii)} \vdash P \vee P \leftrightarrow P; \end{cases}$
- Membros da Equivalência: $\begin{cases} \text{(i)}\ P, Q \vdash P \leftrightarrow Q; \\ \text{(ii)}\ \neg P, \neg Q \vdash P \leftrightarrow Q; \\ \text{(iii)}\ P, \neg Q \vdash \neg(Q \leftrightarrow P); \\ \text{(iv)}\ \neg P, Q \vdash \neg(Q \leftrightarrow P); \end{cases}$
- Equivalência Material: $\begin{cases} \text{(i)} \vdash (P \leftrightarrow Q) \leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q); \\ \text{(ii)} \vdash \neg(P \leftrightarrow Q) \leftrightarrow (P \wedge \neg Q) \vee (\neg P \wedge Q); \end{cases}$

Negação da Equivalência: $\begin{cases} \text{(i)} \vdash \neg(P \leftrightarrow Q) \leftrightarrow (\neg P \leftrightarrow Q); \\ \text{(ii)} \vdash \neg(P \leftrightarrow Q) \leftrightarrow (P \leftrightarrow \neg Q); \end{cases}$

- Comutatividade: $\begin{cases} \text{(i)} \vdash P \wedge Q \leftrightarrow Q \wedge P; \\ \text{(ii)} \vdash P \vee Q \leftrightarrow Q \vee P; \\ \text{(iii)} \vdash (P \leftrightarrow Q) \leftrightarrow (Q \leftrightarrow P); \end{cases}$
- Associatividade: $\begin{cases} \text{(i)} \vdash P \wedge (Q \wedge R) \leftrightarrow (P \wedge Q) \wedge R; \\ \text{(ii)} \vdash P \vee (Q \vee R) \leftrightarrow (P \vee Q) \vee R; \\ \text{(iii)} \vdash (P \leftrightarrow (Q \leftrightarrow R)) \leftrightarrow ((P \leftrightarrow Q) \leftrightarrow R); \end{cases}$
- Absorção: $\begin{cases} \text{(i)} \vdash P \wedge (P \vee Q) \leftrightarrow P; \\ \text{(ii)} \vdash P \vee (P \wedge Q) \leftrightarrow P; \\ \text{(iii)} \vdash P \wedge (\neg P \vee Q) \leftrightarrow P \wedge Q; \\ \text{(iv)} \vdash P \vee (\neg P \wedge Q) \leftrightarrow P \vee Q; \end{cases}$
- Distributividade: $\begin{cases} \text{(i)} \vdash P \wedge (Q \vee R) \leftrightarrow (P \wedge Q) \vee (P \wedge R); \\ \text{(ii)} \vdash P \vee (Q \wedge R) \leftrightarrow (P \vee Q) \wedge (P \vee R); \end{cases}$

- Importação/Exportação: $\vdash P \to (Q \to R) \leftrightarrow (P \wedge Q) \to R$.

2.7.15 Esquemas e Regras de Introdução e Eliminação para Quantificadores

Regras de Introdução e Eliminação para Quantificadores:

- Generalização: Se $\begin{cases} \Gamma \vdash P \\ x \text{ não é livre em } \Gamma \end{cases}$, então $\Gamma \vdash \forall x\, P$;
- $\forall$-Eliminação: $\forall x\, P \vdash P(x|t)$;
- $\exists$-Introdução: $P(x|t) \vdash \exists x\, P$;
- $\exists$-Eliminação: Se $\begin{cases} \Gamma, \exists x\, P, P(x|y) \vdash Q \\ x \text{ não é livre em } \Gamma \cup \{Q\} \end{cases}$, então $\Gamma, \exists x\, P \vdash Q$.

Regras Básicas para Quantificadores:

- Negação de Fórmula Existencial: $\vdash \neg\exists x\, P \leftrightarrow \forall x\, \neg P$;
- Negação de Fórmula Universal: $\vdash \neg\forall x\, P \leftrightarrow \exists x\, \neg P$;

- Instanciação: Se $\begin{cases} \Gamma \vdash P \\ x \text{ não é livre em } \Gamma \end{cases}$, então $\Gamma \vdash P(x|t)$;

- Vacuidade: Se x não é livre em P, então $\begin{cases} \vdash \forall x\, P \leftrightarrow P; \\ \vdash \exists x\, P \leftrightarrow P; \end{cases}$

- Formas Congruentes: Se y não é livre em P, então $\begin{cases} \vdash \forall x\, P \leftrightarrow \forall y\, P(x|y); \\ \vdash \exists x\, P \leftrightarrow \exists y\, P(x|y); \end{cases}$

- Lema da Substituição para Quantificadores: $\begin{cases} \vdash \forall x\, (P \leftrightarrow Q) \leftrightarrow (\forall x\, P \leftrightarrow \forall x\, Q); \\ \vdash \forall x\, (P \leftrightarrow Q) \leftrightarrow (\exists x\, P \leftrightarrow \exists x\, Q). \end{cases}$

Estamos quase em condições de enunciar o *Princípio da Substituição para a Equivalência*, o qual é uma generalização dos Lemas da Substituição para Conectivos e

para Quantificadores. Para completar os requisitos, precisamos ainda de dois conceitos sintáticos: *substituição de fórmula por fórmula em uma fórmula*, e *escopo de uma variável em uma fórmula*.

Definição: Especificamos *substituição de fórmula por fórmula em uma fórmula* de uma forma análoga ao que foi feito anteriormente, na definição de substituição de variável por termo em um termo ou fórmula. Notamos a fórmula obtida substituindo S por P em Q por Q(S|P).

Definição: Um termo t (uma fórmula S) *está no escopo de* uma variável x em uma fórmula Q se Q possui uma subfórmula de uma das formas $\forall x\ R$ ou $\exists x\ R$ tal que t (S) ocorre em R.

Princípio da Substituição para a Equivalência:

Se $\Gamma \vdash Q(S|P_1) \leftrightarrow Q(S|P_2)$, então $\begin{cases} \Gamma \vdash P_1 \leftrightarrow P_2 \\ \text{S não está, em Q, no escopo de nenhuma variável livre em } \Gamma \cap \{P_1,P_2\} \end{cases}$

2.7.16 Esquemas e Regras Complementares para Quantificadores

Regras Básicas para Quantificadores:

- Negação de Fórmula Existencial: $\vdash \neg\exists x\ P \leftrightarrow \forall x\ \neg P$;
- Negação de Fórmula Universal: $\vdash \neg\forall x\ P \leftrightarrow \exists x\ \neg P$;
- Instanciação: Se $\begin{cases} \Gamma \vdash \mathbf{P} \\ \mathbf{x \text{ não é livre em } \Gamma} \end{cases}$, então $\Gamma \vdash P(x|t)$;
- Vacuidade: Se x não é livre em P, então $\begin{cases} \vdash \forall \mathbf{x}\ \mathbf{P} \leftrightarrow \mathbf{P}; \\ \vdash \exists \mathbf{x}\ \mathbf{P} \leftrightarrow \mathbf{P}; \end{cases}$

Formas Congruentes: Se y não é livre em P, então Lema da Substituição para Quantificadores: $\begin{cases} \vdash \forall \mathbf{x}\ (\mathbf{P} \leftrightarrow \mathbf{Q}) \leftrightarrow (\forall \mathbf{x}\ \mathbf{P} \leftrightarrow \forall \mathbf{x}\ \mathbf{Q}); \\ \vdash \forall \mathbf{x}\ (\mathbf{P} \leftrightarrow \mathbf{Q}) \leftrightarrow (\exists \mathbf{x}\ \mathbf{P} \leftrightarrow \exists \mathbf{x}\ \mathbf{Q}). \end{cases} \begin{cases} \vdash \forall \mathbf{x}\ \mathbf{P} \leftrightarrow \forall \mathbf{y}\ \mathbf{P(x|y)}; \\ \vdash \exists \mathbf{x}\ \mathbf{P} \leftrightarrow \exists \mathbf{y}\ \mathbf{P(x|y)}; \end{cases}$

Estamos quase em condições de enunciar o *Princípio da Substituição para a Equivalência*, o qual é uma generalização dos Lemas da Substituição para Conectivos e para Quantificadores. Para completar os requisitos, precisamos ainda de dois conceitos sintáticos: *substituição de fórmula por fórmula em uma fórmula*, e *escopo de uma variável em uma fórmula*.

Definição: Especificamos *substituição de fórmula por fórmula em uma fórmula* de uma forma análoga ao que foi feito anteriormente, na definição de substituição de variável por termo em um termo ou fórmula. Notamos a fórmula obtida substituindo S por P em Q por Q(S|P).

Definição: Um termo t (uma fórmula S) *está no escopo de* uma variável x em uma fórmula Q se Q possui uma subfórmula de uma das formas $\forall x\, R$ ou $\exists x\, R$ tal que t (S) ocorre em R.

Princípio da Substituição para a Equivalência:

$$\text{Se } \begin{cases} \Gamma \vdash \mathbf{P_1 \leftrightarrow P_2} \\ \textbf{S não está, em Q, no escopo de nenhuma variável livre em } \Gamma \cap \end{cases}, \quad \text{então}$$

$\Gamma \vdash Q(S|P_1) \leftrightarrow Q(S|P_2)$.

Regras Complementares para Quantificadores:

- Comutatividade: $\begin{cases} \vdash \forall \mathbf{x} \forall \mathbf{y}\ \mathbf{P} \leftrightarrow \forall \mathbf{y} \forall \mathbf{x}\ \mathbf{P}; \\ \vdash \exists \mathbf{x} \exists \mathbf{y}\ \mathbf{P} \leftrightarrow \exists \mathbf{y} \exists \mathbf{x}\ \mathbf{P}; \end{cases}$
- $\exists$-Importação: $\vdash \exists x \forall y\ P \leftrightarrow \forall y \exists x\ P$;
- Distributividade e Fatorabilidade de Quantificadores sobre Conectivos:
  (i) $\vdash \forall x\ (P \rightarrow Q) \rightarrow (\forall x\ P \rightarrow \forall x\ Q)$;
  (ii) $\vdash \forall x\ (P \wedge Q) \leftrightarrow \forall x\ P \wedge \forall x\ Q$;
  (iii) $\vdash \forall x\ P \vee \forall x\ Q \rightarrow \forall x\ (P \vee Q)$;
  (iv) $\vdash (\exists x\ P \rightarrow \exists x\ Q) \rightarrow \exists x\ (P \rightarrow Q)$;
  (v) $\vdash \exists x\ (P \wedge Q) \rightarrow \exists x\ P \wedge \exists x\ Q$;
  (vi) $\vdash \exists x\ (P \vee Q) \leftrightarrow \exists x\ P \vee \exists x\ Q$;
  (vii) $\vdash \forall x\ (P \leftrightarrow Q) \rightarrow (\forall x\ P \leftrightarrow \forall x\ Q)$;
- Relações Adicionais entre Quantificadores e Conectivos:
  (i) $\vdash \forall x\ (P \rightarrow Q) \rightarrow (\exists x\ P \rightarrow \exists x\ Q)$;
  (ii) $\vdash \forall x\ (P \leftrightarrow Q) \rightarrow (\exists x\ P \leftrightarrow \exists x\ Q)$;
  (iii) $\vdash (\forall x\ P \rightarrow \forall x\ Q) \rightarrow \exists x\ (P \rightarrow Q)$;
  (iv) $\vdash \forall x\ P \wedge \exists x\ Q \rightarrow \exists x\ (P \wedge Q)$;
  (v) $\vdash \forall x\ (P \vee Q) \rightarrow \forall x\ P \vee \exists x\ Q$;

(vi) $\vdash \exists x\,(P \to Q) \leftrightarrow \forall x\,P \to \exists x\,Q;$

$\vdash (\exists x\,P \to \forall x\,Q) \to \forall x\,(P \to Q);$

- Transporte de Quantificadores:

i) Se x não é livre em P, então

(i) $\vdash \forall x\,(P \to Q) \leftrightarrow (P \to \forall x\,Q);$

(ii) $\vdash \exists x\,(P \to Q) \leftrightarrow (P \to \exists x\,Q);$

(iii) $\vdash \forall x\,(P \wedge Q) \leftrightarrow P \wedge \forall x\,Q;$

(iv) $\vdash \exists x\,(P \wedge Q) \leftrightarrow P \wedge \exists x\,Q;$

(v) $\vdash \forall x\,(P \vee Q) \leftrightarrow P \vee \forall x\,Q;$

(vi) $\vdash \exists x\,(P \vee Q) \leftrightarrow P \vee \exists x\,Q;$

ii) Se x não é livre em Q, então

(i) $\vdash \forall x\,(P \to Q) \leftrightarrow (\exists x\,P \to Q);$

(ii) $\vdash \exists x\,(P \to Q) \leftrightarrow (\forall x\,P \to Q);$

(iii) $\vdash \forall x\,(P \wedge Q) \leftrightarrow \forall x\,P \wedge Q;$

(iv) $\vdash \exists x\,(P \wedge Q) \leftrightarrow \exists x\,P \wedge Q;$

(v) $\vdash \forall x\,(P \vee Q) \leftrightarrow \forall x\,P \vee Q;$

(vi) $\vdash \exists x\,(P \vee Q) \leftrightarrow \exists x\,P \vee Q.$

2.7.17 Regras Básicas de Igualdade:

Reflexividade da Igualdade: $\Gamma \vdash t = t$

Substituição em Sinais Funcionais:

$t_1=t_1', \ldots, t_n = t_n' \vdash f(t_1, \ldots, t_n) = f(t_1', \ldots, t_n')$

·Substituição em Sinais Predicativos:

$t_1=t_1', \ldots, t_n = t_n' \vdash f(t_1, \ldots, t_n) = f(t_1', \ldots, t_n')$

2.7.18 Esquemas e Regras Complementares da Igualdade:

- Simetria da Igualdade: $t = t' \vdash t' = t$
- Transitividade da Igualdade: $t = u, u = v' \vdash t = v$
- Princípio da Substituição para Igualdade em termos:

Se $\Gamma \vdash t_1 = t_2$, então $\Gamma \vdash u(x/t_1) = u(x/t_2)$.

- Principio da Substituição para Igualdade em fórmulas:

$$\Gamma \vdash t_1 = t_2$$

Se { X não está no escopo de nenhuma variável livre em $\Gamma \cap \{ t_1 = t_2 \}$

Então $\Gamma \vdash Q(t_1 = t_2) \leftrightarrow Q(x/t_2)$.

O conceito sintático correspondente ao conceito semântico de insatisfatibilidade é o conceito de trivialidade, dado na definição a seguir.

2.7.19 Definição: Dizemos que Γ é trivial se, para toda fórmula P, $\Gamma \vdash P$.

2.7.20 Definição: Dizemos que Γ é consistente se não existe P tal que $\Gamma \vdash P$ e $\Gamma \vdash \neg P$. Caso contrário dizemos que Γ é inconsistente.

O resultado abaixo fornece uma condição necessária e suficiente, com respeito à lógica clássica, para que uma coleção de fórmulas seja trivial.

2.7.21 Teorema: Γ é trivial se, e somente se, Γ é inconsistente

Finalmente, encerramos este capitulo dando alguns resultados que relacionam conceitos semânticos e conceitos sintáticos.

2.7.22 Teorema: Γ é insatisfatível se, e se somente se Γ é trivial

2.7.23 Teorema: Γ é satisfatível se, e somente se, Γ é não trivial

2.7.24 Teorema da Correção e da Completude (do cálculo clássico com respeito à semântica clássica) : $\Gamma \vdash P$ se, e somente se, $\Gamma \models P$.

As duas grandes linhas de pesquisa de Lógica em Informática e Inteligência Artificial, são Modelagem e Automatização do Raciocínio, mas não é possível considerá-las isoladamente sem considerar Lógica como um todo, pois tudo se interrelaciona e é interdependente [Buchsbaum & Pequeno 1994]

Modelagem consiste na construção de Lógicas aptas a satisfazer determinadas aplicações, e Automatização consiste na elaboração de algoritmos de prova automático para as Lógicas visadas.

Os métodos dos tableaux e de resolução não pertencem estritamente à lógica clássica. Existem versões dos mesmos, também, para certas lógicas paraconsistentes e/ou paracompletas, e certas lógicas modais. O método dos tableaux funciona para uma classe ainda maior de lógicas, inclusive para a lógica intuicionista. Não se crê que hajam limites definitivos para quaisquer métodos de automatização. O que se faz mister é um espírito de flexibilidade e adaptação [Buchsbaum & Pequeno 1990].

3 – O MÉTODO DOS TABLEAUX

3. 1 – DADOS GERAIS

Beth e Hintikka [Reeves 1983], que trabalharam separadamente, são considerados como precursores deste método dos tableaux. Este método consiste em apresentar a prova por refutação. O teorema é comprovado quando não há como contradize-lo, quando for infrutífero a tentativa de elaborar sistematicamente um modelo para a sua negação. Na verdade, o método examina uma certa fórmula e constata a impossibilidade da satisfação pela negação.

Trabalhar com método dos tableaux significa gerar uma árvore de fórmulas, conhecida por tableau, e deste tableau inicial, que é um único ramo no qual são construídas todas a premissas da base de conhecimento desejado e prova-se a impossibilidade da negação do cujo teorema.

Usa-se regras de expansão em cima deste tableaux inicial para construir uma árvore a partir de um nó ainda não usado. As subárvores vão surgindo a base das regras de expansão adequadas.

Se os ramos construídos satisfizerem a condição de fechamento do sistema de tableaux, e não puderem receber outro ramo, então ele será considerado fechado.

A questão é fechar todos os ramos, provando, desta forma, que a fórmula inicialmente negada é conseqüência lógica da base de conhecimento considerada. Semanticamente, a insatisfatibilidade da base de conhecimento, juntamente com a negação da formula considerada é satisfeita, quando todos os ramos do tableau estiverem fechados.

3.2 – CONCEITOS BÁSICOS

Alguns elementos são indispensáveis quando se busca a definição formal de um sistema de tableaux: uma linguagem inicial, uma linguagem de trabalho, uma função de

inicialização, responsável pela criação do tableau inicial, um conjunto de regras de expansão, responsável pela proliferação dos nós e um critério de fechamento.

3.2.1 Definição: Um tableau em uma linguagem L e uma arvore finita de nos cujo conteúdo e, no mínimo, uma formula de L e uma marca, a qual, por convenção, pertence ao conjunto {0, 1}. Se N e um nó, denotamos a formula de N por formula(N). Dizemos que um nó está marcado se sua marca é 1 Dizemos que um nó está em L se sua formula é de L.

3.2.2 Definição: Cada nó de um tableau pertence a um único nível, o qual é rotulado por algum número natural. Cada nó de nível n+1 é filho de um único nó de nivel n. Existe somente um nó de nível 0, que é chamado nó raiz ou nó inicial do tableau. Um nó e dito folha se este não possui mais sucessores. Se p é o nível de um no do tableau e não existe outro no de nível maior que p, então p é chamado proximidade do tableau.

3.2.3 Definição: Um ramo de um tableau é uma seqüência finita de nós $N_0 \ldots N_k$, tal que N_0 é o nó inicial do tableau e, para i = 1, ..., k, N_i é um nó sucessor de N_{i-1}, e N_k é um nó folha, isto é, o ramo termina em N_k.

3.2.4 Definição: Sejam L uma linguagem formal, L' uma extensão de L, τ a coleção de todos os tableaux em L', PF(τ) a coleção de todos o subconjuntos finitos de τ, θ a coleção de todos os ramos em L', I uma função de PF(L) em τ, F uma função de θ em {aberto, fechado}, e finalmente R uma coleção de funções parciais de L' x θ em PF(τ), tal que o domínio de cada uma dessas funções parciais e da forma L" x θ, onde L" é um subconjunto de L', podendo variar para cada função parcial. Um sistema de tableaux é uma quíntupla ordenada S = < L, L', I, F, R>, onde L é dita a linguagem inicial de S, L' é dita a linguagem de trabalho de S, I é dita a função de inicialização de S, F é dito o critério de fechamento de S e R é dita a coleção de regras de S.

3.2.5 Definição: Se r é uma regra de S e o seu domínio como função parcial é L' x θ, dizemos então que L" é o seu domínio como regra.

3.2.6 Definição: Sejam S = < L, L', I, F, R> um sistema de tableaux, P uma fórmula em L', N um nó em L' e r uma regra de S. A regra r é dita ser aplicável à fórmula P se P pertence ao domínio da regra r. A regra r é dita ser aplicável ao nó N se r é aplicável a fórmula(N).

3.2.7 Definição: Sejam S = < L, L', I, F, R> um sistema de tableaux, P uma fórmula em L' e r uma regra de S. P é dita uma fórmula excluída de S caso não exista regra r de S que seja aplicável a P.

3.2.8 Definição: Sejam S = < L, L', I, F, R> um sistema de tableaux e Γ uma coleção de fórmulas em L. I(Γ) e dito ser o tableau inicial para Γ em S.

3.2.9 Definição: Seja S = < L, L', I, F, R> um sistema de tableaux. Se ρ e um ramo em L' e F(ρ) = fechado, ρ é dito fechado em S; caso contrario, ρ é dito aberto em S.
3.2.10 Definição: Seja S = < L, L', I, F, R> um sistema de tableaux. Um ramo em L' é dito exaurido em S caso todos os seus nós que possuem fórmulas não excluídas estejam marcados.

3.2.11 Definição: Sejam S = < L, L', I, F, R > um sistema de tableaux e T,T' dois tableaux em L'. T' é dito uma extensão imediata de T em S se T é não exaurido e existe um nó N em T e uma regra r em S, tal que r seja aplicável a N, e T' pode ser obtido de T marcando-se N e acrescentando-se r (fórmula(N), ρ) em cada ramo aberto ρ de T onde N figure.

3.2.12 Definição: Sejam S um sistema de tableaux e $(T_i)_{i \in I}$ uma seqüência de tableaux de S, onde I é N ou i é da forma { 1, ..., n}, onde n $\in$ N, de modo que para cada i $\in$ I, se i > 0, então T_i é uma extensão imediata de T_{i-1} em S. Dizemos então que $(T_i)_{i \in I}$ é dita uma seqüência de desenvolvimento de tableaux em S. Se $(T_i)_{i \in I}$ é uma seqüência de desenvolvimento de tableaux em S, tal que T_0 é o tableaux inicial para Γ, então $(T_i)_{i \in I}$ é dita ser uma seqüência de desenvolvimento de tableaux em S para Γ.

3.2.13 Definição: Dizemos que uma seqüência T de desenvolvimento de tableaux em S é completa se as seguintes condições forem satisfeitas:

· se I e finito, então $(T_i)_{i \in I}$ termina com uma confutação ou com um tableau possuindo um ramo exaurido aberto;

· se I e infinito, então $(T_i)_{i \in I}$ tende para uma árvore-limite possuindo um ramo infinito exaurido aberto.

3.2.14 Escólio: Dado um tableau T_0 sempre é possível obter uma seqüência de desenvolvimento completa iniciando com T_0 utilizando, por exemplo, algum dos procedimentos conhecidos de busca em largura ou de busca em profundidade.

3.2.15 Definição: Sejam S = < L, L', I, F, R > um sistema de tableaux e T,T' dois tableaux em L'. T' é dito um desenvolvimento de T em S caso exista uma seqüência de tableaux T_0...,Tn,.. , tal que T_0 é T, e Tn é Γ.

3.2.16 Definição: Sejam S = < L, L', I, F, R > um sistema de tableaux, T um tableau em L' e Γ uma coleção de fórmulas em L'. T é dito um tableau para uma coleção de fórmulas Γ em S se T e um desenvolvimento do tableau inicial para Γ.

3.2.17 Definição: Sejam S um sistema de tableaux e Γ uma coleção de fórmulas na linguagem inicial de S. Uma confutação em S é um tableau em S no qual todos os seus ramos estão fechados. Uma canfutação para Γ em S é uma confutação em S que é um tableau para Γ em S.

3.2.18 Definição: Dizemos que um nó N gera um nó N' em um tableau T com respeito a um sistema de tableaux S, se uma das seguintes condições for satisfeita:

· existe um tableau T_1, tal que N é um nó não marcado de T_1, e existe um tableau T_2, obtido de T_1 através da aplicação de uma das regras de S sobre N (em particular, T_2 é uma extensão imediata de T_1 em S), de modo que N' é um dos novos nós de T_2 obtidos por esta aplicação, e T é um desenvolvimento de T_2 em S;

· existe um nó N", tal que N gera N" em T com respeito a S, e N" gera N' em T com respeito a S.

3.2.19 Definição: Dizemos que uma fórmula P gera uma fórmula Q, em um tableaux T com respeito a um sistema S, se existem nós N e N' tais que P é a fórmula de N, Q é a fórmula de N', e N gera N' em T.

Quando se pensa em fazer refinamento do método dos tableaux a primeira adaptação a ser feita no sistema de tableaux tradicional para a lógica quantificacional clássica visa eliminar repetições desnecessárias de nós na árvore de refutação. Por exemplo, dada uma árvore de refutação, se um determinado nó dessa árvore, cuja fórmula é $P \wedge Q$, está sendo expandido, ocorrerá o acréscimo do tableau, mostrado pela Figura 2, a cada ramo aberto descendente daquele nó. Porém, se um dos ramos, que sofrerá o acréscimo deste tableau, contiver um nó coma fórmula P' congruente a P, por exemplo, então não existe a necessidade que um novo nó contendo P seja acrescido ao ramo em questão.

Q

|

Q

Figura 2: Ramo Acrescentado pela Expansão da Fórmula $P \wedge Q$

Assim, as regras do sistema de tableaux tradicional sofrem alterações quanto aos tableaux que deverão ser acrescentado ao final de cada ramo descendente do nós em expansão

Portanto, a quíntupla $S_0 = < L_0, L_1, I_0, F, R_o >$, especificada no capítulo anterior, é alterada, neste refinamento, para $S_1 = <L_0, L_1, I_0, F, R_1 >$, onde P' e Q' denotam, respectivamente, fórmulas congruentes a P e Q, e R_1, é o novo conjunto de regras, dado a seguir.

Regra $\neg\neg P$ - Se P' ocorre no ramo, então nada é acrescentado ao ramo, contrário o acréscimo é realizado segundo a forma tradicional.

Regra $P \rightarrow Q$ - Se $\neg P'$ e Q' ocorrem no ramo, então nada é acrescentado;

-Se $\neg P'$ e Q' não ocorrem no ramo e $\neg P$ é congruente a Q, então o acréscimo é feito segundo a Figura 3;

- Se $\neg P'$ e Q' não ocorrem no ramo e $\neg P$ não é congruente Q, então o acréscimo é feito da forma tradicional.

$P \rightarrow Q$

|

Q

Figura 3. Regra $P \rightarrow Q$ modificada.

Regra $P \wedge Q$

- Se P' e Q' ocorrem no ramo, então nada é acrescentado;

-Se P' ocorre no ramo e Q' não ocorre no ramo, então o acréscimo é feito segundo a Figura 4;

- Se P' não ocorre no ramo e Q' ocorre no ramo, então o acréscimo é feito segundo a Figura 5.

- Se nem P' e nem Q' ocorrem no ramo, então o acréscimo é feito da forma tradicional;

$P \wedge Q$

|

Q

Figura 4: primeira Regra $P \wedge Q$ modificada.

$P \wedge Q$

|

P

Figura 5. Segunda Regra $P \wedge Q$ modificada.

Regra $P \vee Q$

- Se P' ou Q' ocorrem no ramo, então nada é acrescentado;

- Se P' e Q' não ocorrem no ramo e P é congruente a Q, então o acréscimo é feito segundo a Figura 6;

- Se P' e Q' não ocorre no ramo e P não é congruente a Q, então o acréscimo é feito da forma tradicional;

$P \vee Q$

|

Q

Figura 6. Regra P $\vee$ Q modificada.

Regra $\neg(P \rightarrow Q)$

- Se P' e $\neg$Q' ocorrem no ramo, então nada é acrescentado;

- Se P' ocorre no ramo e $\neg$Q' não ocorre no ramo, então o acréscimo é feito segundo as Figura 7;

- Se P' não ocorre no ramo e $\neg$Q' ocorre no ramo, então o acréscimo é feito segundo as Figura 8;

Se nem P' e nem $\neg$Q' ocorrem no ramo, então o acréscimo é feito da forma tradicional.

$\neg(P \rightarrow Q)$

|

$\neg Q$

Figura 7. Primeira Regra $\neg(P \rightarrow Q)$ modificada.

$\neg(P \rightarrow Q)$

|

P

Figura 8. Segunda Regra $\neg(P \rightarrow Q)$ modificada.

Regra $\neg(P \wedge Q)$

- Se $\neg$P' e $\neg$Q' ocorrem no ramo, então nada é acrescentado;

- Se $\neg$P' e $\neg$Q' não ocorrem no ramo e P é congruente a Q, então o acréscimo é feito segundo as Figura 9;

Se $\neg$P' e $\neg$Q' não ocorrem no ramo e P não é congruente a Q, então o acréscimo é feito da forma tradicional.

$\neg(P \wedge Q)$

|

$\neg P$

Figura 9. Regra $\neg(P \wedge Q)$ modificada.

$\neg(P \vee Q)$

- Se $\neg$P' e $\neg$Q' ocorrem no ramo, então nada é acrescentado;

- Se $\neg$P' ocorre no ramo e $\neg$Q' não ocorre no ramo, então o acréscimo é feito segundo a Figura 10;

- Se $\neg P'$ ocorre no ramo e $\neg Q'$ ocorre no ramo, então o acréscimo é feito segundo as Figura 11;

- Se nem $\neg P'$ e nem $\neg Q'$ ocorrem no ramo, então o acréscimo é feito da forma tradicional.

$\neg(P \vee Q)$

|

$\neg Q$

Figura 10. Primeira Regra $P \vee Q$ modificada.

$\neg(P \vee Q)$

|

$\neg P$

Figura 11. Segunda Regra $\neg(P \vee Q)$ modificada

Regra $\exists x$ P

Utiliza-se a Regra $\exists$ x P de R.

Regra $\forall x$ P

Utiliza-se a Regra $\forall x$ P de R, porém com os nós com as fórmulas $P(x/t_i)$, para i = 1,... n, onde $P(x/t_i)$, a menos de congruência, já ocorre no ramo, não devem ser acrescentados.

Regra $\neg\exists x$ P

Se uma fórmula congruente a $\forall x \neg P$ ocorre no ramo, então nada é acrescentado, caso contrário utiliza-se a Regra $\neg\exists$ x P de R.

Regra $\neg\forall x$ P

Se uma fórmula congruente a $\exists x \neg P$ ocorre no ramo, então nada é acrescentado, caso contrário utiliza-se a Regra $\neg\forall x$ P de R

No segundo refinamento altera somente a regra para fórmulas universais, que deverá ser como segue:

Regra $\forall x$ P – Acrescentam-se os nós com as fórmulas $P(x/t_i)$, para $i = {}_1,\ldots, n$, para as quais $P(x/t_i)$, a menos de congruência, não ocorre no ramo. Repete-se o nó com a

fórmula $\forall x$ P, somente quando existir no ramo uma fórmula Q, de um nó não marcado, tal que Q propague novos termos no ramo em questão.

No terceiro refinamento volta-se novamente para a regra para expansão de fórmulas universais. Neste momento altera também o processo de construção do tableau inicial, onde a função de inicialização, além substituir todas as variáveis livre do conjunto inicial de fórmulas por novas constantes e retirar os quantificadores vácuos de cada uma destas fórmulas, também realiza uma renomeação de variáveis, de modo que variáveis sob escopo de quantificadores diferentes possam ser mais facilmente identificados. Isto é feito para facilitar o processo de unificação. O processo de unificação modificado trata do ponto crucial desta alteração. No momento da expansão de uma fórmula universal obtemos, através deste processo, uma determinada lista de termos, que nos dirá para quais termos a fórmula universal deve ser instanciada, e se existe a necessidade ou não de repetição desta fórmula no ramo em expansão. [Fendt 2000]

Para se estabelecer um sistema de tableaux escolhe-se a linguagem inicial e de trabalho. Na definição da uma linguagem inicial do sistema de tableaux baseia-se a premissas e conclusões. Pode-se acrescentar à linguagem inicial novos sinais, se for preciso, para definir certas regras de expansão. Na lógica proposicional não modal acontece uma coincidência na linguagem de trabalho e linguagem inicial, mas isto não ocorre na lógica quantificacional clássica, onde tanto na linguagem de trabalho como na inicial precisam ser adicionados novas constantes.

O conjunto de regras e elaborado a partir da definição adequada de cada uma das regras, as quais dependem do nó a que estão sendo aplicadas, bem como do ramo a que este nó pertence. Em lógica de primeira ordem não modal, por exemplo, o tratamento dado a fórmulas quantificadas universalmente e existencialmente depende do ramo em que estes nós figuram. Porém, para fórmulas não quantificadas o ramo a que o nó pertence não é relevante.

A função de fechamento varia com respeito a lógica a que o tableau se destina. Usando, mais uma vez, a lógica clássica como exemplo, a função de fechamento trata da ocorrência de contradições, isto é. um ramo é dito fechado caso ocorram duas fórmulas contraditórias.

4 - MÉTODO DAS RESOLUÇÕES

Uma das facetas da inteligência humana é a capacidade de demonstrar teoremas. Isto foi pesquisado e desenvolvido na segunda metade dos anos 60 por Robinson e Smullyan, que, quando comprovaram a possibilidade de demonstrar teoremas no computador, introduziram a lógica como método computacional para a solução de problemas. E uma das áreas que mais faz uso desta técnica é a dos Sistemas Especialistas.

O objetivo principal da Prova Automática de Teoremas é provar que uma fórmula (teorema) é conseqüência lógica de outras fórmulas. Os métodos adotados normalmente usam a prova por refutação, que prova indireta, demonstrando que a negação da fórmula leva a inconsistências, isto é, se a negação de um teorema é falsa, então ele será verdadeiro.

Trabalha-se apenas com operadores como "e", "ou" e "não".

Afirma Barreto: "A teoria da resolução, proposta por Robinson em 1965 a partir dos trabalhos de Herbrand, Davis e Putnam, parte da transformação da fórmula a ser provada para a forma canônica conhecida como forma clausal."

O método é baseado em uma regra de inferência única, chamada REGRA DA RESOLUÇÃO, e utiliza intensivamente um algoritmo de casamento de casamento de padrões chamado ALGORITMO DE UNIFICAÇÃO.

O fato de ser possível associar uma semântica operacional a um procedimento de prova automática de teoremas permitiu a definição de uma linguagem de programação baseada em lógica, a linguagem PROLOG [Clocksin & Melish, 1981].

Ainda hoje a área de prova automática de teoremas permanece bastante ativa, sendo objeto de diversas conferências internacionais.

Algumas Definições:

PROVA: É a demonstração de que um teorema (ou fórmula) é verdadeiro.

FORMA NORMAL CONJUNTIVA: É quando uma fórmula F for composta de uma conjunção de outras fórmulas (F1 ^ F2 ^ ... ^ Fn).

FORMA NORMAL DISJUNTIVA: É quando uma fórmula F for composta de uma disjunção de outras fórmulas (F1 v F2 v ... v Fn).

- FORMA NORMAL PRENEX: É quando numa fórmula F, na lógica de primeira ordem, todos os quantificadores existentes prefixam a fórmula, isto é, se e somente se estiver na forma $Q_1x_1...Q_nx_n(M)$.

Onde:

$Q_ix_i = \forall x_i$ ou $\exists x_i$, e

(M) = uma fórmula que não contenha quantificadores.

- Procedimento para Obtenção da Forma Normal Prenex

1. Eliminar os conectivos lógicos $\rightarrow$ e $\leftrightarrow$ usando as seguintes leis:
   - $F \leftrightarrow G = (F \rightarrow G) \wedge (G \rightarrow F)$
   - $(F \rightarrow G) = \neg F \vee G$
2. Repetir o uso das seguintes leis:
   - $\neg \neg F = F$
   - $\neg (F \vee G) = \neg F \wedge \neg G$
   - $\neg (F \wedge G) = \neg F \vee \neg G$
   - $\neg (\forall x F(x)) = \exists x(\neg F(x))$
   - $\neg (\exists x F(x)) = \forall x(\neg F(x)$
   - 3. Estas leis são utilizadas para trazer os sinais de negação para antes dos átomos.

Procedimento para Obtenção da Forma Normal Prenex

4. Usar as leis abaixo de forma a mover os quantificadores para a esquerda da fórmula para obter a Forma Normal PRENEX.
   - $Qx\ F(x) \vee G = Qx\ (F(x) \vee G)$
   - $Qx\ F(x) \wedge G = Qx\ (F(x) \wedge G)$
   - $\forall x\ F(x) \wedge \forall x\ G(x) = \forall x\ (F(x) \wedge G(x))$
   - $\exists x\ F(x) \vee \exists x\ G(x) = \exists x\ (F(x) \vee G(x))$
   - $Q_1x\ F(x) \vee Q_2x\ G(x) = Q_1x\ Q_2z(F(x) \vee G(z))$
   - $Q_3x\ F(x) \wedge Q_4x\ G(x) = Q_3x\ Q_4z(F(x) \wedge G(z))$

EXEMPLO 1

$\forall x\ P(x) \rightarrow \exists x\ Q(x)$

- $\forall x\ P(x) \rightarrow \exists x\ Q(x) = \neg \forall x\ P(x) \vee \exists x\ Q(x)$

- $\exists x (\neg P(x)) \vee \exists x Q(x)$
- $\exists x (\neg P(x) \vee Q(x))$

EXEMPLO 2

$\forall x \forall y ((\exists z (P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)) =$

- $\forall x \forall y (\neg (\exists z (P(x,z) \wedge P(y,z))) \vee \exists u Q(x,y,u)) =$
- $\forall x \forall y (\forall z (\neg P(x,z) \vee \neg P(y,z))) \vee \exists u Q(x,y,u)) =$
- $\forall x \forall y \forall z \exists u (\neg P(x,z) \vee \neg P(y,z) \vee Q(x,y,u))$

- Eliminação dos quantificadores existenciais (Skolemização ou Funções de Skolem)
  - Quando uma fórmula está na forma normal Prenex, pode-se eliminar os quantificadores existenciais por uma função, se as variáveis estiverem no escopo do quantificador universal; caso estejam fora, substitui-se por uma constante.
  - As constantes e funções usadas para substituir as variáveis existenciais são chamadas constante e funções de Skolem
  - Ex.: $\forall x \exists y P(x,y)$

    Skolemizando: $\forall x P(x,f(x))$

    onde f(x) tem por único propósito garantir que existe algum valor (y) que depende de x pois está dentro do seu escopo. No entanto, se o quantificador existencial não residir no escopo do quantificador universal, como em $\exists y \forall x P(x,y)$, a variável quantificada existencialmente será substituída por uma constante $\forall x P(x,a)$ que assegure sua existência, assim como sua independência de qualquer outra variável.
- Procedimento para Obtenção da Forma Clausal
  - Cláusula é uma disjunção de literais

  1. Passar para a forma normal PRENEX.
  2. Skolemizar as variáveis quantificadas existencialmente.
  3. Abandona-se os quantificadores pré-fixados.

  EXEMPLO

  $\forall x \forall y ((\exists z (P(x,z) \wedge P(y,z)) \rightarrow \exists u Q(x,y,u)) =$

  - $\forall x \forall y (\neg (\exists z (P(x,z) \wedge P(y,z))) \vee \exists u Q(x,y,u)) =$

- $\forall x\ \forall y\ (\forall z\ (\neg P(x,z) \vee \neg P(y,z))) \vee \exists u\ Q(x,y,u)) =$
- $\forall x\ \forall y\ \forall z \exists u\ (\neg P(x,z) \vee \neg P(y,z) \vee Q(x,y,u))$
- $\forall x\ \forall y\ \forall z\ (\neg P(x,z) \vee \neg P(y,z) \vee Q(x,y,f(x,y,z)))$
- $\neg P(x,z) \vee \neg P(y,z) \vee Q(x,y,f(x,y,z))$

que é perfeitamente equivalente à fórmula original.

- Resolução
  - Seria útil, do ponto de vista computacional, que tivéssemos um procedimento de prova que realizasse, em uma única operação, a variedade de processos envolvidos no raciocínio, com declarações da lógica dos predicados.
  - Este procedimento é a RESOLUÇÃO, que ganha sua eficiência por operar em declarações que foram convertidas à forma clausal, como mostrado anteriormente.
  - A Resolução produz provas por REFUTAÇÃO, ou seja, para provar uma declaração (mostrar que ela é válida), a resolução tenta demonstrar que a negação da declaração produz uma contradição com as declarações conhecidas (não é possível de ser satisfeita).

A BASE DA RESOLUÇÃO

- É um processo interativo onde, em cada passo, duas cláusulas, denominadas cláusulas paternas, são comparadas (resolvidas), resultando em uma nova cláusula, dela inferida.
- A nova cláusula representa maneiras em que as duas cláusulas paternas interagem entre si.

Exemplo:

- Inverno v Verão
- ¬ Inverno v Frio

As duas cláusulas deverão ser verdadeiras (embora pareçam independentes, são realmente conjuntas).

- Agora, observamos que apenas um entre Inverno e ¬Inverno será verdadeiro, em qualquer ponto. Se Inverno for verdadeiro, então Frio também deverá ser, para garantir a verdade da segunda cláusula. Se ¬Inverno for verdadeiro, então também Verão deverá ser, para garantir a verdade da primeira cláusula.
- Assim, dessas duas cláusulas, podemos deduzir que

- Verão v Frio
- Esta é a dedução feita pelo procedimento de resolução.

- A resolução opera tirando suas cláusulas que contenham cada uma, o mesmo literal, neste exemplo Inverno.
- O literal deverá ocorrer na forma positiva numa cláusula e na forma negativa na outra.
- O resolvente é obtido combinando-se todos os literais das duas cláusulas paternas,
- exceto aqueles que se cancelam.
- Se a cláusula produzida for vazia, então foi encontrada uma CONTRADIÇÃO, o que valida a fórmula.
- Na Lógica Proposicional, o procedimento para produzir uma prova pela resolução da proposição S, com relação a um conjunto de axiomas F, é o seguinte:

1. Converter todas as proposições de F em cláusulas.
2. Negar S e converter o resultado em cláusulas. Acrescente-as ao conjunto de cláusulas obtidas no passo 1.
3. Repetir até que seja encontrada uma contradição ou não se possa fazer progresso:
   3.1. Escolher duas cláusulas, que serão chamadas cláusulas pais.
   3.2. Resolva-as. A cláusula resultante, denominada *resolvente*, será a disjunção de todos os literais de ambas as cláusulas pais, com a seguinte exceção:
      Se houver qualquer par de literais L e $\neg$ L, tal que uma das cláusulas pais contenha L e a outra $\neg$ L, então elimine tanto L como $\neg$ L do resolvente.
   3.3. Se o resolvente for uma cláusula vazia, terá sido encontrada uma contradição. Se não for, acrescente-o ao conjunto de cláusulas disponíveis para o procedimento.

EXEMPLO: P, $(P \wedge Q) \rightarrow R$, $S \vee T \rightarrow Q$, $T \vdash R$

- Primeiro convertemos os axiomas em cláusulas.

1. P
2. $\neg P \vee \neg Q \vee R$
3. $\neg S \vee Q$
4. $\neg T \vee Q$
5. T
6. $\neg R$

- Começamos então a escolher a par de cláusulas para resolver. Embora qualquer par de cláusulas possa ser resolvido, apenas aqueles pares que contenham literais complementares produzirão um resolvente com possibilidade de produzir uma cláusula vazia.

EXEMPLO: P, $(P \wedge Q) \rightarrow R$, $S \vee T \rightarrow Q$, T|— R

- Começamos por resolver com a cláusula ¬ R, pois ela é uma das cláusulas que deverão estar envolvidas na contradição que estamos tentando encontrar.

1.	P	
2.	¬ P v ¬ Q v R	
3.	¬ S v Q	
4.	¬ T v Q	
5.	T	
6.	¬ R	
7.	¬ P v ¬ Q	(2 e 6)
8.	¬ Q	(1 e 7)
9.	¬ T	(4 e 8)
10.	VAZIA	(5 e 9)

- Na Lógica Proposicional é fácil determinar que dois literais não possam ser verdadeiros ao mesmo tempo. (Simplesmente procure L e ¬ L)
- Na Lógica dos Predicados este processo de casamento ("matching") é mais complicado.Por exemplo Homem(Henry) e ¬ Homem(Henry) é uma contradição, enquanto que Homem(Henry) e ¬Homem(Spot) não o é.
- Assim, para determinar contradições, precisamos de um procedimento de matching que compare dois literais e descubra se existe um conjunto de substituições que os torne idênticos.
- O ALGORITMO DE UNIFICAÇÃO é um procedimento recursivo direto que faz exatamente isto.
- Para apresentar a unificação, consideramos as fórmulas como lista em que o primeiro elemento é o nome do predicado e os elementos restantes são os argumentos.
- Exemplo:

- (TentarAssassinar Marco Cesar)
- (TentarAssassinar Marco (Soberanode Roma))

- Para tentar unificar dois literais, primeiro conferimos se seus primeiros elementos são iguais. Caso contrário não há meio de serem unificados, independentemente de seus argumentos.
- Se o primeiro casar, podemos continuar com o segundo e assim por diante.
- Constantes, funções e predicados diferentes não podem casar, os idênticos podem. Uma variável pode casar com outra variável, ou com qualquer constante, função ou expressão de predicados.

1. Se L1 ou L2 for um átomo, então faça o seguinte:
   1.1. Se L1 e L2 forem idênticos, retornar NIL
   1.2. Caso contrário, se L1 for uma variável, faça
      1.2.1. Se L1 ocorrer em L2, retornar F;
      1.2.2. Caso contrário, retornar (L2/L1)
   1.3. De outro modo, se L2 for uma variável, faça
      1.3.1. Se L2 ocorrer em L1, retornar F;
      1.2.2. Caso contrário, retornar (L1/L2)
   1.4. Caso contrário, retornar F.
2. Se comprimento(L1) não for igual a comprimento(L2) retornar F.
3. Designar a SUBST o valor NIL. (ao final do procedimento, SUBST conterá todas as substituições utilizadas para unificar L1 e L2).
4. Para i=1 até o número de elementos de L1, faça:
   4.1. Chame UNIFICA com o i-ésimo elemento de L1 e o i-ésimo elemento de L2, colocando o resultado em S.
   4.2. Se S = F, retornar F.
   4.3. Se S não for igual a NIL, faça:
      4.3.1. Aplicar S tanto ao final de L1 como de L2.
      4.3.2. SUBST := APPEND(S,SUBST)
      4.3.3. Retornar SUBST

- Duas fórmulas-atômicas são contraditórias se uma delas puder ser unificada com o não da outra. Assim, por exemplo, Homem(x) e ¬ Homem(Spot) podem ser unificados.
- Isto corresponde à intuição que diz que não pode ser verdadeiro para todos os x, que Homem(x) se houver conhecimento de haver algum x, digamos Spot, para o qual Homem(x) é falso.
- Na lógica de predicados utilizaremos o algoritmo de unificação para localizar pares de fórmulas-atômicas que se cancelem.

1. Converter todas as declarações de F em cláusulas.
2. Negar S e converter o resultado em cláusulas. Acrescentá-las ao conjunto de cláusulas obtidas em 1.

Repetir até que uma contradição seja encontrada, e nenhum progresso possa ser feito, ou até que se tenha gasto um quantidade pré-determinada de esforço:

3.1. Escolher duas cláusulas e chamá-las de cláusulas pais.

3.2. Resolvê-las. O resolvente será a disjunção de todos os literais de ambas as cláusulas pais com as substituições apropriadas realizadas, ressalvando-se o seguinte:

3.2.1. Se houver um par de literais T1 e ¬ T2 tal que uma das cláusulas pais contenha T1 e a outra contenha T2, e ainda se T1 e T2 forem unificáveis, então nem T1 nem T2 devem aparecer no resolvente.

3.2.2. Chamaremos T1 e T2 literais complementares. Utilize a substituição produzida pela unificação para criar o resolvente.

3.3. Se o resolvente for uma cláusula vazia, então foi encontrada uma contradição. Se não for, acrescente-o ao conjunto de cláusulas disponíveis para o procedimento.

- Se a escolha de cláusulas a resolver em cada passo for feita de maneira sistemática, o procedimento de resolução encontrará uma contradição, se ela existir.
- Isto contudo, poderá levar muito tempo.
- Existem estratégias opcionais para acelerar o processo.

- resolver apenas pares de cláusulas que contenham literias complementares, pois somente essas resoluções produzem cláusulas novas mais difíceis de satisfazer que seus pais.
- Eliminar cláusulas do tipo tautologias e cláusulas que estejam incluídas em outras cláusulas (P v Q é incluída por P).
- Sempre que possível, resolver com uma das cláusulas que estamos tentando refutar ou com uma cláusula gerada por uma resolução com tal cláusula.
- Sempre que possível, dar preferência a cláusulas com um único literal.

EXEMPLO:

Homem(Marco)

Pompeiano(Marco)

$\forall x$ Pompeiano(x) $\rightarrow$ Romano(x)

Soberano(Cesar)

$\forall x$ Romano(x) $\rightarrow$ (LealA(x,Cesar) v Odiar(x,Cesar))

$\forall x \exists y$ LealA(x,y)

$\forall x \forall y$ (Homem(x) ^ Soberano(y)) v (TentarAssassinar(x,y) ^ LealA(x,y))

TentarAssassinar(Marco,Cesar)

Logo, Odiar(Marco, Cesar)

EXEMPLO:

- Primeiro convertemos os axiomas em cláusulas.

1. Homem(Marco)
2. Pompeiano(Marco)
3. $\neg$ Pompeiano(x1) v Romano(x1)
4. Soberano(Cesar)
5. $\neg$ Romano(x2) v LealA(x2,Cesar) v Odiar(x2,Cesar)
6. LealA(x3,f1(x3)
7. $\neg$ Homem(x4) v $\neg$ Soberano(y1) v $\neg$ TentarAssassinar(x4,y1) v $\neg$ LealA(x4,y1)
8. TentarAssassinar(Marco,Cesar)
9. $\neg$ Odiar(Marco,Cesar)

- Começamos então a escolher o par de cláusulas para resolver.

10. ¬ Romano(Marco) v LealA(Marco,Cesar) (SUBST(Marco,x2) em 5 e 9)

11. ¬ Pompeiano(Marco) v LealA(Marco,Cesar) (SUBST(Marco,x1 em 3 e 10)

12. LealA(Marco,Cesar) (2 e 11)

13. ¬ Homem(Marco) v ¬ Soberano(Cesar) v ¬TentarAssassinar(Marco,Cesar)
(SUBST(Marco,x4) e SUBST(Cesar,y1) em 7 e 12)

14. ¬ Soberano(Cesar) v ¬TentarAssassinar(Marco,Cesar) (1 e 13)

15. ¬TentarAssassinar(Marco,Cesar) (4 e 14)

16. VAZIA (8 e 15) [Roisenberg 2001]

5 - QUADRO COMPARATIVO DOS MÉTODOS TABLEAUX E DA RESOLUÇÃO

Pode-se elaborar uma dada forma de raciocínio escolhendo um tipo de lógica ou fazendo uma combinação de duas ou mais formas de lógica, baseado num método de prova automático, parcial ou total para tal lógica, os quais melhor se adaptam, o qual não é o propósito desta pesquisa que, apenas visa traçar um quadro comparativo entre os dois métodos de prova automática: o método dos tableaux e o método de resolução.

O método dos tableaux e um método de prova automático, isto é, um algoritmo sistemático que decide quando uma dada fórmula é conseqüência lógica de uma dada base de conhecimento.

A automatização do raciocínio lida com dois obstáculos básicos: todo algoritmo de automatização, no decorrer do seu processamento, provoca, em geral, um crescimento explosivo do espaço de busca; todo algoritmo de automatização esta sujeito a entrar em um processamento perpétuo para certos dados que não podem fornecer respostas negativas.

A autora do trabalho: "Refinamento para o método dos tableaux", utilizando como lógica subjacente a lógica clássica, propõe alterações sobre o método dos tableaux, de forma a aprimora-lo. Para isto, especifica e implementa três refinamentos sobre este método, de modo que a proliferação de dados intermediários desnecessários, produzidos ao longo do processamento, diminua, e também, o número de problemas para os quais obtemos soluções aumente. A principal alteração realizada sobre o método dos tableaux foi a sua associação a procedimento de unificação [Fendt 2000]. Com respeito ao primeiro refinamento ao método dos tableaux, foi incluído no algoritmo um procedimento de unificação, o qual é uma expansão do algoritmo de unificação usado tradicionalmente nos métodos de resolução, tal como apresentado, por exemplo, em [Lloyd 1987], outras apresentações deste algoritmo para o método da resolução podem ser encontradas em [Chang & Lee 1973], [Loveland 1978] e { Fitting 1990].

Fez uma série de testes, verificando que. na maioria dos casos, cada refinamento reduziu o número de nós da árvore, bem como, resolveu um número maior de

problemas. Além disso, observou-se a viabilidade de utilização do procedimento de unificação junto ao método dos tableaux.

Ambos os métodos, tanto dos tableaux como das resoluções fazem parte daquilo que se chama prova automática de teoremas. O método da Resolução trabalha com prova por refutação: "se a negação de um teorema é falsa, então ele será verdadeiro." Este método é baseado em uma regra de inferência única, chamada regra de resolução, e utiliza intensivamente um algoritmo de casamento de padrões chamado algoritmo de unificação.

No método dos tableaux, pode-se citar pontos muito específicos,pois, o algoritmo dos tableaux não necessita de transformações para qualquer forma normal. Segundo [Oppacher & Suen 1988], a principal vantagem do método dos tableaux, em relação, ao método tradicional e conhecido de resolução é fato deste evitar utilização de formas clausais. Alem disso, segundo [Buchsbaum 1988], o método dos tableaux é flexível e possui uma descrição bem simples.

Uma das principais idéias da programação em lógica é de que um algoritmo é constituído por dois elementos disjuntos: a lógica e o controle. O componente lógico corresponde à definição do que deve ser solucionado, enquanto que o componente de controle estabelece como a solução pode ser obtida. O programador precisa somente descrever o componente lógico de um algoritmo, deixando o controle da execução para ser exercido pelo sistema de programação em lógica utilizado. A tarefa do programador passa a ser simplesmente a especificação do problema que deve ser solucionado. Um programa em lógica é então a representação de determinado problema ou situação expressa através de um conjunto finito de um tipo especial de sentenças lógicas denominadas cláusulas. O paradigma fundamental da programação em lógica é o da programação declarativa, em oposição aos outros paradigmas. Mais precisamente, os sistemas de programação em lógica reduzem a execução de programas à pesquisa da refutação das sentenças do programa em conjunto com a negação da sentença que expressa a consulta, seguindo a regra: uma refutação é a dedução de uma contradição.

O termo PROLOG, que é o nome da principal implementação do paradígma de programação em lógica, vem de **PRO**gramação em **LOG**ica. Pode-se então expressar conhecimento (programas e/ou dados) em Prolog por meio de cláusulas de dois tipos: fatos e regras. Um fato denota uma verdade incondicional. Uma regra define as

condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira. Como fatos e regras podem ser utilizados conjuntamente, nenhum componente dedutivo adicional precisa ser utilizado. As características mais marcantes dos sistemas de programação em lógica em geral - e da linguagem Prolog em particular - são as seguintes: Especificações são Programas; Capacidade Dedutiva; Não-determinismo; Reversibilidade das Relações; Tríplice Interpretação dos Programas em Lógica; Recursão.

Um dos primeiros usos de Prolog foi a representação e análise de subconjuntos da linguagem natural. Aplicação que fez Alain Colmerauer a desenvolver sua primeira implementação. São as principais aplicações: Sistemas Baseados em Conhecimento; Sistemas de Bases de Dados; Sistemas Especialistas; Processamento da Linguagem Natural; Educação;Arquiteturas Não-Convencionais (Parlog, Concurrent Prolog, Prolog++, etc.)

Algumas das principais características da linguagem Prolog são: É uma linguagem orientada ao processamento simbólico. Representa uma implementação da lógica como linguagem de programação. Apresenta uma semântica declarativa inerente à lógica. Permite a definição de programas reversíveis, isto é, programas que não distinguem entre os argumentos de entrada e os de saída. Permite a obtenção de respostas alternativas. Suporta código recursivo e iterativo para a descrição de processos e problemas, dispensando os mecanismos tradicionais de controle, tais como while, repeat, etc. Permite associar o processo de especificação ao processo de codificação de programas. Representa programas e dados através do mesmo formalismo Incorpora facilidades computacionais extralógicas e metalógicas

6 - PROLOG: SUA HISTÓRIA E IMPLEMENTAÇÃO

Como falar em programação em lógica sem mencionar o Prolog? Uma das principais idéias da programação em lógica é de que um algoritmo é constituído por dois elementos disjuntos: a lógica e o controle. O componente lógico corresponde à definição do que deve ser solucionado, enquanto que o componente de controle estabelece como a solução pode ser obtida.

O programador precisa somente descrever o componente lógico de um algoritmo, deixando o controle da execução para ser exercido pelo sistema de programação em lógica utilizado

A tarefa do programador passa a ser simplesmente a especificação do problema que deve ser solucionado. Um programa em lógica é então a representação de determinado problema ou situação expressa através de um conjunto finito de um tipo especial de sentenças lógicas denominadas cláusulas.

O paradigma fundamental da programação em lógica é o da programação declarativa, em oposição aos outros paradigmas. Mais precisamente, os sistemas de programação em lógica reduzem a execução de programas à pesquisa da refutação das sentenças do programa em conjunto com a negação da sentença que expressa a consulta, seguindo a regra: uma refutação é a dedução de uma contradição

O termo PROLOG, que é o nome da principal implementação do paradigma de programação em lógica, vem de PROgramação em LÓGica

Pode-se então expressar conhecimento (programas e/ou dados) em Prolog por meio de cláusulas de dois tipos: fatos e regras. Um fato denota uma verdade incondicional. Uma regra define as condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira. Como fatos e regras podem ser utilizados conjuntamente, nenhum componente dedutivo adicional precisa ser usado.

As características mais marcantes dos sistemas de programação em lógica em geral - e da linguagem Prolog em particular - são as seguintes: Especificações são Programas; Capacidade Dedutiva; Não-determinismo; Reversibilidade das Relações; Tríplice Interpretação dos Programas em Lógica; Recursão.

Um dos primeiros usos de Prolog foi a representação e análise de subconjuntos da linguagem natural. Aplicação que fez Alain Colmerauer ('71) a desenvolver sua primeira implementação. As principais aplicações são: Sistemas Baseados em Conhecimento; Sistemas de Bases de Dados; Sistemas Especialistas
Processamento da Linguagem Natural; Educação; Arquiteturas Não-Convencionais (Parlog, Concurrent Prolog, Prolog++, etc.)

Algumas das principais características da linguagem Prolog são: É uma linguagem orientada ao processamento simbólico; representa uma implementação imperfeita da lógica como linguagem de programação; apresenta uma semântica quase declarativa inerente à lógica; permite a definição de programas reversíveis, isto é, programas que não distinguem entre os argumentos de entrada e os de saída; permite a obtenção de respostas alternativas; suporta código recursivo e iterativo para a descrição de processos e problemas, dispensando os mecanismos tradicionais de controle, tais como while, repeat, etc; permite associar o processo de especificação ao processo de codificação de programas; representa programas e dados através do mesmo formalismo; incorpora facilidades computacionais extralógicas e metalógicas.

A Quinta Geração de computadores, anunciada pelo Ministério do Comércio e Indústria Japonês (MITI), em outubro de 1981, foi um plano audacioso porque visava produzir computadores inteligentes, combinando conceitos de hardware e software. Adivinhem que linguagem foi escolhida ? Uma linguagem que até então não fora usada pela inteligência artificial, a linguagem Prolog. Isto surpreendeu o mundo computacional e fez com que a linguagem do Prolog se projetasse no cenário internacional, levando estudiosos da área da inteligência artificial travarem conhecimento com esta linguagem. Em alguns círculos científicos chegou-se a acreditar que esta linguagem era a solução para os problemas da inteligência artificial e chegaram a dar uma atenção exagerada a esta linguagem resumindo cursos só em Prolog. Importante observar que o Prolog não implementa perfeitamente o paradigma de programação lógica. A implementação é imperfeita pois na lógica os conectivos $\vee$ e $\wedge$ são comutativos e que não ocorre com Prolog. Ex Seja A: - B, C.

B: - x = 2.

C: - x = 3.

(isto é para A ser verdade ele faz B e C serem verdade, logo inicialmente x vale 2 e ao término x vale 3.

Mas se escrever:

A : - C, B

Ao final da execução x = 2! Logo o "e" não é comutativo!

Com relação ao "ou" o exemplo pode ser ainda mais drástico! Basta olhar o cálculo recursivo. Se trocar a ordem de declaração entra em ciclo infinito e o programa não pára.

Logo Prolog é uma tentativa de atingir a programação em lógica não conseguindo por ter conectivos não comutativos e não ser declarativa (a ordem das cláusulas influencia no resultado).

"Por outro lado, Prolog incorpora um mecanismo de resolução automática de problemas: resolução. Desta forma, Prolog pode ser considerado como mais do que uma linguagem, e se aproxima do conceito de "shell"" [Barreto, 1999].

Foi por volta de 1970 na França que surgiu a idéia de fazer esta nova linguagem, que acabou sendo concebida P. Russel em 1973. Kowalsky [Kowalsky, 1983] teve o mérito de transformar esta linguagem numa linguagem de fato de programação, mas a popularidade de Prolog veio só com a Quinta Geração de computadores dos japoneses.

Para exemplificar, tomamos a liberdade de descrever um programa baseado na sintaxe do Prolog II de Marseille, citado por Jorge Muniz Barreto em seu livro Inteligência Artificial, página 286 e 287.

Para ilustrar, seja usando a sintaxe do Prolog II de Marseille, um programa para dar dados sobre uma árvore genealógica. (As linhas tracejadas correspondem a outras instruções semelhantes).

homem(Georges) ->;

.....

homem(Richard) ->;

.....

mulher(Mary) ->;

.....

mulher(Doralice) ->;

.....

```
pai(Georges,Charles) ->;
.....
pai(Ane,Mark) ->;
.....
mae(George,Rose) ->;
.....
mae(Ane,Dora) ->;

 avohomem(x,f) ->
     pai(x,z)        pai(z,f);
avohomem(x,f) ->
    pai(x,z)         pai(z,f);
avomulher(x,f) ->
     pai(x,z)        mae(z,f);
avomulher(x,f) ->
    mae(x,z)         mae(z,f);
irmao(x,y) ->
   mae(x,z)       mae(y,z)        dif (x,z)       homem (y);
 irmao(x,y) ->
    pai(x,z)        pai (y,z)        dif (x,y)        homem(y);
filho (x,y) -> pai (y,x) homem(y);
filho (x,y) -> mae (y,x) homem(y);

;End world: Normal
```

Nota-se, por inspeção que o programa contém um conjunto de cláusulas representando fatos declarados (quem é homem e mulher, quem é pai e mãe de quem) e regras definindo o que é avô (variável chamada "avohomem" no programa), avó (da mesma forma "avomulher"), irmão, filho. Ambos essencialmente com a mesma forma, apenas o segundo membro das regras é vazio o que significa que não existe condição necessária para o primeiro membro ser verdade. Uma cláusula tem uma cabeça e uma cauda. A cabeça é o primeiro membro (lado esquerdo) da cláusula e a cauda o segundo membro (lado direito). Uma cláusula que tem a cauda vazia é um fato pois será verdade

de modo incondicional e se a cauda contiver uma fbf será verdade se sua cauda o for. Além disto as variáveis e predicados usam identificadores (ex: mae, x) começando por letras minúsculas e constantes por letras maiúsculas (ex: Mary, Doralice). Todas as cláusulas terminam por ";". Em uma seqüência de fórmulas atômicas (ex: pai(x,y)) separadas por brancos, os brancos se interpretam como o "e" lógico e cláusulas começando com mesmo predicado indicam "ou" .

Existem várias sintaxe: de Marseille, de Edimburg, do "Imperial College", a "Simple". Destas parecem que mantém a popularidade as duas primeiras. Afirma que a última oferece grandes vantagens.

Transcrevemos novamente uma parte do livro de Jorge Muniz Barreto, já citado, página 288:

Para ilustrar a sintaxe de Edinburg apresenta-se o código para resolver o problema das Torres de Hanói.

```
/*  Use o comando >hanoi(4)?
    para resolver o problema com 4 discos
*/

hanoi (N) :- mova(N,  left,  center,  righ).

mova(0, ~, ~, ~) = !.

mova(N, A,  B, C) :-
M is N - 1,
mova(M, A, C, B),
mova(fundo, A, B),
 mova(M, C, B, A).

mova(fundo, A, B) :-
 print("Mova o disco "),
      print(A),
      print(" para "),
```

```
print(B),
nl.
```

São duas as semânticas presentes no Prolog: a declarativa e a operacional.

Podemos resumir as principais características do Prolog no seguinte:

a) Backtracking está embutido no programa o que permite achar múltiplas soluções . b) Programa e dados tem a mesma estrutura.,

c) Prolog favorece o paradigma da programação em Lógica.

d) Devido a sua estrutura quase declarativa, é adequado ao uso de paralelismo.

e) Sua concisão e conseqüente curto tempo de desenvolvimento fazem de Prolog uma linguagem útil para prototipagem.

f) Prolog sendo uma linguagem de alto nível requer muitos recursos computacionais, motivando avanços no hardware de computadores.

Existem várias linguagens Lógicas paralelas atualmente. Uma das primeiras foi "Concurrent Prolog", proposta por Shapiro [Shapiro,1983]. Usa paralelismo "e" e "ou". Concurrent Prolog usa as primitivas "freeze" e "commit" representadas respectivamente por "?" e "/" com a finalidade de eliminar possíveis problemas com paralelismo[Barreto 1999].

Pode-se então expressar conhecimentos (programas e/ou dados) em Prolog por meio de cláusulas de dois tipos: fatos e regras. Um fato denota uma verdade incondicional, enquanto que uma regra define as condições que devem ser satisfeitas para que uma certa declaração seja considerada verdadeira.

6.1-A IMPLEMENTAÇÃO DOS MÉTODOS DOS TABLEAUX E DA RESOLUÇÃO

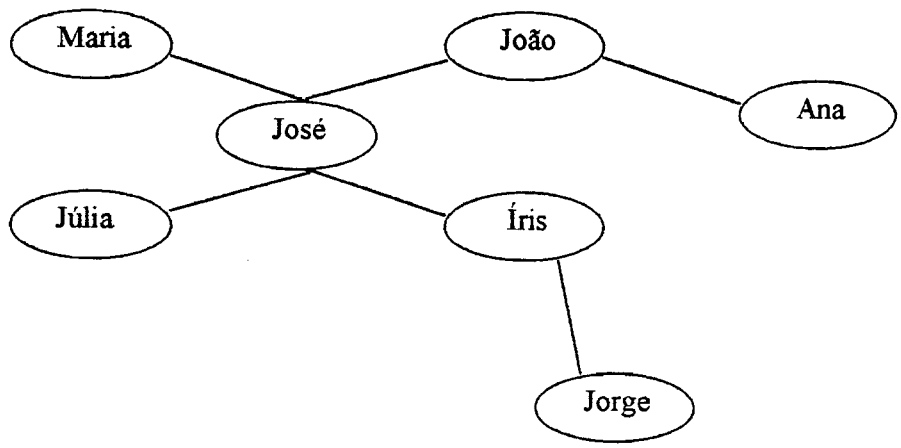
Deseja-se fazer uma implementação comparativa entre o método do tableau e das resoluções para caracterizar melhor o quadro comparativo.

A função inicializa_arvore constrói o tableau inicial. Recebe como parâmetro uma lista de fórmulas contendo as fórmulas da teoria e a negação do possível teorema. Retorna um ponteiro para o tableau inicial ou uma mensagem indicando que o tableau inicial é inconsistente. Observação: vai-se usar a versão do Prolog 3.3

```
(defun inicializa arvore (I algoritmo)
( let (a no b m con tipo form termos_ fechados.
 (setf list _var nil)
(setf b 1 )
(setf a (length 1))
(loop
    (setf no (newatom "no" b))
if(>b 1) (let(vf)
    (setf vf (list no))
    (putt (car m) vf'filhos)))
(if (= b 1 ) (putt no nil 'pai))
(if (> b 1 ) (putt no (car m) 'pai))
(setf form (come_parenteses (car 1)))
(setf form (retira_equivalencia form))
(setf tipo (ver_tipo form))
(setf form (retira_vacuo form))
(if (atom form) (setf form (list form)) )
(setf form (come_parenteses form))
(if(equal algoritmo':algoritmo3)
    (putt no (propaga_termos form tipo)'rep)
    (putt no 0 'rep))
(if (equal algoritmo ': algoritmo4)
    (setf form (marca form'p)))
    (putt no form 'formula)
(putt no nil 'n_cte)
(putt no nil 'l_fec)
(if (= b a) (putt no  nil 'filhos))
(putt no nil 'inst)
(if(=b 1)
(putt con 'sim 'consist)

 (let()
     (setf con (ver consist form tipo (car m) algoritmo))))
     (putt no (get con'consist)'consistencia)
(if(eq (get con'consist)'nao)
     (retum-from inicializa arvore'inconsistente))
(setf m (cons no m))
(setf l (cdr 1))
(if (= b a) (let ()
     (putt (car m) 1 'n_cte)
     (setf termos_fechados (acha_termos_fechados))
     (putt (car m) termos_fechados'l_fec)
         (return)))
(setf b (1+ b)))
(setf m (reverse m))
m))
```

Com respeito ao método das resoluções sugerimos a implementação de uma árvore genealógica. Considere a árvore genealógica mostrada na figura abaixo. Dela extraem-se fatos; exemplo: João é progenitor de José; progenitor (maria, josé). Observação: Todos têm a relação de progenitores.



Árvore genealógica

progenitor(maria, josé).
progenitor(joão, josé).
progenitor(joão, ana).
progenitor(josé, júlia).
progenitor(josé, íris).
progenitor(íris, jorge).

O programa acima compõe-se de seis cláusulas cada uma das quais denota um fato acerca da relação progenitor.

Se o programa for submetido a um sistema Prolog, este será capaz de responder algumas questões sobre a relação ali representada. Por exemplo: José é o progenitor de Íris?

No prompt de Prolog, você consultaria

?- Progenitor (josé, íris).

Prompt sim *sua consulta*

? - Resposta

Outras consultas

? - progenitor(luís, maria).
Não

. Mais

? - progenitor(josé, X).
X =júlia;
X = íris;
Não

? - progenitor(X,Y)
X=maria y=josé;
X= joão y =josé;
X=joão y = ana.

Observe que desta forma, a consulta esgota todas as possibilidades.
É possível, também, uma composição

? - progenitor(X,Y), progenitor(Y, ,jorge).
X=josé Y = íris

Na verdade você está querendo saber quem é um dos avós de jorge. Isso pode ser expresso por uma regra

Avô(X,Z):-
Progenitor (X,Y)
Progenitor(Y,Z).

A consulta agora seria
?- avô(X,jorge).
X=josé

Poderiam ser criadas regras para irmão, tia, bisavó, etc.

```
mzi! Prolog Listener
Version: 3.3 Mar96+ 16-bit 286 Protected Mode Windows
May 12 1996 18:46:51

Consulting ENV.PRO

yes
?- progenitor(jose,iris).consult('C:\\AMZIEXPL\\ARVGEN.PRO')
yes
?- progenitor(luis,maria).consult('C:\\AMZIEXPL\\ARVGEN.PRO')
no
? - progenitor(jose,x).consult('C:\\AMZIEXPL\\ARVGEN.PRO')
x = julia
x = iris
no
?- progenitor(x,y).consult('C:\\AMZIEXPL\\ARVGEN.PRO')
x=maria y=jose;
x=joao  y=jose;
x=joao  y=ana.
?-progenitor(x,y), progenitor(y,jorge).consult('C:\\AMZIEXPL\\ARVGEN.PRO')
x=jose y =iris
?-avo(x,jorge).consult('C:\\AMZIEXPL\\ARVGEN.PRO')
x=jose.
```

7 – CONCLUSÃO

A lógica, como estudo do mecanismo do pensamento, que tem mais de vinte séculos de história, método da representação do conhecimento e modelagem do raciocínio, sem dúvida desempenha importantíssimo papel dentro da Inteligência Artificial. Falando em Inteligência Artificial e lógica não podemos deixar de mencionar a complexidade computacional.

Há necessidade de se tecer algum comentário sobre a complexidade computacional. O computador pode ser considerado como máquina de resolver problemas, logo, é natural imaginar que tanto a possibilidade de resolver um problema específico, como quanto vai ser gasto em recursos na tarefa, dependem da máquina usada. Ao fato de que um problema possa ser resolvido com recursos finitos chama-se computabilidade [Kfoury; Moll,.& Arbid, 1982] e a quantidade de recursos envolvidos complexidade[Goldschlager & Lister, 1982]. Fala-se em computabilidade prática; por exemplo, um problema que requeira um tempo de 100 anos do mais rápido computador disponível não é praticamente computável.

As duas grandes linhas de pesquisa de Lógica em Informática e Inteligência Artificial são Modelagem e Automatização do Raciocínio, mas não é possível considerá-las isoladamente sem considerar Lógica como um todo, pois tudo se inter-relaciona e é interdependente.

Modelagem consiste na construção de Lógicas aptas a satisfazer determinadas aplicações, e Automatização consiste na elaboração de algoritmos de prova automático para as Lógicas visadas.

Os métodos dos tableaux e de resolução não pertencem estritamente à lógica clássica. Existem versões dos mesmos, também, para certas lógicas paraconsistentes e/ou partacompletas,[Buchsmann & Pequeno 1993] e certas lógicas modais. O método dos tableaux funciona para uma classe ainda maior de lógicas, inclusive para a lógica intuicionista. Acredita-se que não hajam limites definitivos para quaisquer métodos de automatização. O que se faz mister é um espírito de flexibilidade e adaptação.

Após a informação de que Beth e Hintikka foram os precursores do assim chamado sistema de tableaux, que é um método de prova por refutação, no qual se prova o teorema pela impossibilidade de sua negação através da construção sistemática de um modelo, e, que o método dos tableaux consiste na geração de uma árvore de fórmulas (tableau) onde nos nós são construídos sub-árvores com regras de expansão, cujos ramos sempre são fechados; e. por outro lado ter verificado que o método da resolução foi introduzido por Robinson e Smullyan, em 1960, como método computacional, que se baseia na prova por refutação, baseada numa regra única da resolução, utilizando algoritmo de unificação, constata-se muita semelhança entre ambos os métodos.

O quadro comparativo dos Métodos dos Tableaux e das Resoluções de fato mostra muitos pontos em comuns, ou seja, existe uma afinidade entre ambos os métodos. O Método da Resolução é um método tradicional e conhecido. Ambos partem das provas automáticas de teoremas. Método da Resolução trabalha com refutação, com contradição e algoritmo de unificação. Neste quadro comparativo entre o método dos tableaux e o método das resoluções fica evidente que o método dos tableaux leva vantagem porque evita usar cláusulas em suas formas.

Agora com respeito a implementação exige muita de paciência. Pois estilo de programação varia de implementação para implementação. Em geral, é baseada na experiência passada. Portanto, é lógico que inicialmente é preciso familiarizar-se com uma sintaxe de uma implementação particular de um determinado paradigma com a leitura sobre programas nesse paradigma, escolher o seu estilo.

Que tal, como exercício, procurar uma linguagem preferida. Como se pode observar, não é possível de noite para o dia formar um estilo em programação. É necessário vivência e convivência. Essa é a regra geral que deve ser seguida para que se adquira um estilo próprio. Não há mágica, mas fracassos e sucessos. Especialistas não deixam de serem aprendizes. O sucesso está um pouco além de onde as pessoas comuns desistem.

8 - REFERÊNCIAS BIBLIOGRÁFICAS

[Barreto 1999] BARRETO, Jorge Muniz. *Inteligência Artificial no Limiar do Século XXI*. 2ª ed. Florianópolis:1999.

[Buchsbaum, 2001] "Notas de Lógica - Alguns Objetivos para um Primeiro Curso de Graduação ou Pós-Graduação", Apostila não publicada, de Arthur Buchsbaum.

[Buchsbaum 1988] BUCHSBAUM, Arthur. *Um Método Automático de Prova para a Lógica Paraconsistente*. Dissertação de Mestrado do Departamento de Informática da Pontifícia Universidade Católica do Rio de Janeiro, 1988.

[Buchsbaum & Pequeno 1990] BUCHSBAUM, Arthur & Pequeno Tarcisio. *O Método dos Tableaux Generalizado e sua Aplicação ao Raciocínio Automático em Lógicas Não Clássicas*. O que nos faz pensar - Cadernos do Departamento de Filosofia da Pontifícia Universidade Católica do Rio de Janeiro, n° 3, setembro de 1990.

[Buchsbaum & Pequeno 1993) BUCHSBAUM, Arthur & Pequeno Tarcisio. *Uma família de Lógicas Paraconsistente e/ou Partacompletas com Semânticas Recursivas*. Coleção Documentos, Série Lógica e Teoria da Ciência, n° 14, Instituto de Estudos Avançados. Universidade de São Paulo, setembro de 1993.

[Buchsbaum & Pequeno 1994] BUCHSBAUM, Arthur & Pequeno Tarcisio. *Automated Deduction with Non Classical Negations*. Proceedings of 3rd Workshop on Theorem Proving with Analytic Tableaux and Related Methods - 1994 - pp. 51-64

[Carrol, 1881 e 1891] CARROL, Lewis. *Symbolic Logic – The game of Logic", Mc Millan,1881 e 1891 (Repinted in Fac-simele por Dover)*

[Chang & Lee 1973] CHANG, Chin-Liang & Lee, Richard Char-Tung. *Simbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[Church 1936] CHURCH, Alonzo. *A Note on the Entscheidungsproblem*. The Journal of Symbolic Logic vol.l, pp 40-41, 101-102, 1936.

Clocksin & Melish, 1981] CLOCKSIN, W.F & MELISH, C.S. Programming in Prolog. Springer-Verlag, Berlim, 1981.

[Copi 1978] COPI, Irving M.. *Introdução à Lógica*. Editora Mestre Jou, 2ª edição, 1978.

[Costa 1994J COSTA, Newton C. A.da. Ensaio sobre os Fundamentos da Lógica, Editora Hucitec, 2ª edição, 1994.

[Ebbinghaus & Flum & Thomas 1989] EBBINGHAUS, H. D. & Flum, J. & Thomas W.. *Mathemathical Logic*. Springer-Verlag, New York,. 1989.

[Enderton 1972] ENDERTON, Herbert B.. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[Fendt 2000] FENDT, Leticia Carvalho Pivetta. *Refinamentos para o método dos tableaux*. Dissertação de Mestrado em Ciência da Computação da Universidade Federal de Santa Catarina, Florianópolis, 2000.

[Fitting 1990J FITTING, Melvin. *First-Order Logic and Automated Theorem Proving*. Springer Verlag, 1990.

[goldschlager & lister, 1982] GOLDSCHLAGER, L. & LISTER, A. *Computer science: a modern introduction*. Prentice Hall, New Jersey, 1982.

[Kfoury, Moll & Arbid, 1982] KFOURY, A. J.; MOLL, R. N.& ARBID, M. A. *A*

Kowalsky 1983) KOWALSKY, R. *Logic Programing*. In IFIP'83, 1983 p. 133-145. *programmming Approach to Computability*. Springer Verlag, 1982.

[Lloyd 1987] LLOYD, J. W.. *Foundations of Logic Programming*. Springer Verlag, 1987.

[Loveland 1978] LOVELAND, Donald W.. *Automated Theorem Proving*. North-Holland Publishing Company, 1978.

[Oppacher & Suen 1988] OPPACHER, F. & Suen E. Harp: *A Tableau-Based Theorem Prover.* Journal of Automated Reasoning Kluner Academic Publishers, 1988.

[Prawitz 1989] PRAWITZ, Dag. *Natural Deduction - A Proof-Theoretical Stydy.* Almqvist & Wiksell, 1965.

[Roisenberg, 2001] ROISENBERG, http://www.inf.ufsc.br/~mauro/curso/curso.html

[Shapiro, 1983] SHAPIRO, e. Systems Programming in Concurent Prolog. Relatório Técnico, ICOT, 1983.

ANEXO 1

O que é um teorema?

De acordo com Euclides postulados são afirmações geométricas que dispensam provas. São evidentes por si. Por ex.: Pode-se traçar uma linha reta de um ponto qualquer a outro ponto qualquer. Euclides chamou de axiomas as proposições mais gerais também evidentes por si. Ex.: Coisas iguais a uma terceira são iguais entre si.

Muito bem, toda proposição demonstrada, isto é, comprovada, a partir de proposições primitivas e de argumentos lógicos, é chamada teorema. A dúvida é com respeito a palavra teorema. Teoremas são proposições que podem ser comprovadas ou demonstradas. Usa-se teoremas para solução de problemas. Veja agora a dúvida, em informática existe um método computacional, conhecido por método da resolução, que coloca a seguinte afirmação: "Se não há como negar um teorema, ele é verdadeiro." O teorema não é algo já comprovado? O teorema não é evidente por si mesmo? Por que tentar negá-lo? Isto não seria um absurdo? Ou seria a cada novo teorema criado? O que não pode ser o caso. Talvez tenha um outro sentido a palavra teorema? É muito pertinente esta pergunta.

"Se não há como negar um teorema, ele é verdadeiro." Pode-se adiantar o assunto com um exemplo. O conjunto vazio é subconjunto de todo conjunto. Veja, caso não fosse, haveria um ponto pertencente ao conjunto vazio que estaria fora do conjunto. Porém, o vazio não possui elementos, logo não se pode afirmar nada contrário a respeito dele. Ou seja, é algo como dizer: "Todo político é honesto" esta afirmação é verdadeira enquanto não houver exemplos em contrários. Enquanto eu estiver procurando por algum desonesto, a afirmação continua verdadeira. Vamos a alguns detalhes. Os matemáticos avaliam a importância de um teorema (do grego théorema) de acordo com seu impacto para o resto da matemática. Em primeiro lugar, um teorema é considerado importante se contém uma verdade universal, isto é, se ele se aplica a todo um conjunto (de números, por exemplo). Em segundo lugar, os teoremas devem revelar alguma verdade profunda e subjacente a respeito de um conjunto de idéias. Ele é o trampolim para a produção de todo um novo conjunto de teoremas, ou até mesmo inspirar o desenvolvimento de um novo ramo da matemática (exemplo:a topologia que veio da análise por causa de um bendito teorema) E por último um teorema é

importante se ele resolver um problema numa área de pesquisa anteriormente obstruída pela ausência de uma conexão lógica.

No caso da computação, os teoremas representam algo um pouco mais avançado, pois a prova de um teorema equivale a provar se um argumento é falso ou verdadeiro. Caso você colocar para um computador que:"Todos os políticos são honestos", ele irá aceitar isto como verdadeiro, enquanto não surgir algum caso contrário,

Já na matemática clássica isto é diferente pois veja: ao final do século XIX a matemática assistia uma procura pelos fundamentos dela. Matemáticos como Cantor, Poincaré e Hilbert tratavam de dar sustentação a toda estrutura matemática (e olha que estrutura). Tanto que, Hilbert apresentou 10 enunciados. Enunciados estes que seriam os desafios para o século XX. Os primeiros 9 ele assistiu em vida sendo vencidos. O décimo, Hilbert não viu sendo provado. Aliás, provado ao contrário. Hilbert havia colocado que deveria haver uma forma de provar aonde a matemática nascia. Pois é, em 1931 um matemático austríaco de 25 anos mostrou que isto não é possível, pois ou a matemática sai de uma base forte. Por exemplo: a geometria de Riemann vale, pois vem da geometria de Euclides que é assumida como verdadeira. Porém quem disse que Euclides está certo. Como acreditar em: por dois pontos distintos passa uma e somente uma reta (postulado ou axioma)? INTUIÇÃO. Esta é a resposta. Certas idéias são intuitivas, e não há como prová-las.

Helmut Willy Knoblauch

ÍNDICE DE AUTORES

ÍNDICE SISTEMÁTICO