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

Parcilene Fernandes de Brito

DEDUÇÃO AUTOMÁTICA POR TABLEAUX ESTRUTURADA EM XML

Arthur Ronald de Vallauris Buchsbaum
(Orientador)

Florianópolis, Maio de 2003

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

Parcilene Fernandes de Brito

DEDUÇÃO AUTOMÁTICA POR TABLEAUX ESTRUTURADA EM XML

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

Arthur Ronald de Vallauris Buchsbaum
(Orientador)

Florianópolis, Maio de 2003

DEDUÇÃO AUTOMÁTICA POR TABLEAUX ESTRUTURADA EM XML

Parcilene Fernandes de Brito

‘Esta dissertação foi julgada adequada para a obtenção do título de Mestre em Ciências 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’.

_____

Arthur Ronald de Vallauris Buchsbaum, Dr. - Orientador

_____

Fernando Ostuni Gauthier, Dr.

Coordenador do Curso de Pós-Graduação em Ciência da Computação

Banca Examinadora:

_____

Arthur Ronald de Vallauris Buchsbaum, Dr.

_____

Raul Sidnei Wazlawick, Dr.

_____

João Bosco da Mota Alves, Dr.

“E assim, através de todas as espessas neblinas das dúvidas confusas da minha imaginação, surgem de vez em quando intuições divinas que iluminam minha perplexidade como um raio celestial e por isso dou graças a Deus, porque todos duvidam e muitos negam, porém, duvidando ou negando, são poucos os que delas têm intuições. Dúvidas de todas as coisas terrestres e intuições de algumas coisas celestiais, eis uma combinação que não faz ninguém crente ou descrente e sim uma criatura que observa tudo isso com um olhar imparcial.”

Moby Dick, Herman Melville

À minha mãe.

AGRADECIMENTOS

A Deus.

À minha mãe, que contou as histórias necessárias para o meu entendimento de que o mundo é simples e de que nós somos apenas e suficientemente o reflexo de nossas ações e de nossos pensamentos.

Ao meu pai, que me fez enxergar em seu silêncio que eu e meus irmãos somos a melhor parte de sua vida.

Aos meus irmãos, Edio, Di e Cal, responsáveis pelo meu entendimento do silêncio, da generosidade e da bondade. Que aceitaram um bebê-menina em suas vidas calmas e me ajudaram e ajudam em todos os dias do meu já tão longo breve caminho.

Aos meus sobrinhos: Isadora e sua capacidade de entender e de enxergar. Douglas e sua agilidade. Tayrine e seus sonhos. Gabriela e seu silêncio. E a Carol, a primeira e, por isso mesmo, a mais aguardada e aquela que teve que suportar mais a minha presença e minhas crises. Essas cinco pessoas me fizeram entender que devo continuar, ainda que a fadiga às vezes seja imensa e as interrogações infinitas.

Ao Quíntuplo, meus amigos de infância, meus amigos de sempre: Irenides, Ricardo, Darlene e Karylleila. Essas pessoas são parte de grande parte da minha vida, de muitos dos melhores momentos que já vivi. Mesmo que sejamos todos, por vezes, distantes e contraditórios, a amizade de tanto tempo me dá a segurança para vencer desafios e para criar novos mundos e novas histórias.

À minha madrinha, minhas tias e meus primos, que criaram o ambiente necessário para expandir minha imaginação e minha curiosidade.

Ao Pe. Alano, que através de sua bondade e de sua extrema empatia com o próximo me ajuda a enxergar Deus nas pessoas e em mim.

Ao Fabiano, por tentar me ensinar a escrever textos científicos sem romantismos e dramaticidades e perder muito do seu tempo e das suas férias estudando comigo. Por causa desses estudos, o XML está sendo usado nesse trabalho. E, principalmente, por ser meu amigo e me ajudar quando o trabalho na coordenação do curso parecia extinguir todas as minhas forças.

Ao Fernando, meu aluno, orientando e grande amigo. Que esteve comigo em todos os momentos desta dissertação. Que me ajudou a ter esperanças quando tudo parecia desmoronar e foi o apoio necessário e fundamental para que a implementação do provador pudesse existir.

À Thereza e à Cristina, que me ajudaram em todas as etapas burocráticas que envolvem a coordenação de um curso e estiveram por perto todas as vezes que eu cansei e mesmo nas vezes que eu fingia não estar cansada.

Ao prof. Hugo, pela extrema confiança e pelo apoio em todos esses seis anos de trabalho em conjunto. À D. Carmem, por sempre ter as palavras certas para me ajudar a suportar momentos difíceis e por partilhar momentos alegres. À Paula, pelo carinho, atenção e cuidado, ainda que eu seja grossa o suficiente para não demonstrar que o carinho, a atenção e o cuidado são recíprocos. À professora Delzimar, pelas conversas sobre lógica, sobre Chomsky, sobre literatura e sobre a vida e pelo extremo carinho. Ao Pedro, por me lembrar que há vida na terra. À Régia, pela paciência e pelo cafezinho.

Aos meus queridos amigos, Jackson e Ricardo Marx, por escreverem as palavras certas em momentos difíceis, e, como alguém já me disse uma vez, por me ajudarem a abrir os olhos nos momentos em que eles pareciam querer estreitar o caminho das possibilidades.

Aos professores e colegas de trabalho, Lilissanne, Mário Sérgio, Mádia, Andrés, Piveta e Deise, por fazerem do curso um lugar bom de se ficar por tantas e infinitas horas.

Aos meus alunos e amigos, Wagner Tigrão (pelo bom papo entre Paraíso e Palmas, e por alguém se manter acordado enquanto eu dirigia), branquinhos (Edeilson, Lucas e Clodomir, que possuem um plano secreto para dominar o mundo), Danilo, Leandro, Paulo, Carlos, Jorge, Ary, Anderson, Álvaro, D'Ilton e Polly (que sempre estão por perto); Michael (o aluno que fez da Informática e Sociedade uma disciplina sempre importante na minha vida) e todos os outros que freqüentam os temáticos e nos trazem alegrias e surpresas.

Ao prof. Arthur, meu orientador, pela paciência e apoio. Ao prof. Raul, pela confiança e pelas ótimas aulas. Aos professores Beth e Bosco pelo companheirismo e alegria. Aos meus professores de física, Saulo e Carla, que me ajudaram a entender o universo infinito de possibilidades que carregamos conosco.

Ao Mulder e à Scully, pelos nove anos que passamos juntos à procura da verdade que está lá fora.

Ao Saramago, Machado de Assis, Fernando Pessoa, Drummond, Nietzsche, Umberto Eco, Gabriel Garcia Márquez, Emily Brontë e Hermman Melville, alguns dos muitos escritores que através de suas palavras são, em grande parte (para o bem ou para o mal), responsáveis pela pessoa que sou hoje.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho tem como objetivo descrever o desenvolvimento de um provador de teoremas segundo o Método dos Tableaux. Para isso, foram propostos dois refinamentos com o intuito de tornar o método mais eficiente durante as etapas de prova. Esses refinamentos tratam da ordenação das fórmulas e da instanciação das variáveis, buscando diminuir a inserção de elementos na árvore, propiciando, assim, uma maior eficiência em relação ao número de nós. A linguagem Java foi utilizada para a implementação do provador, possibilitando a manipulação das fórmulas estruturadas como documentos XML. Ao final do trabalho, é verificada a eficiência do provador através da comparação com outros provadores em relação à sistematização das suas etapas de prova e dos resultados obtidos.

ABSTRACT

This thesis main goal is to describe the development of a theorem prover according to the Tableau Method. To achieve this, two refinements are proposed to make the method more efficient concerning to the proof steps. These refinements are related to the formulas sorting and to variable instantiation, aiming to decrease the number of elements inserted into the tree and to increase the efficiency regarding to the number of nodes. The Java language was used to implement the prover, allowing the manipulation of formulas specified as XML documents. In the end of this work, the theorem prover efficiency is verified against others provers, comparing the systematization of proof steps and the results achieved.

Capítulo 1

Introdução

Provadores automáticos de teoremas são programas que utilizam métodos de resolução para a verificação da validade ou invalidade de fórmulas lógicas. Segundo BOYER (1995), provador de teorema é um programa de manipulação de símbolos que tenta provar uma determinada fórmula aplicando as regras de inferência da lógica. Acrescenta, ainda, que o comportamento do provador de teorema é determinado em grande parte pela base de dados e pelas centenas do heurísticas de controle de uso das regras de inferência, dos axiomas, das definições e de teoremas previamente provados.

Para a implementação de provadores de teoremas é necessário utilizar um método de prova (ex.: dedução natural, cálculo de seqüentes, método da resolução). O princípio básico da resolução foi proposto por ROBINSON (1965) e é considerado um método de prova por refutação. A resolução é realizada a partir da prova de que a negação de um teorema implica em uma contradição. Ou pela constatação de que uma dada fórmula não é verdadeira, ou seja, não ocorreu a contradição em todas as etapas de prova.

O problema fundamental da automatização do raciocínio possui como dados uma coleção T finita de fórmulas e uma fórmula adicional P, e como proposição o enunciado “P é conseqüência de T”. Alonzo Church mostrou que este problema é indecidível, ou seja, não existe nenhum algoritmo que sempre dê uma resposta positiva quando T |- P, e dê uma resposta negativa quando T $\nvdash$ P(FENDT, 2000). No entanto, existem algoritmos de prova automática como os métodos de resolução e o método dos tableaux que fornecem uma solução semidecidível para esta questão e, ainda, fornecem respostas que informam que algumas fórmulas não são conseqüência lógica das premissas.

O desenvolvimento de provadores de teoremas não é um tema necessariamente novo, mas as técnicas utilizadas para desenvolvê-los e os resultados obtidos a partir de modificação em algoritmos de automatização podem trazer soluções interessantes e mais eficientes, ou, pelo menos a comprovação da ineficiência de tais algoritmos. O trabalho aqui desenvolvido tem como objetivo a construção de um provador de teoremas para a lógica quantificacional. Para isso, buscou-se, num primeiro momento, obter uma visão geral da lógica, mais especificamente da parte da lógica que trata de sistemas de prova de teoremas.

O método dos tableaux foi utilizado como a base do provador, assim as regras de tratamento dos conectivos e quantificadores foram estabelecidas a partir das regras gerais dos tableaux. Foram realizados refinamentos no método para que o provador alcançasse uma maior eficiência. Esta eficiência é entendida, neste trabalho, como a realização da prova de fórmulas quantificacionais utilizando uma quantidade mínima de nós. Para estabelecer se os resultados alcançaram eficiência, foram realizadas comparações com os resultados obtidos em outros provadores.

A partir do estudo da lógica foi sendo delineado o trabalho e suas características principais. Em primeiro lugar, decidiu-se fazer um provador enfatizando, especialmente, o uso de tecnologias distintas das usuais para esse tipo de trabalho. Então, a utilização de Java e XML (*Extensible Markup Language*) na implementação serviu como uma das formas de diferenciar este trabalho dos provadores desenvolvidos por outros pesquisadores. A idéia básica foi o desenvolvimento de métodos a partir de um padrão que permitisse a criação de uma estrutura comum a fórmulas quantificacionais, e que tal estrutura já fizesse a validação das mesmas. Desta forma, definido o XML como o padrão que permitia tal flexibilidade na criação de uma estrutura para fórmulas quantificacionais, buscou-se uma linguagem que pudesse trabalhar de forma eficiente os documentos formados em XML. Assim, definiu-se a linguagem Java para ser utilizada na implementação do provador. Isto porque a mesma tem uma API (*Applications Programming Interface*) que permite a geração de uma árvore na memória a partir de um documento XML, e esta API possui métodos que permitem a manipulação de tal árvore.

Definido o tema (provador de teoremas), o método (tableaux), a linguagem (Java) e o padrão para a estruturação das fórmulas (XML), foi desenvolvido o trabalho, tendo como principal objetivo a definição de refinamentos que pudessem tratar os conectivos lógicos, as instanciações das variáveis quantificadas e o estabelecimento de regras que permitissem a inferência da validade e invalidade das fórmulas. Assim, a partir da criação da estrutura em XML para a linguagem quantificacional, dos refinamentos realizados no método dos tableaux para esta linguagem e da implementação da árvore de prova, foi possível verificar, a partir de testes comparativos com outros provadores, que as linguagens utilizadas e o algoritmo para a verificação da validade ou invalidade das fórmulas quantificadas estavam coerentes e apresentavam resultados eficientes

No próximo capítulo será apresentada uma revisão sucinta de Lógica, enfocando a lógica quantificacional. São apresentados, também, os elementos básicos que compõem o método dos tableaux. Em seguida, uma explanação sobre a linguagem XML (*Extensible Markup Language*) e a API DOM (*Document Object Model*), cujos conceitos são necessários para o entendimento da implementação realizada. No capítulo 3 é apresentada a utilização do padrão XML para o desenvolvimento da estrutura das fórmulas. No capítulo 4 são descritos os refinamentos realizados no Método dos Tableaux e, no capítulo 5, as etapas de implementação. A monografia encerra-se com as considerações finais apresentadas no capítulo 6, as referências (cap. 7) e os anexos (cap. 8) utilizados no trabalho.

Capítulo 2

Revisão de Literatura

2.1 Lógica

A definição de Lógica não é um trabalho trivial, pois sua complexidade e abrangência dão margem a algumas interpretações e várias definições. Segundo da Costa (1980), Lógica é a disciplina que tem como um de seus objetivos básicos o estudo das inferências legítimas, ou seja, das inferências tais que, sempre que as premissas forem verdadeiras, a conclusão também o é. Noutras palavras, uma das principais finalidades da lógica é a investigação da relação de conseqüência lógica (ou de conseqüência semântica). Para Bessie e Glennan (2000), a lógica pode ser definida como o estudo dos métodos para determinar se uma conclusão pode ou não ser corretamente inferida a partir de um conjunto de suposições. Já em J.Dopp ((J.DOPP apud THIRY,1996), a Lógica Formal é apresentada como uma ciência que determina as formas corretas (ou válidas) de raciocínio. É definida, também, como o estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto (Copi, 1978). E, como o estudo da legitimidade de inferências - ou argumentos (Hegenberg, 1995).

Um ponto em comum entre estas definições é a afirmação de que a Lógica trabalha com inferências que visam atingir um objetivo. E quando são citadas formas válidas ou corretas de raciocínio não há relação com a verdade no âmbito real, mas sim da possibilidade de uma dada construção ser verdadeira a partir da análise da sua formação, conforme é observado nas definições abaixo:

> A Lógica é (...) o exame dessa parte do raciocínio que depende da maneira como as inferências são formadas (...). Nesse sentido, não tem nada a ver com a verdade de fatos, opiniões ou pressupostos da qual uma inferência é derivada. (DE MORGAN, in Formal Logic, cap. 1 apud THIRY, 1996)

> A Lógica Formal investiga as estruturas das proposições e do raciocínio dedutivo, ignorando o conteúdo das proposições que venham a ser consideradas, para concentrar-se apenas em sua forma (Hegenberg, 1995).

> Lógica Formal é o estudo de formas de argumentos, regras abstratas de raciocínio separadas em vários argumentos diferentes (Nolt, 1988)

Segundo THIRY (1996), o raciocínio ou inferência é um desenvolvimento do conhecimento através de meios lógicos a partir de elementos conhecidos ou admitidos, denominados premissas ou antecedentes. Esse raciocínio se apóia em alguns princípios fundamentais da lógica clássica:

- O princípio de não-contradição

É impossível afirmar e negar simultaneamente uma mesma fórmula para um mesmo sujeito. E, mais simplesmente: A $\neq$ não A. Este princípio é o ponto de partida da reflexão lógica que distingue o verdadeiro do falso, o erro da verdade.

- O princípio de identidade

Aquilo que é, é; o que não é, não é. Mais simplesmente: A = A. Este princípio não é uma trivialidade. É preciso ter a certeza da estabilidade dos conceitos para se poder defender um raciocínio.

- O princípio do terceiro-excluído

Toda coisa é ou não é. Este princípio próprio da lógica binária clássica fixa a impossibilidade de um juízo ter outro valor de verdade que o verdadeiro ou o falso (exclusivamente).

Estes princípios são os pilares da Lógica Clássica, no qual este trabalho está inserido. Sendo assim, a formalização e a validação das fórmulas trabalhadas levarão em consideração a identidade dos termos, a percepção da refutação de

duas fórmulas contraditórias e a inexistência de um terceiro conceito, pois serão consideradas apenas a verdade e falsidade das fórmulas.

A Lógica Clássica compreende geralmente duas partes: a Lógica Proposicional e a Lógica Quantificacional. A primeira toma como unidade de base a proposição que expressa um acontecimento ou um fato. As unidades de base ou proposições atômicas podem ser combinadas para formar proposições moleculares. A lógica proposicional estuda todas as associações possíveis: é uma lógica inter-proposicional. A segunda toma como unidade de base os termos no interior da proposição, expressando então uma relação entre objetos ou conjuntos de objetos. É a lógica quantificacional ou lógica intraproposicional. (THIRY, 1996)

Ambas trabalham com o conceito de argumento. Segundo BESSIE e GLENNAN (2000), um argumento é uma coleção de enunciados, sendo que um destes enunciados é a conclusão e os demais são as premissas. É dedutivamente válido se, e somente se, é impossível que ele tenha todas as premissas verdadeiras e a conclusão falsa.

Nesse trabalho, serão apresentadas a Lógica Proposicional e a Lógica Quantificacional distintamente, para uma melhor visualização do aumento da complexidade de um provador que irá trabalhar com fórmulas quantificacionais.

2.1.1 Lógica Proposicional

A Lógica Proposicional trabalha com letras sentenciais e conectivos lógicos, formando, assim, uma linguagem com termos bem definidos, mas infinitamente mais limitada do que as linguagens ditas linguagens naturais, como por exemplo, português e inglês. Essa linguagem é constituída por:

- Letras sentenciais: em JOHNSON (1996), uma sentença é definida como uma fórmula fechada, isto é, uma fórmula que não contém variáveis livres. Como a lógica proposicional não contém variáveis, toda fórmula proposicional é uma sentença.

- Conectivos lógicos: sinais para a negação (não), a conjunção (e), a disjunção (ou), o condicional (se... então) e o bicondicional (se e somente se). Os símbolos utilizados neste trabalho são apresentados na tabela 1.
- Parênteses.

Tabela 1: Símbolos dos conectivos lógicos

Conectivo Lógico	**Símbolo**
Negação – Não	~
Conjunção – e	$\wedge$
Disjunção – ou	$\vee$
Condicional – se... então	$\rightarrow$
Bicondicional – se e somente se	$\leftrightarrow$

Além de ter que seguir uma linguagem bem definida, uma fórmula, para ser analisada como válida ou inválida, necessita, inicialmente, ser bem formada. Na Lógica Proposicional é definido que:

- Qualquer letra sentencial é uma Fórmula Bem Formada (FBF);
- Dado uma fórmula qualquer, a sua negação também é um FBF.
- Se P e Q são FBFs, então P # Q (podendo # ser entendido como qualquer um dos conectivos lógicos binários citados na tabela 1) será uma FBF.

Os símbolos descritos acima correspondem ao alfabeto da Lógica Proposicional. Já a semântica é dada por uma função I, denominada função de interpretação. Esta função permite a associação de um valor veritativo "*verdadeiro*" ou "*falso*" para cada fórmula da Lógica Proposicional. Segundo SOUZA (2002), uma interpretação I, na Lógica Proposicional, é uma função binária tal que o domínio de I é constituído pelo conjunto das fórmulas da Lógica Proposicional; o contradomínio de I é o conjunto {V, F}; o valor da interpretação de I, tendo como argumentos os símbolos veritativos, é dado por I[*verdadeiro*] = Ve I[*false*] = F. Dado um símbolo proposicional P, então I[P] $\in$ {V, F}.

A interpretação de expressões complexas depende da interpretação de cada uma das suas instâncias. No caso da Lógica Proposicional, podem ser usadas *tabelas veritativas* para a verificação da validade ou invalidade de um argumento, conforme pode ser observado na definição abaixo.

Definição: uma forma de argumento é válida (através da utilização de *tabelas veritativas*) se, e somente se, todas as suas instâncias são válidas. Uma instância de um argumento é válida se é impossível que o valor veritativo da conclusão seja **falso** enquanto todos os valores veritativos das premissas são **verdadeiros** (Figura 1) .

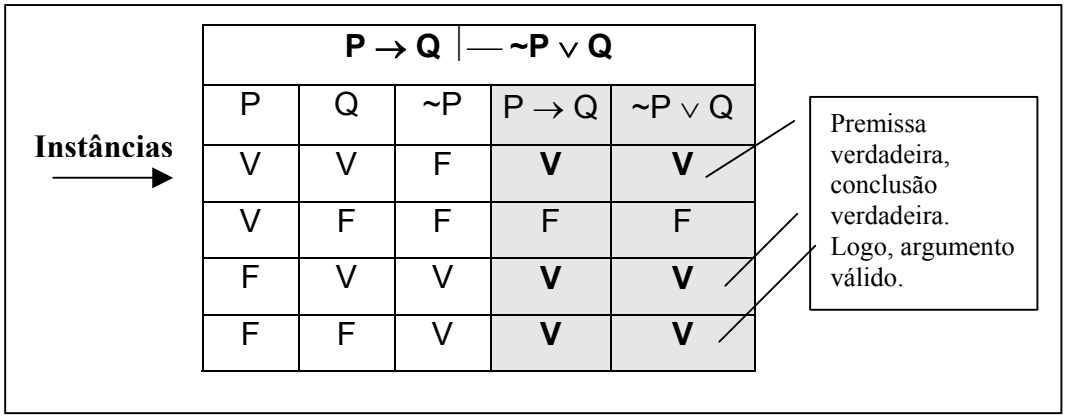

Figura 1: Verificação da validade de um argumento utilizando tabela veritativa

É possível, também, construir as fórmulas da Lógica Proposicional com apenas três conectivos lógicos: a negação, a conjunção e a disjunção. Isto porque o condicional é equivalente à disjunção de seus elementos, com o antecedente negado e o conseqüente permanecendo da mesma forma. O bicondicional, por sua vez, é equivalente à disjunção de duas conjunções, sendo que o primeiro elemento da disjunção é a conjunção dos elementos do bicondicional na sua forma original e o segundo elemento da disjunção é a conjunção dos mesmos elementos, mas na sua forma negada. Entretanto, apesar desta possibilidade de economia de conectivos, serão trabalhadas no provador as fórmulas compostas pelos cinco conectivos lógicos. Isso se dá, especialmente, pelo caráter elucidativo que é proposto na apresentação dos passos gerados pelo provador, ou seja, pretende-se que o usuário possa verificar o que acontece logicamente com cada conectivo nas etapas de prova.

2.1.2 Lógica Quantificacional

A Lógica Quantificacional permite a manipulação de uma forma de raciocínio que se aproxima mais da linguagem natural. E, para isso, acrescenta outros elementos à linguagem da Lógica Proposicional.

A linguagem da Lógica Quantificacional é dividida em duas partes: os símbolos lógicos (cuja interpretação permanece fixa em todos os contextos) e os símbolos não-lógicos (cuja interpretação varia de problema para problema) (NOLT, 1991).

Símbolos Lógicos: conectivos lógicos: '~', '$\wedge$', '$\vee$', '$\rightarrow$', '$\leftrightarrow$'; quantificadores: '$\forall$', '$\exists$.

Símbolos Não-Lógicos: letras nominais ou constantes: letras minúsculas de 'a' a 't' e suas derivações 'a_1, a_2, ...' a 't_1, t_2, ...'; variáveis: letras minúsculas de 'u' a 'z' e suas derivações 'u_1, u_2, ...' a 'z_1, z_2, ...'; sinais predicativos: letras maiúsculas.

Seguem, abaixo, algumas definições de grande importância para esse trabalho retiradas de JOHNSON (1996).

Definição 1 (Termos): Dado o conjunto de variáveis V e o conjunto de constantes N, o conjunto de termos T = V $\cup$ N.

Definição 2 (Literais): Literais podem ser denotados como P (t_1 ... t_n) ou ~P(t_1 ... t_n) onde P(t_1 ... t_n) são as n possibilidades de relação do predicado (P) com um dado termo, e cada t_i , para $0 < i \leq n$, é um termo.

Definição 3 (Fórmula): Uma fórmula é definida como:

- Um literal;
- Uma fórmula do tipo: P $\wedge$ Q, P $\vee$ Q, P $\rightarrow$ Q, P $\leftrightarrow$ Q, ~ P, (P), [P], {P}; onde P e Q são fórmulas;

- Uma fórmula de um dos tipos: $\forall x\, Px$ ou $\exists x\, Px$; onde P é uma fórmula e x é uma variável.

Definição 4 (Variáveis livres) (Fendt, 2000):

- x é livre em r (t1 ... tn) se x ocorre em um dos termos t1, ..., tn;
- x é livre em ~ P se x é livre em P;
- x é livre em $(P \wedge Q)$, $(P \vee Q)$, $(P \rightarrow Q)$, $(P \leftrightarrow Q)$ 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.

Nem toda construção a partir da utilização destes símbolos pode resultar numa fórmula correta da Lógica Quantificacional, daí a importância do conceito de Fórmula Bem Formada (FBF). O conceito de FBF da Lógica Quantificacional é definido pelas seguintes regras de formação (NOLT, 1991):

1. Toda fórmula atômica é uma FBF;
2. Se ϕ é uma FBF, então sua negação também o é;
3. Se ϕ e ψ são FBFs, então $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \rightarrow \psi)$, $(\phi \leftrightarrow \psi)$ são FBFs;
4. Se ϕ é uma FBF contendo uma constante α, então qualquer fórmula da forma $\forall_\beta \phi^\beta/_\alpha$ ou $\exists_\beta \phi^\beta/_\alpha$ é uma FBF, onde $\phi^\beta/_\alpha$ é o resultado de se substituir uma ou mais ocorrências de α em ϕ por uma variável β que ocorre em ϕ.

O conceito de interpretação em fórmulas quantificacionais é bem mais amplo que a interpretação na lógica proposicional, tendo em vista que a interpretação deve ser definida em todos os símbolos do domínio especificado. Assim, torna-se fundamental, para a interpretação de uma fórmula, a interpretação dos sinais predicativos, dos nomes e dos demais sinais que a compõem. Uma determinada estrutura T consiste em um conjunto de elementos S, chamados de *domínio*, juntamente com uma função de interpretação I, em que estão os nomes, os sinais predicativos e os demais sinais dos elementos de S.

Para a verificação da validade ou invalidade de uma fórmula proposicional ou quantificacional pode ser utilizado um método de prova. Nesse trabalho será utilizado o método dos tableaux, apresentado na próxima seção.

2.2 O Método dos Tableaux

O método dos tableaux é um método de prova indireta que teve como percussores Beth e, independentemente, Hintikka (REEVES, 1983). É um método de prova por refutação, no qual um teorema é provado pelo insucesso na tentativa de construção sistemática de um modelo para a sua negação. O método tem fundamento semântico: ao tentar provar que α é um teorema lógico, o que o método, em sua concepção original, faz de fato é verificar a impossibilidade da insatisfação de $\sim\alpha$ (BUCHSBAUM e PEQUENO, 1990). Segundo SMULLYAN (1995), oito fatos formam os fundamentos básicos para a construção deste método:

1) a) Se ~X é verdadeiro, então X é falso.
   b) Se ~X é falso, então X é verdadeiro.
2) a) Se a conjunção $X \wedge Y$ é verdadeira, então X, Y são ambos verdadeiros.
   b) Se a conjunção $X \wedge Y$ é falsa, então X é falso, ou Y é falso ou ambos são falsos.
3) Se a disjunção $X \vee Y$ é verdadeira, então X é verdadeiro, ou Y é verdadeiro, ou ambos são verdadeiros.
   b) Se a disjunção $X \vee Y$ for falsa, então X e Y são ambos falsos.
4) a) Se $X \rightarrow Y$ é verdadeiro, então X é falso ou Y é verdadeiro.
   b) Se $X \rightarrow Y$ é falso, então X é verdadeiro e Y é falso.

O método dos tableaux segue o formato de uma árvore em que, dada uma coleção de fórmulas T e uma conclusão S, busca-se através da negação da sua conclusão verificar a validade ou invalidade da fórmula. Segundo REEVES (1983), as sentenças em $T \cup \{\sim S\}$ são chamadas sentenças iniciais do Tableau. E, $T \vdash S$ é a representação das sentenças iniciais $T \cup \{\sim S\}$.

Uma fórmula é considerada válida se é possível refutar a negação da sua conclusão, ou seja, se todos os ramos da supracitada árvore são fechados. Um ramo fechado é aquele que possui uma contradição (ex.: P, ~P). Segundo BUCHSBAUM e PEQUENO (1990), além da utilidade do método dos tableaux na prova de teoremas, a forma exaustiva e sistemática com que o espaço de construção de modelos é percorrido faz com que ele seja um valioso instrumento na determinação de resultados negativos em cálculos decidíveis (ou em fragmentos decidíveis de cálculos indecidíveis). Dessa forma, a falha na tentativa de fechar um ramo pode ser efetivamente decidida após um número finito de passos. O não fechamento de um ramo indica a impossibilidade de refutação, permitindo a inferência de que uma dada fórmula não é um teorema do cálculo.

Na figura 2 é apresentada a verificação da validade de uma fórmula proposicional utilizando o método dos tableaux. Neste exemplo, é possível verificar que a fórmula é válida, pois todos os ramos da árvore foram fechados. Desta forma, refuta-se a conclusão negada e infere-se como verdadeira a conclusão no seu modo normal, dada a coleção de fórmulas T que foram estabelecidas como premissas.

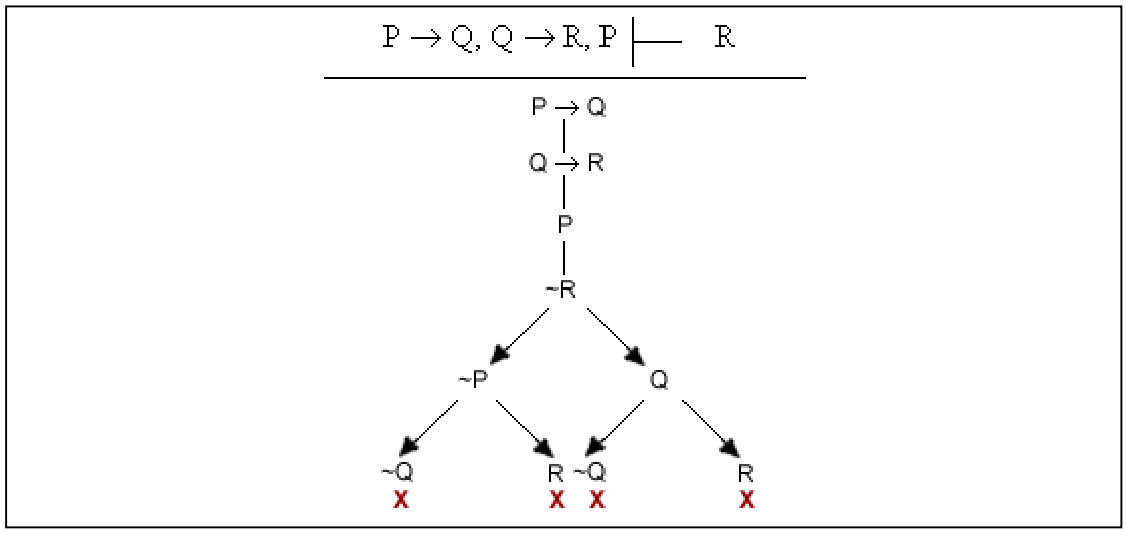

Figura 2: Verificação da validade de uma fórmula utilizando o método dos tableaux

Para um melhor entendimento do método de Tableaux, é necessário entender alguns termos. As definições abaixo são baseadas em REEVES e CLARK (apud JOHNSON, 1996):

- Definição (Tableau): Um tableau é representado por uma árvore binária, T = <s, V, L>. V é um conjunto de vértices, s é uma função sucessor e L é o mapeamento dos vértices. Dado algum vértice $v \in V$, $s(v) \subseteq V$ e s(v) é o conjunto de vértices imediatamente "abaixo" de v na árvore. Cada vértice é mapeado para uma fórmula de L.

- Definição (Raiz): Dado um tableau T = <s, V, L>, v_r é definido como a raiz. Apesar de $v_r \in V$, ele não ocorre no conjunto de sucessores de algum $v \in V$, então é correto afirmar que $v_r \notin s(v)$.

- Definição (Folha): Uma folha v_l de um tableau T = <s, V, L> é definido por $s(v_l) = \varnothing$. Portanto, v_l não tem sucessores.

- Definição (Caminho): Um caminho é uma seqüência finita de vértices $<v_o, ..., v_n>$ onde v_o é o vértice raiz e $s(v_n) = 0$.
- Assim, o número de caminhos (nesse trabalho também chamado de ramos) será equivalente ao número de folhas do tableau.

- Definição (Contradição): Duas fórmulas F e G são contraditórias se $F = \neg G$ ou $\neg G = F$. A identificação da contradição é a base para sistemas de prova por refutação.

Desta forma, é possível verificar que um sistema de tableaux é um método de prova por refutação, que segue o formato de uma árvore. O nó inicial dessa árvore consiste, muitas vezes, na negação do teorema a ser provado. E a descendência desses nós é advinda da utilização de alguma regra referente a um dos conectivos lógicos ou dos quantificadores. Para a verificação da validade de uma fórmula, verifica-se, a partir do nó raiz, se em cada ramo da árvore há a ocorrência de uma contradição. Se o tableaux for fechado isso indica que todos os caminhos expandidos a partir do nó raiz estavam inconsistentes, ou seja, apresentaram uma contradição.

Para (BUCHSBAUM e PEQUENO, 1990), um sistema de tableaux é uma quíntupla ordenada S = < L, L', I, F, R>, em L representa a linguagem inicial, L' - a derivação que pode existir da linguagem L, ou seja, a linguagem de trabalho, I - a função de inicialização de S, F – o fator de fechamento dos ramos da árvore e R – a coleção de regras necessárias para a existência eficiente do método.

Algumas definições genéricas sobre situações que podem existir durante a execução do método são apresentadas a seguir:

1. Dado S = < L, L', I, F, R>, uma fórmula P é excluída em S se não há regra R em S aplicável à fórmula P presente em L'.

2. Dado S = < L, L', I, F, R>, se ϕ é um ramo aberto em L' e F(ϕ) = fechado, ϕ é dito um ramo fechado em S. Um ramo fechado é aquele que possui duas fórmulas contraditórias.

3. Dado S = < L, L', I, F, R>, um ramo em L' é dito exaurido em S se todos os nós que possuem fórmulas não excluídas estejam marcados. Assim, é identificado que todas as fórmulas foram trabalhadas, ainda que possa não ter sido possível encontrar uma conclusão satisfatória.

4. Definição: É dito que uma seqüência T de desenvolvimento de tableaux em S é completa se forem satisfeitas as condições abaixo:
   - Se I é finita, então $(T_i)_{i \in I}$ termina com uma refutação de todos os ramos, ou com algum ramo exaurido aberto;
   - Se I é infinita, então $(T_i)_{i \in I}$ tende para uma árvore que possui um ramo infinito exaurido aberto.

Para um entendimento de como é utilizado o método para a verificação da validade das fórmulas proposicionais, é apresentada, na figura 3, a estrutura que cada tipo de fórmula formada por algum conectivo lógico deve possuir na árvore.

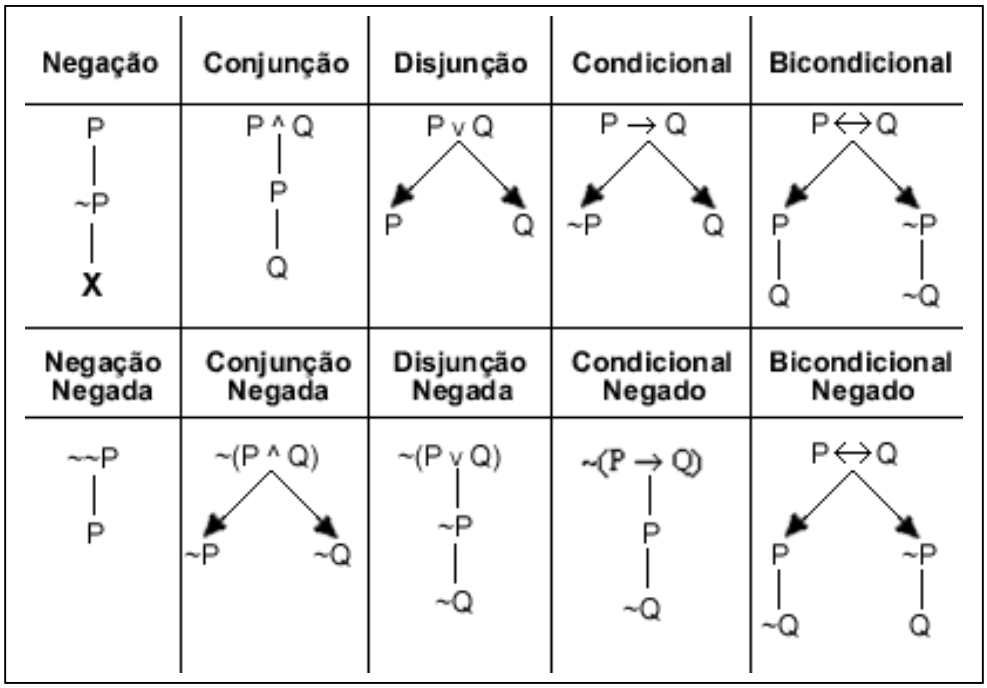

Figura 3: A estrutura dos conectivos lógicos segundo o método dos tableaux

Basicamente, as bifurcações e os caminhos seqüenciais são estabelecidos pela relação T ∪ {~S}, ou seja, pela coleção de fórmulas e pela negação da conclusão.

Para a utilização do método dos tableaux para as fórmulas quantificacionais é acrescentada a formalização das fórmulas universais e existenciais. E, tendo em vista a indecidibilidade da Lógica Quantificacional, nem toda fórmula quantificacional pode ser provada num tempo passível de ser analisado. O tratamento destinado a este ponto na implementação do provador é primordial para a eficiência do programa. Na figura 4 é apresentada a forma de utilização dos quantificadores para o tableau.

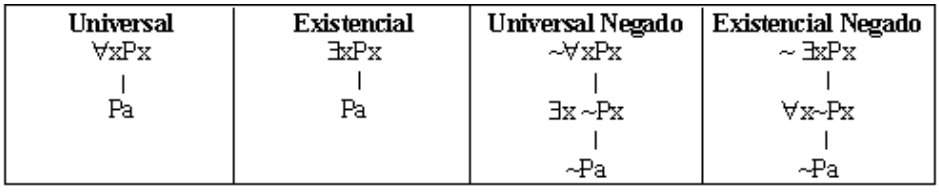

Figura 4: Tratamento dado aos quantificadores universal e existencial no tableau

A diferença básica no tratamento dos quantificadores universal e existencial está na instanciação da variável quando é realizada a “eliminação” desses quantificadores. Por exemplo, quando é criada uma instância de um conjunto que tenha uma determinada propriedade P, essa propriedade passa a pertencer também a um indivíduo específico.

Quando a variável está quantificada universalmente esta instanciação pode ser feita sempre com o mesmo indivíduo. Assim, se foi adotado um indivíduo “a” em uma determinada substituição de variável por uma constante e esta adoção é necessária para fechar um ramo da árvore, então outras substituições para as variáveis quantificadas universalmente podem ser realizadas utilizando o mesmo indivíduo “a” (Figura 5).

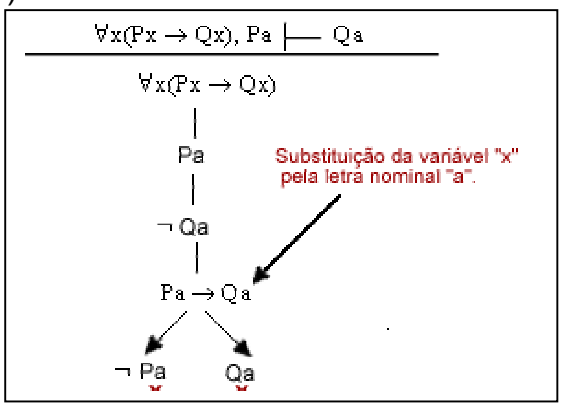

Figura 5: Instanciação universal

No entanto, se a variável está quantificada existencialmente, não pode ser feita uma instanciação com uma constante que esteja sendo utilizada no ramo em que está sendo realizada a substituição. Ou seja, se P for uma propriedade para um indivíduo “a” no ramo vigente, a instanciação de um outro elemento da árvore deve ser realizada considerando um outro indivíduo (Figura 6).

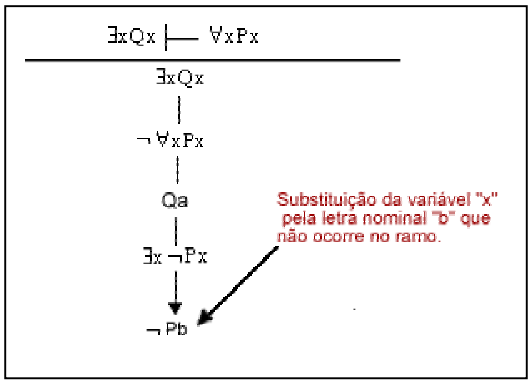

Figura 6: Instanciação existencial

2.3 Extensible Markup Language e Document Object Model

O padrão XML (*eXtensible Markup Language)* descreve uma linguagem de marcação que simplifica a linguagem SGML (*Standard Generalized Markup Language*), na qual é baseada (W3C, 1998a). De forma geral, SGML é uma linguagem para descrição da estrutura lógica de um documento utilizando-se de marcações (HERWIJNEN, 1994). As linguagens de marcação determinam a forma em que o documento está estruturado e, de acordo com essa estrutura, determinam como ele será apresentado. Regras explícitas determinam onde estas estruturas começam e terminam através de marcas (*tags*) que são colocadas antes e depois do conteúdo associado (KIRK e PITTS-MOULTIS, 2000) (MCGRATH, 1999).

Para a manipulação de documentos XML é necessária a utilização de um analisador sintático; nesse trabalho é utilizado o DOM (*Document Object Model*), da W3C. Segundo ANDERSON (2001), o DOM pode ser definido como uma plataforma – e linguagem – de interface neutra que permite a manipulação de documentos estruturados em árvore.

2.3.1 Extensible Markup Language

Os elementos que constituem um documento XML formam uma árvore hierárquica simples. Sendo assim, possui um único nó raiz, preenchido com o elemento que inicia o documento (Figura 7).

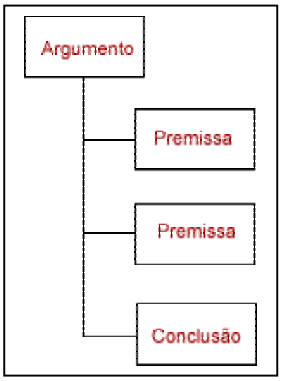

Figura 7: Exemplo de uma estrutura em árvore de um documento XML

O XML apresenta um conjunto de características que o diferencia do HTML e do SGML. Essas características são apresentadas a seguir.

- Com relação ao HTML:
    - extensibilidade: permite que o desenvolvedor crie novas marcas e atributos;
    - validação: possibilita a verificação da estrutura do documento quanto a sua validade, no momento do processamento;
    - foco na informação: aplicativos criados com XML focam a informação e não a apresentação, como ocorre com os desenvolvidos em HTML. O objetivo do XML é permitir que tags sejam criadas e que possibilitem a identificação da informação.
- Com relação ao SGML:
    - o XML foi criado para ser usado, principalmente, na Internet;
    - os documentos XML são mais rápidos de serem criados, visto que incluem uma gama menor de recursos do que o SGML.

Os documentos descritos em XML baseiam-se em três componentes distintos: estrutura, conteúdo e apresentação, e o fato destes três componentes serem independentes constitui uma das vantagens deste padrão. A seguir essa divisão será detalhada.

- *Document Type Definition* (DTD): possui a função de estruturar o documento, definindo que tipos de elementos e atributos existirão e como as instâncias desses elementos estão hierarquicamente relacionadas – a estrutura.
- *eXtensible Markup Language* (XML): documento que pode estar ou não relacionado com o DTD, porém, caso esteja, deverá seguir as regras definidas pelo mesmo. Esse documento é o responsável por armazenar as informações propriamente ditas – o conteúdo.
- *eXtensible Style Language* (XSL): enquanto o DTD define as regras de armazenamento, o XSL define as regras de apresentação do documento. O papel de um documento XSL é similar ao HTML, porém com a vantagem de, nele, conter apenas as regras da apresentação e não o conteúdo.

Outra vantagem é que o XSL pode se relacionar com diversos documentos XML, desde que todos sejam estruturados com o mesmo DTD, aplicando a mesma apresentação a todos eles.

No trabalho aqui desenvolvido foi necessária a definição da estrutura e do conteúdo, pois buscou-se a partir disso validar o algoritmo proposto a partir do método dos tableaux. Não há, portanto, nesse trabalho, uma preocupação com a interface de apresentação do resultado e, sim, que o resultado reflita corretamente os passos seguidos para a prova de um dado problema. Para a criação da estrutura de armazenamento das fórmulas quantificacionais é necessária uma abordagem mais pormenorizada do DTD. Assim será possível o entendimento de alguns termos utilizados para a construção do documento que foi usado para validar e construir o documento XML proposto nesse trabalho.

A estrutura é definida através de um DTD em que estão contidas as informações e elementos responsáveis pela organização do documento [MCGRATH, 1999], ou seja, é na definição do tipo do documento que é estruturada a sintaxe da linguagem que está sendo desenvolvida. Desta forma, o DTD fornece um conjunto de regras para o documento XML associado a ele. Estas regras vão desde a definição do tipo dos elementos, sua multiplicidade, até a definição de possíveis atributos. O conteúdo é o valor atribuído aos elementos no documento XML.

Para compor um documento XML é necessário utilizar os termos especificados no DTD. Quatro tipos de declarações de marcação são usados na construção de um DTD (ANDERSON et al, 2001):

- *ELEMENT*: Declaração de um tipo de elemento XML;
- *ATTLIST*: Declaração dos atributos que podem ser designados a um tipo de elemento específico e os valores permissíveis destes atributos.
- *ENTITY*: Declaração de conteúdo reutilizável;
- *NOTATION*: Declarações de formato para conteúdo externo que não deve ser analisado em termos de sintaxe (dados binários, por exemplo), e a aplicação externa que lida com o conteúdo.

Para a definição da estrutura que comporta as fórmulas quantificacionais foi necessário definir os elementos e seus atributos. Desta forma, os dois últimos tipos não foram considerados neste trabalho.

Segundo ANDERSON (2001), a estrutura dos elementos é declarada nos modelos de conteúdo. Esses modelos consistem em um conjunto de parênteses envolvendo algumas combinações de nomes de elemento filho, operadores e palavra-chave (#PCDATA). Os operadores são usados para indicar cardinalidade e indicar como os elementos e dados de caracteres podem ser combinados (Tabela 2).

Tabela 2: Operadores para combinar elementos e operadores de cardinalidade

Operadores de Ordem	**Significado**
,	(vírgula) seqüência estrita
|	(barra) opção
Operadores de Cardinalidade	Significado
?	Opcional
*	Zero ou mais
+	Um ou mais

2.3.2 Document Object Model

Document Object Model (DOM) é uma plataforma (e linguagem) neutra (independente do sistema operacional usado) para manipulação dinâmica do conteúdo, estrutura e estilo de um documento (W3C, 1998). Essa plataforma foi desenvolvida pela W3C com objetivo de ser uma interface padrão, independente de plataforma e da linguagem usada, para o tratamento de documentos XML (W3C, 1998). Segundo ANDERSON (2001), O DOM pode ser definido, também, como uma API (Applications Programming Interface) que define uma funcionalidade padrão para navegação e manipulação do conteúdo e da estrutura em documentos XML. O DOM transforma o arquivo texto (documento XML) em uma estrutura em forma de árvore. Assim, cada elemento é considerado como um nó da árvore gerada.

O DOM caracteriza-se por montar e manter a árvore do documento XML em memória (ANDERSON, 2001). Essa característica permite um aumento na

performance do processo de varredura (busca de elementos) da árvore, visto que não haverá a necessidade de, a cada acesso, reabrir o documento, montar a estrutura e apenas depois poder percorrê-la. Na Figura 8 são apresentados um documento XML e sua representação após o processamento.

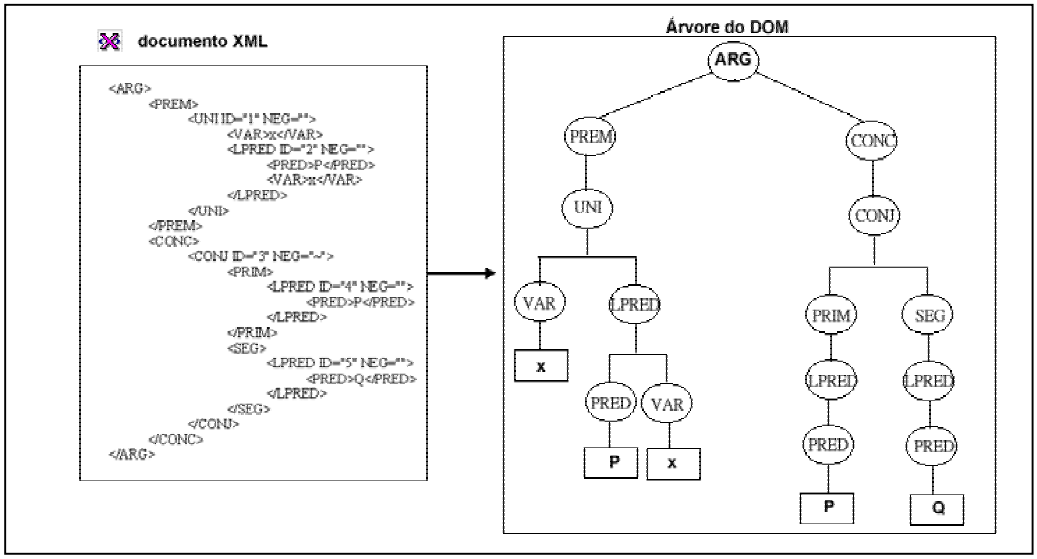

Figura 8: Representação de uma árvore DOM

Uma vantagem do DOM para manipular documentos XML é que ele possibilita o acesso aleatório aos elementos da árvore e os nós podem ser criados e anexados a qualquer momento e em qualquer lugar da árvore XML (ANDERSON, 2001). Devido à existência do DTD é possível garantir que as alterações na árvore serão válidas apenas se forem bem formadas, ou seja, se estiverem de acordo com o DTD associado ao documento XML em questão.

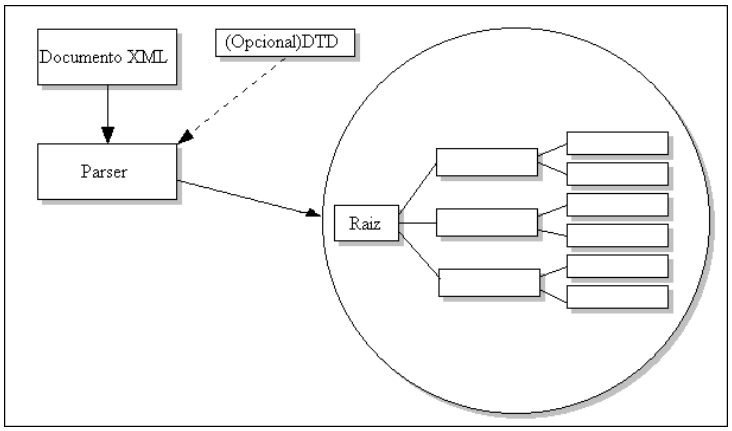

Figura 9: Representação do Modelo Conceitual(FEDERIZZI, 2002) – Modificada

De acordo com a figura 9, a API DOM analisa um documento XML, validado opcionalmente por uma DTD, a partir da utilização de um *parser* que faz a leitura do documento e o armazena na memória no formato de uma árvore. Sendo assim, o conteúdo e a estrutura do documento são identificados e, com a utilização dos métodos existentes na API DOM, a árvore gerada pode ser manipulada. No trabalho aqui desenvolvido é utilizado o DTD de forma obrigatória, dada a necessidade relevante da fórmula estar bem formada para ser validada ou não pelo sistema.

A árvore gerada e armazenada em memória ficará disponível até a finalização da execução do programa. Assim, durante essa execução, pode-se percorrer e buscar os elementos armazenados para seu correto tratamento no provador. Na figura 10 são apresentados os métodos da API DOM mais utilizados nesse trabalho.

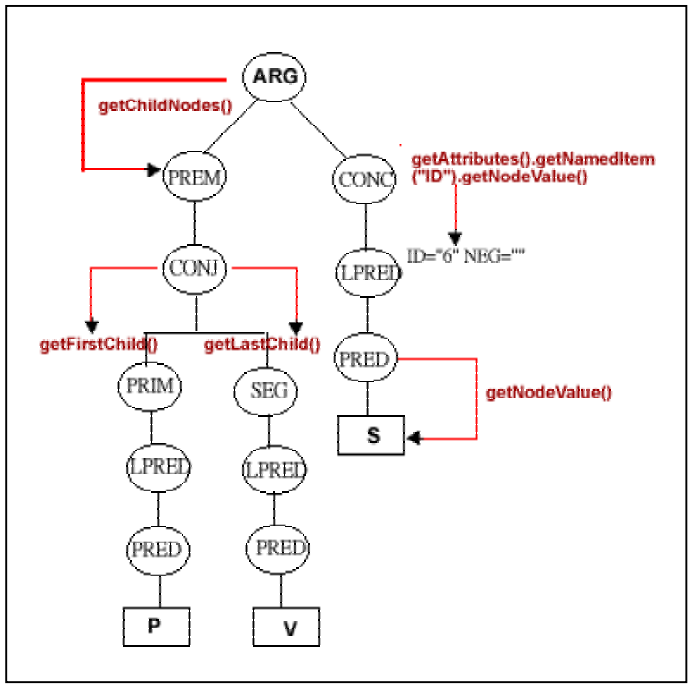

Figura 10: Métodos da API DOM

Através dos métodos citados na Figura 10, a implementação de algumas funcionalidades do provador, como acesso ao conteúdo dos elementos, aos filhos dos nós, aos atributos dos elementos, ao primeiro e último filho de uma *tag* foram facilitadas. Assim, com a presença de um identificador do elemento, foi possível saber todos os dados relativos a qualquer elemento da árvore e o papel que este elemento representava na hierarquia da árvore do documento XML.

2.4 Considerações

A definição de lógica e, mais especificamente, os conceitos de lógica proposicional e quantificacional, é a base para a realização deste trabalho. No entanto, não se buscou no texto supracitado fazer um tutorial sobre lógica. Foram descritas as definições necessárias para o entendimento das etapas de desenvolvimento do provador.

O método dos tableaux, o sistema adotado para a definição do algoritmo automático de prova, foi apresentado a partir da definição dos seus termos e da descrição do tratamento dado a cada conectivo e cada quantificador da lógica quantificacional. Sendo assim, o refinamento proposto no trabalho é apresentado a seguir, já com o embasamento necessário para a sua compreensão.

Para alcançar o objetivo principal do trabalho, a implementação do provador, foi necessário adquirir conhecimentos teóricos e práticos sobre XML e sobre a API que trata os documentos XML em JAVA, DOM. Sem esses conhecimentos, a proposta de refinamento, em especial a que lida com a ordenação dos elementos que compõem a fórmula, alcançaria uma complexidade ainda maior. Desta forma, foram apresentados os conceitos utilizados tanto de XML como da API DOM que foram necessários para a implementação do provador.

Capítulo 3

A Utilização de XML para a definição da estrutura das Fórmulas Quantificacionais

3.1 A aplicação

A estrutura do documento XML é que fornecerá a base para o tratamento adequado de cada elemento que compõe as fórmulas quantificacionais. Assim, o DTD deve atender às necessidades da lógica quantificacional no que tange à especificação dos elementos, da relação existente entre eles e da relevância e obrigatoriedade das partes que o compõem.

Na figura 11 é apresentado o DTD para fórmulas quantificacionais. No DTD são especificados os elementos que compõem a linguagem quantificacional. Ou seja, é especificado que um argumento (ARG) é composto por zero ou mais premissas (PREM) e, necessariamente, uma conclusão (CONC).

A seguir, é definido que uma premissa (PREM) pode ser um quantificador universal (UNI), um quantificador existencial (EXI), uma fórmula atômica (LPRED), uma conjunção (CONJ), uma disjunção (DISJ), um condicional (COND) ou um bicondicional (BIC). O operador que separa os elementos da premissa (PREM) é o operador da disjunção em XML, ou seja, a barra vertical. Desta forma, é definido que uma premissa pode conter um dos elementos por vez, pois a barra vertical funciona como uma disjunção exclusiva. A conclusão (CONC) segue o mesmo formato da premissa.

Os quantificadores universal (UNI) e existencial (EXI) devem conter obrigatoriamente uma variável e um dos sete elementos básicos da linguagem quantificacional (fórmula atômica, conjunção, disjunção, condicional, bicondicional, fórmula universal e fórmula existencial). A fórmula atômica (LPRED) deve ter,

necessariamente, um sinal predicativo e zero ou mais variáveis e/ou letras nominais. Variáveis e letras nominais são do tipo *#PCDATA*, ou seja, são caracteres.

Um condicional (COND) tem que conter, obrigatoriamente, um termo antecedente (ANT) e um termo conseqüente (CONS), por isso os termos estão separados por vírgula (,). Desta forma, um documento XML que contivesse um elemento COND somente com o termo ANT, por exemplo, seria mal formado de acordo com o DTD. Se um dos elementos não tivesse necessariamente que ocorrer, deveria ser colocado o operador de multiplicidade "*" junto ao elemento para indicar que poderia conter 0 (zero) ou mais elementos, assim se ele não existisse numa determinada situação, a fórmula em questão ainda teria validade perante o DTD

A disjunção (DISJ), a conjunção (CONJ) e o bicondicional (BIC) seguem o mesmo padrão: possuem, necessariamente, dois elementos, o primeiro termo (PRIM) e o segundo (SEG). Os termos antecedente, conseqüente, primeiro e segundo, seguem o padrão das premissas e da conclusão, ou seja, podem ser uma fórmula universal, uma fórmula existencial, uma conjunção, uma disjunção, um condicional, um bicondicional ou uma fórmula atômica.

Pode ser também observado na figura 11 que os elementos próprios da linguagem quantificacional (fórmula universal, fórmula existencial, fórmula atômica, condicional, bicondicional, conjunção e disjunção) podem possuir o atributo negação (NEG), que foi identificado como *CDATA*, pois esta é a palavra reservada em XML para identificar um tipo de atributo *caracter*. A presença do atributo identificador (ID) em cada elemento se deve à necessidade de sua utilização na identificação do elemento na árvore que será gerada a partir do documento XML.

```
<!ELEMENT ARG (PREM*, CONC)>
<!ELEMENT PREM (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT CONC (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT VAR (#PCDATA)>
<!ELEMENT LNOM (#PCDATA)>
<!ELEMENT PRED (#PCDATA)>
<!ELEMENT LPRED (PRED, (VAR*|LNOM*), (VAR*|LNOM*))>
<!ELEMENT COND (ANT,CONS)>
<!ELEMENT BIC (PRIM,SEG)>
<!ELEMENT DISJ (PRIM,SEG)>
<!ELEMENT CONJ (PRIM,SEG)>
<!ELEMENT ANT (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT CONS (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT PRIM (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT SEG (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT UNI (VAR, (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI))>
<!ELEMENT EXI (VAR, (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI))>
        <!ATTLIST LPRED ID CDATA #IMPLIED>
        <!ATTLIST LPRED NEG CDATA #IMPLIED>
        <!ATTLIST COND ID CDATA #IMPLIED>
        <!ATTLIST COND NEG CDATA #IMPLIED>
        <!ATTLIST BIC ID CDATA #IMPLIED>
        <!ATTLIST BIC NEG CDATA #IMPLIED>
        <!ATTLIST CONJ ID CDATA #IMPLIED>
        <!ATTLIST CONJ NEG CDATA #IMPLIED>
        <!ATTLIST DISJ ID CDATA #IMPLIED>
        <!ATTLIST DISJ NEG CDATA #IMPLIED>
        <!ATTLIST UNI ID CDATA #IMPLIED>
        <!ATTLIST UNI NEG CDATA #IMPLIED>
        <!ATTLIST EXI ID CDATA #IMPLIED>
        <!ATTLISTEXI NEG CDATA #IMPLIED>
```

Figura 11: DTD para fórmulas quantificacionais

A partir da definição do DTD é possível associa-lo ao conteúdo do documento XML (figura 12) e validá-lo. O documento XML é preenchido pelos elementos da fórmula digitada pelo usuário ou inseridos diretamente através de um editor de texto ou de uma ferramenta própria para trabalhar com documentos XML.

Fórmula: $\forall x(Px \rightarrow Qx), Pa \vdash Qa$

```
ARG>
        <PREM>
                <UNI ID="0" NEG="">
                        <VAR>
                         x
                        </VAR>
                        <COND ID ="1" NEG="">
                                <ANT>
                                        <LPRED ID ="3" NEG="">
                                                <PRED>P</PRED>
                                                <VAR>x</VAR>
                                        </LPRED>
                                </ANT>
                                <CONS>
                                        <LPRED ID="4" NEG="">
                                                <PRED>Q</PRED>
                                                <VAR>x</VAR>
                                        </LPRED>
                                </CONS>
                        </COND>
                </UNI>
        </PREM>
        <PREM>
                <LPRED ID ="5" NEG="">
                        <PRED>P</PRED>
                        <LNOM>a</LNOM>
                </LPRED>
        </PREM>
        <CONC>
                <LPRED ID ="5" NEG="">
                        <PRED>Q</PRED>
                        <LNOM>a</LNOM>
                </LPRED>
        </CONC>
</ARG>
```

Figura 12: Documento XML preenchido pelos elementos da fórmula predicativa.

Após a etapa de inserção dos elementos no documento XML e sua validação através do DTD, é possível tratá-lo no ambiente de programação JAVA. Para isso, é utilizado um *parser* que faz a leitura do documento, gerando em seguida, uma árvore na memória como forma de representação do documento. Conforme citado no capítulo 2, seção 2.3.2, a API DOM trata estes elementos através de um conjunto de métodos próprios para trabalhar com a estrutura gerada.

3.2 Considerações

A partir da definição da estrutura da linguagem foi possível estabelecer os critérios necessários para o tratamento das fórmulas quantificacionais. As ações

realizadas para a definição das etapas de prova levam em consideração o tratamento de cada elemento e a estrutura destes elementos é fundamental para que esta ação tenha êxito.

No próximo capítulo é explicitado o refinamento realizado no método dos tableaux para tornar a árvore de prova, que será definida a partir de cada problema, mais eficiente. Esta eficiência está diretamente relacionada, neste trabalho, ao número de nós necessários para a resolução de cada problema.

Assim, o trabalho realizado de implementação está relacionado ao entendimento da árvore gerada pela API DOM, a partir do documento XML, da lógica utilizada para trabalhar esses elementos e do refinamento dado ao método para torná-lo mais eficiente.

Capítulo 4

Refinamentos do Método dos Tableaux

Conforme visto anteriormente, o método dos tableaux é estruturado na forma de árvore, sendo assim, a cada manipulação de um elemento podem ser acrescentados um ou mais níveis na árvore. Se este método for utilizado sem um tratamento para a questão da ampliação dos níveis da árvore, pode ficar computacionalmente ineficiente, dada a possível redundância e os acréscimos desnecessários de elementos. Logo, faz-se necessário um trabalho de refinamento na árvore para que sejam incluídos apenas os nós absolutamente necessários para a prova, segundo critérios estabelecidos no primeiro refinamento.

Além de controlar a quantidade de nós da árvore de prova, deve ser realizado, para as fórmulas quantificacionais, um tratamento das variáveis, ou seja, deve ser utilizado um algoritmo de unificação para identificar a melhor forma de instanciar uma determinada variável. Estas questões são referentes ao segundo refinamento proposto nesse trabalho.

4.1 Primeiro Refinamento

O primeiro refinamento funciona em sua extensão para fórmulas proposicionais, fazendo com que aconteça uma diminuição significativa, conforme será visto posteriormente, na quantidade de nós da árvore de prova.

Para determinar o refinamento foi necessário estabelecer as situações que cada fórmula poderia apresentar em suas etapas de prova e definir estratégias de manipulação de nós. Assim, foi definida uma situação geral de refinamento, vista a seguir:

- Depois da inserção de uma fórmula P na árvore de prova, sempre é verificado se o elemento inserido no ramo possui uma contradição. Caso

tenha, o nó deste ramo será retirado da lista de nós que serão usados para posteriores inserções.

Um outro ponto analisado foi o tratamento diferenciado dos conectivos cujos caminhos são bifurcados daqueles que seguem caminhos seqüenciais. Assim, a ordem em que os elementos são tratados nas etapas de prova é primordial para que a árvore tenha um número menor de nós.

Desta forma, foi efetuada a ordenação dos elementos que compõem a fórmula a ser analisada. O motivo para esta ordenação foi evitar o máximo possível de bifurcações na árvore de prova, ou seja, serão priorizados como elementos iniciais aqueles que são ligados aos conectivos lógicos seqüenciais, ou aqueles que não possuem conectivos lógicos. Desta forma, a ordenação dos elementos segue o critério enumerado abaixo:

1. Fórmula Atômica, Fórmula Atômica Negada
2. Conjunção, Disjunção Negada, Condicional Negado
3. Disjunção, Condicional, Conjunção Negada
4. Bicondicional, Bicondicional Negado

São apresentadas, a seguir, as particularidades dos conectivos em cada refinamento:

- Conjunção

1. Existindo um elemento contraditório ao primeiro elemento da conjunção, então não é necessário incluir o segundo elemento. O ramo em questão é retirado do vetor de nós folhas, sendo assim, não serão mais incluídos elementos no ramo em que ocorreu a contradição (Figura 13).

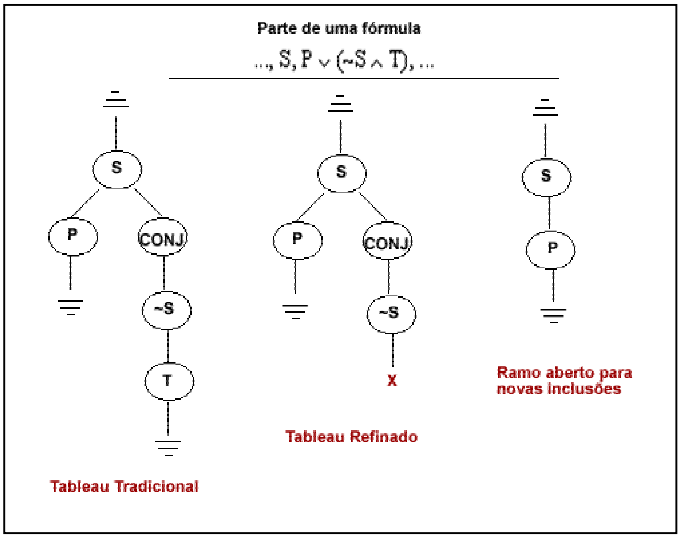

Figura 13: Tratamento da Contradição - Conjunção

- Disjunção

1. Existindo uma fórmula que contradiga um dos elementos da disjunção, apenas o ramo da bifurcação em que não ocorreu a contradição continuará a existir como possibilidade para novas inclusões. Se os dois elementos da disjunção resultarem numa contradição, o ramo em que a disjunção foi inserida será eliminado dos nós folhas, ou seja, estará interrompido para novas inclusões (Figura 14).

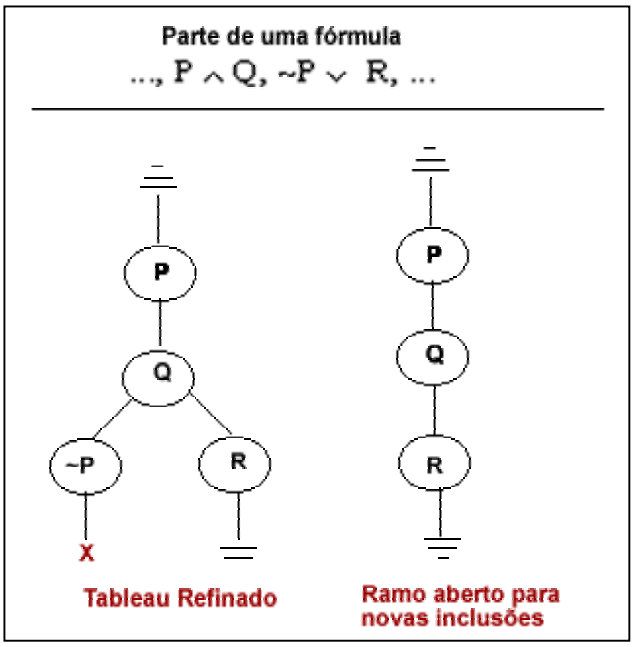

Figura 14: Tratamento da Contradição – Disjunção

- Bicondicional

1. Seja uma fórmula do tipo P ↔ Q e, caso seja encontrada uma contradição em algum dos ramos depois da inclusão dos elementos, este(s) ramo(s) será(ão) eliminado(s) do vetor nós folhas (figura 15).

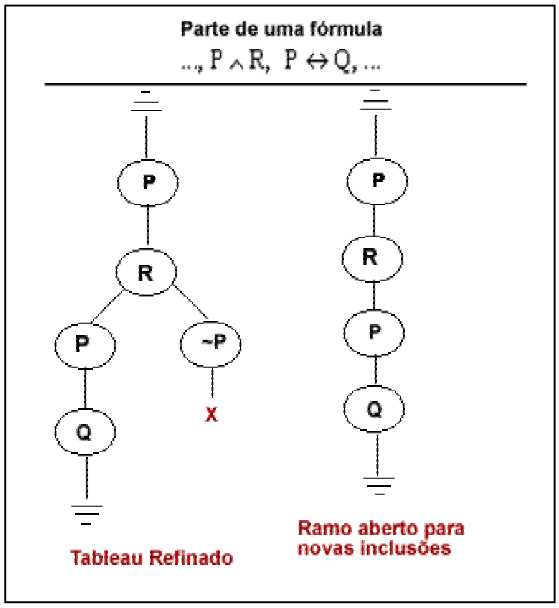

Figura 15: Tratamento da Contradição - Bicondicional

Com a descrição do procedimento envolvido para a inserção dos elementos de forma refinada para os conectivos supracitados é possível entender o comportamento dos outros conectivos, pois a estrutura na árvore varia entre seguir um caminho seqüencial ou executar uma bifurcação. A diferença está na forma como cada elemento é inserido na árvore, ou seja, se um dado elemento deve ser inserido negado ou simplesmente deve ser inserido na sua forma original.

4.2 Segundo Refinamento

No segundo refinamento são apresentadas as etapas necessárias para o tratamento das fórmulas quantificacionais. São verificadas as particularidades dos quantificadores universal e existencial para tornar possível a identificação da melhor forma de manipulá-los durante a execução do problema.

Algoritmos de unificação são citados em REEVES (1987), LOVELAND(1978), FITTING (1990) e FENDT (2000) e são responsáveis pelo tratamento das variáveis

nas etapas de prova. O algoritmo proposto nesse trabalho segue uma linha em comum com os demais algoritmos, ou seja, também busca a instanciação correta das variáveis quantificadas universalmente.

De modo semelhante ao algoritmo desenvolvido em FENDT (2000), o algoritmo proposto nesse trabalho tem como base um tableau inicial e, a partir dele, todos os ramos são derivados. Assim, a forma de fechamento de cada ramo é o caminho percorrido desde o início do tableau até o ponto da última instanciação de variável ou do último tratamento de um conectivo lógico.

Outro ponto em comum com o trabalho de FENDT (2000) é o tratamento dado às variáveis livres. Ou seja, as variáveis livres são substituídas por novas letras nominais. No entanto, no trabalho aqui proposto, as variáveis livres não são trocadas inicialmente, e sim, à medida que as fórmulas estão sendo usadas, e não são trocadas necessariamente por letras nominais “novas”, mas por letras nominais que sejam adequadas para a etapa de prova (ou seja, que podem proporcionar ao predicado no qual elas estão associadas possibilidades de encontrar uma contradição).

Em relação aos quantificadores vácuos, diferentemente do que ocorreu no trabalho acima, eles não eliminados do tableau inicial, no entanto, não são chamados pelos métodos de instanciação, pois para serem chamados seria necessário que um elemento tivesse uma variável ligada a ele.

No método dos tableaux, o tratamento do universal se dá a partir da instanciação de sua variável por qualquer constante. Qual constante utilizar depende do contexto da fórmula. Elucidar e sistematizar a relação entre o quantificador e o contexto em que ele está inserido é um dos objetivos deste trabalho.

O diferencial da instanciação de variáveis para o quantificador existencial se deve ao fato de que a existência requer um indivíduo original, ou seja, não é possível instanciar as variáveis associadas aos quantificadores da existência indivíduos que já estão associados a outros predicados. Desta forma, o processo de

substituir a constante pela variável no quantificador existencial é mais simples, pois basta substituir a variável por uma constante que não esteja vigente no ramo da árvore em que está ocorrendo a instanciação.

Para o tratamento dos quantificadores é necessário distinguir duas etapas: a ordenação dos elementos que compõem a fórmula e a instanciação das variáveis. A primeira etapa dá um caráter intuitivo ao provador, porque a ordenação é responsável pela indicação do próximo passo na etapa de prova. A segunda etapa indica quais letras nominais devem ser escolhidas no momento da instanciação da variável; assim será possível o fechamento dos ramos da árvore ou o entendimento que um determinado ramo não poderá ser fechado, utilizando uma menor quantidade de nós. Para o desenvolvimento do algoritmo de unificação para estas duas etapas foram estabelecidas duas heurísticas, uma com os critérios para a ordenação das fórmulas e a outra com os critérios de instanciação das variáveis. Essas heurísticas são descritas na próxima seção.

4.2.1 Critérios de Ordenação de Fórmulas Quantificacionais

O tableau inicial, conforme explicitado anteriormente, é composto pelas premissas (se existirem) e a negação da conclusão. Sendo assim, a partir desta inicialização é preciso estabelecer a heurística necessária para definição do algoritmo que fornece o próximo passo para o provador.

Devem ser consideradas para a construção da heurística de ordenação as seguintes etapas:

- priorizar a instanciação das variáveis relacionadas aos quantificadores existenciais, pois assim é possível diminuir a utilização de diferentes letras nominais;
- priorizar a utilização dos conectivos lógicos que não bifurcam o caminho na árvore de prova, para a diminuição da quantidade de nós.

A heurística de ordenação para as fórmulas quantificacionais fornece os caminhos de prova que devem ser seguidos. Então, cada item enumerado abaixo

deve ser considerado, na ordem em que são apresentados, para a inferência do próximo passo.

1. Fórmula atômica de aridade zero, Fórmula atômica de aridade zero negada – fórmulas atômicas de aridade zero não têm, agregadas a elas, muitos parâmetros para serem analisados no momento da sua inclusão na árvore de prova. Logo, não há a preocupação com o aumento de letras nominais que devam ser tratadas ou aumento de bifurcação nos ramos. (Figura 16)

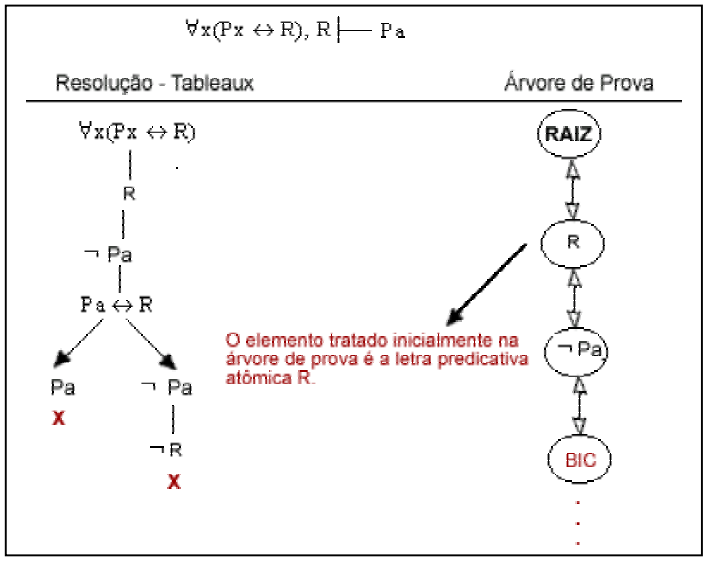

Figura 16: Ordenação relativa à fórmula atômica de aridade zero

2. Maior quantidade de quantificadores existenciais externos – a presença de quantificadores existenciais externos tende a provocar muitas instanciações na fórmula e, caso uma fórmula apresente muitos sinais predicativos, todos eles serão afetados com a substituição de uma determinada variável por uma constante que ainda não tenha sido usada no ramo em que está ocorrendo a instanciação. Assim, a presença de uma diversidade de letras nominais pode tornar a verificação da contradição entre fórmulas atômicas uma tarefa mais complexa e mais suscetível a erros (Figura 17). Isso se dá pelo fato de que duas fórmulas contraditórias têm que apresentar a mesma formação, tendo como único diferencial a negação.

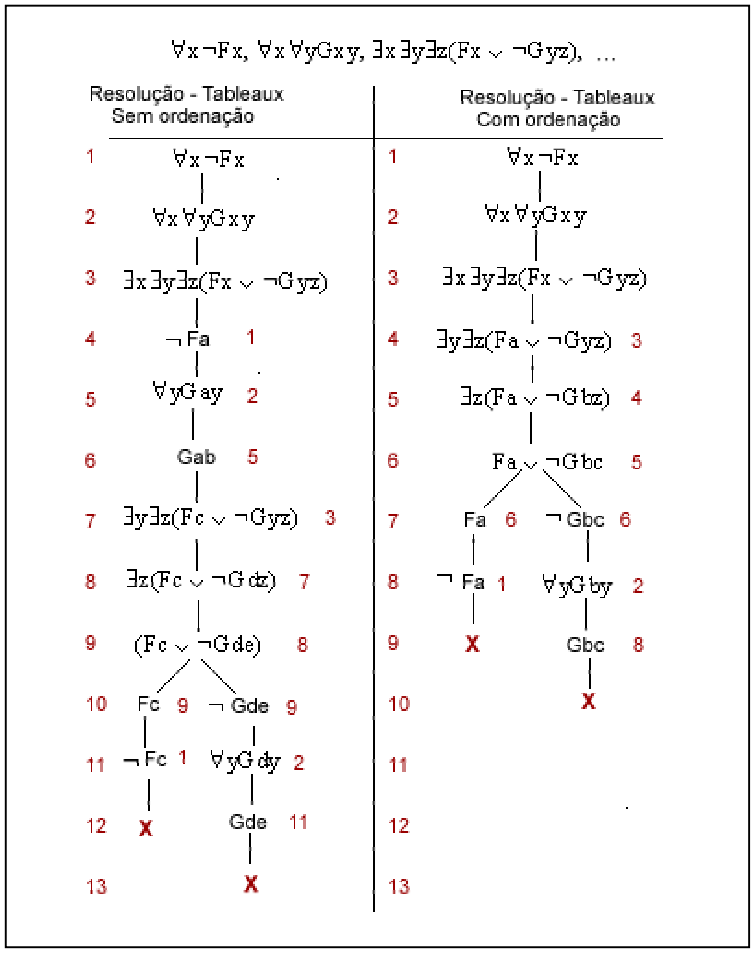

Figura 17: Ordenação pela maior quantidade de quantificadores existenciais externos

3. Maior quantidade de quantificadores existenciais internos – a ordenação tem que considerar sempre as etapas necessárias para a construção da heurística, assim são priorizadas as fórmulas que apresentam quantificadores existenciais (Figura 18).
4. Ordenação por conectivo lógico (visto na seção 4.1).

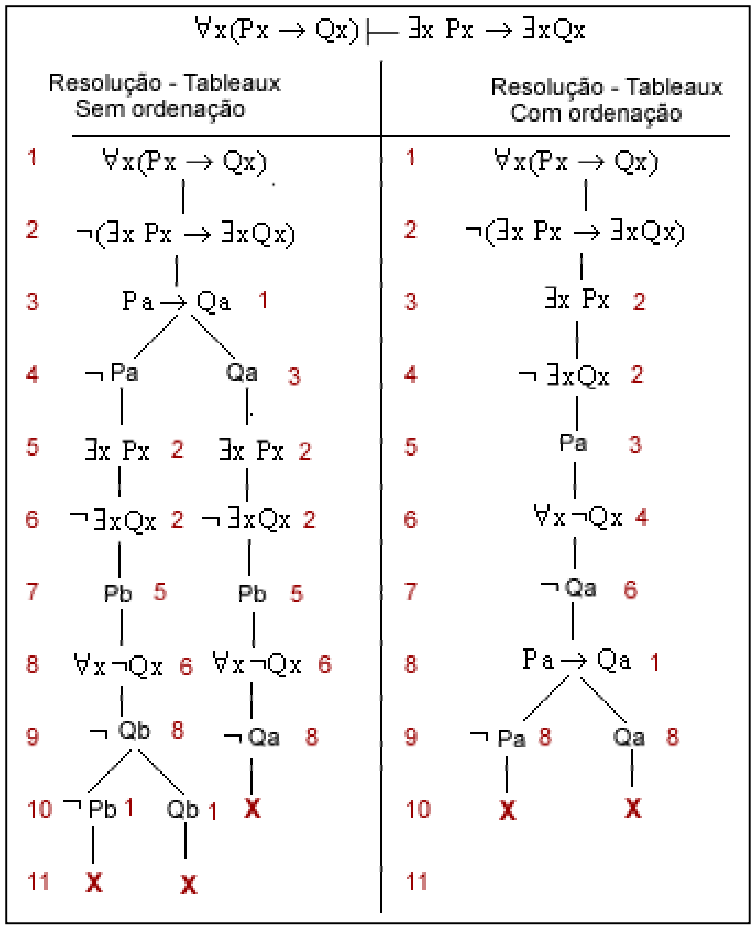

Figura 18: Ordenação pela maior quantidade de quantificadores existenciais internos

Para a ordenação das fórmulas quantificacionais, é necessário fazer a equivalência de fórmulas, considerando a maneira como são trabalhadas as negações dos quantificadores no método dos tableaux. Ou seja, sendo P um conjunto de fórmulas, temos:

$\neg\forall xP \equiv \exists x\neg P$

$\neg\exists xP \equiv \forall x\neg P$

Assim, na etapa de verificação da quantidade de existenciais numa fórmula, a equivalência deve ser realizada para que a ordenação seja feita a partir da representação final de cada elemento da fórmula. Se isso não ocorresse, o processo de ordenação não estaria refletindo as situações reais de cada fórmula. E, estaria, por essa razão, incorreto.

Um outro ponto importante é a forma como cada elemento será inserido na árvore de prova. Assim, se o termo for o antecedente de um condicional, por exemplo, e este antecedente for uma fórmula universal, então o provador tem que fazer a leitura desse antecedente como uma fórmula existencial com o seu conteúdo negado. Desta forma, sendo P e Q conjuntos de fórmulas, é estabelecida a equivalência de acordo com a figura 19.

$$\neg (P \wedge Q) \equiv \neg P \vee \neg Q$$
$$\neg (P \vee Q) \equiv \neg P \wedge \neg Q$$
$$P \rightarrow Q \equiv \neg P \vee Q$$
$$\neg (P \rightarrow Q) \equiv P \wedge \neg Q$$
$$P \leftrightarrow Q \equiv (P \wedge Q) \vee (\neg P \wedge \neg Q)$$
$$\neg (P \leftrightarrow Q) \equiv (P \wedge \neg Q) \vee (\neg P \wedge Q)$$

Figura 19: Equivalência de Fórmulas

Assim, quando cada fórmula for verificada, deve ser realizada a equivalência apresentada na figura 19. Apenas após esta equivalência será possível verificar os critérios de ordenação. Então, será acumulada cada característica apresentada, conforme a seqüência a seguir:

Primeiro: inserção das fórmulas atômicas de aridade zero nas primeiras posições.

Segundo: somatório de Quantificadores Existenciais externos.

Tendo mais de uma fórmula obtido a mesma quantidade de quantificadores existenciais externos, estas fórmulas ficarão nas primeiras posições, mas a ordem delas será escolhida aleatoriamente.

Caso o somatório dos quantificadores existenciais externos resulte em zero, então são verificados os demais casos.

Terceiro: somatório de quantificadores existenciais internos nas fórmulas.

Se mais de uma fórmula apresentar a mesma quantidade de quantificadores existenciais internos, então será realizada uma ordenação por conectivo lógico entre as fórmulas que apresentarem esta mesma situação.

Quarto: ordenação por conectivo lógico.

4.2.2 Instanciação das variáveis

Para tornar possível a instanciação correta das variáveis das fórmulas quantificacionais foi definida uma árvore denominada árvore de prova. Ela é iniciada com um nó raiz e cada nó é composto por um atributo denominado **Histórico** responsável por armazenar, respectivamente, o elemento da negação (~) ou o espaço vazio (indicando que o elemento não é negado), o sinal predicativo juntamente com as variáveis (se existirem), o identificador do sinal predicativo e, caso esse sinal predicativo seja um elemento interno a quantificadores, então são guardados, também, os quantificadores, na ordem em que eles aparecem na fórmula, juntamente com o seu respectivo identificador. Além deste atributo, o nó ainda possui quatro outros atributos: “esquerdo”, “centro”, “direito” e o “noPai”; os três primeiros são usados para referenciar os nós filhos e o quarto para referenciar o nó pai do nó em questão (Figura 20).

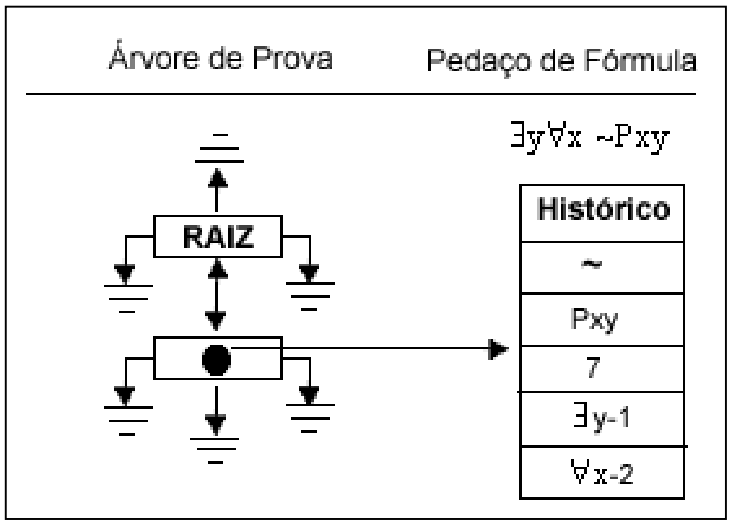

Figura 20: Árvore de prova

Um ponto fundamental para executar a instanciação das variáveis é a identificação do quantificador que a variável está relacionada, pois as formas de instanciação das variáveis relacionadas aos quantificadores universal e existencial são bem distintas. Na próxima seção serão apresentadas as particularidades das duas formas de instanciação.

4.2.2.1 Instanciação das variáveis relacionadas ao Quantificador Existencial

Para a identificação da constante que deverá ser utilizada na instanciação de uma variável relacionada ao quantificador existencial, é necessário verificar as constantes que já ocorrem no ramo e buscar uma que seja distinta. Na realização dessa ação alguns passos devem ser seguidos.

Na figura 21 é apresentado o início do processo de construção de uma árvore de prova. Nesta figura, pode ser verificado que o primeiro elemento que será tratado é a fórmula ∃wTw, que faz parte de uma conjunção e é interno a um quantificador existencial, que por sua vez é o conseqüente de um condicional. Toda a fórmula é, ainda, interna a dois quantificadores existenciais, sendo que o quantificador mais externo está negado. Assim, quando a fórmula é sub-dividida em antecedente e conseqüente (relativo ao conectivo lógico mais externo depois dos quantificadores iniciais), os elementos são ordenados a partir do processo de ordenação apresentado na seção 4.2.1 e é determinada a escolha do segundo elemento da conjunção (∃wTw) para ser trabalhado.

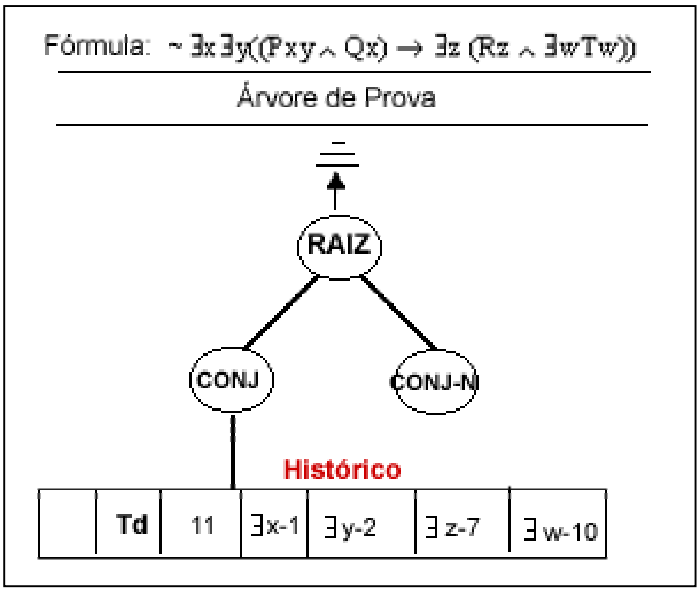

Figura 21: Início do processo de construção da árvore de prova

Na instanciação de variáveis existenciais é necessário um conhecimento prévio do ramo em que está sendo realizada a instanciação. Um outro ponto é a

questão das variáveis dos quantificadores externos, que fazem parte, às vezes, de mais de um ramo, pois, nestes casos, a substituição deve ocorrer de forma igual em todos os ramos. A seguir estas duas situações serão detalhadas.

Na tabela 3 são apresentados os dados que devem ser consultados no momento da instanciação de uma variável. Essa tabela fornece uma memória simplificada da árvore de prova. A partir dela é possível saber qual variável já foi instanciada e a que quantificador ela está associada, pois se houver duas variáveis com a mesma denominação, é primordial saber se ambas estão relacionadas ao mesmo quantificador ou a quantificadores distintos. Para isso, cada quantificador está agregado ao seu identificador. Ainda nesta tabela está armazenada a constante que foi utilizada na instanciação.

Tabela 3: Tabela de instanciação do quantificador existencial

Variável	Quantificador	Constante
x	∃x-1	a
y	∃y-2	b
,z	∃z-7	c
w	∃w-10	d

De acordo com a tabela 3 é possível fazer instanciações corretas, levando em consideração que em uma dada fórmula todas as ocorrências de uma determinada variável relacionada a um mesmo quantificador devem ser instanciadas com a mesma constante. Se isso não for realizado, ocorrerá uma discrepância de constantes na árvore, ou seja, constantes diferentes para mesma variável, relacionada a um mesmo quantificador, num mesmo momento de instanciação. Na figura 22 é apresentada uma instanciação correta, realizada a partir dos dados que estão armazenados na tabela de instanciação. Assim, é possível identificar que existe a ocorrência de uma variável e de um quantificador identificados da mesma forma e, então, utilizar a mesma constante na instanciação do elemento semelhante.

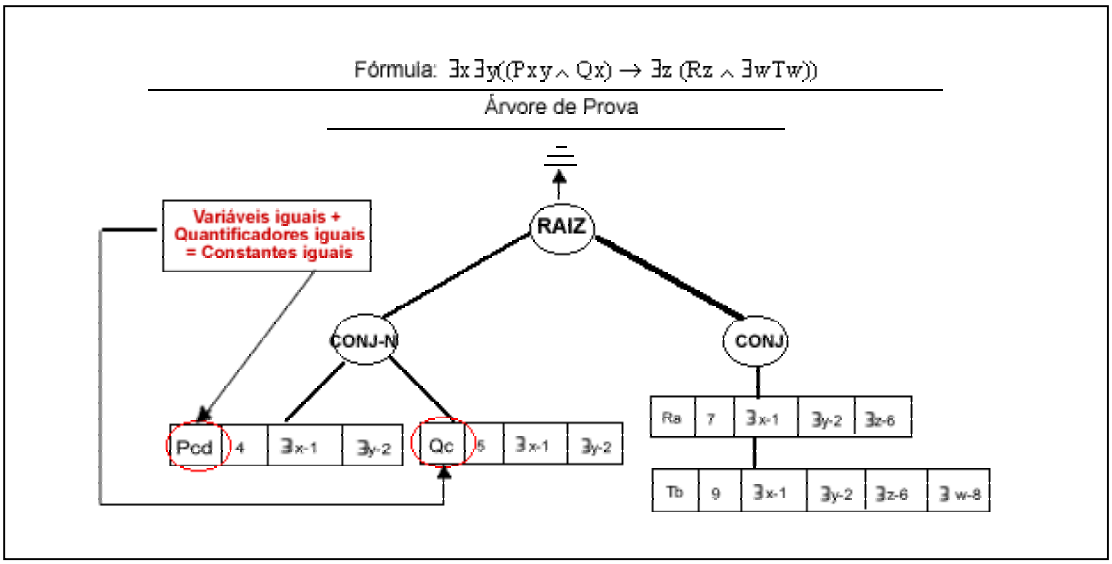

Figura 22: Processo de instanciação de variáveis

A tabela de instanciação do existencial é preenchida à medida que são encontrados quantificadores existenciais nas fórmulas que compõem o argumento. São duas as situações existentes:

- se for o primeiro elemento existencial da primeira fórmula utilizada, então é verificada se na fórmula já existe uma constante:
  - Se existir, a variável será instanciada pela próxima letra desta constante.
  - Se não existir, a variável é instanciada pela letra “a”.
- se já existirem outros elementos instanciados, então é verificado se na tabela de instanciação existe um quantificador com o mesmo identificador do quantificador que está sendo tratado.
  - Se existir, a variável será instanciada com a mesma letra que está armazenada junto ao quantificador de mesmo identificador.
  - Se não existir, a variável será instanciada com a próxima letra depois da maior constante encontrada no(s) ramo(s) em que irá acontecer a instanciação.

A partir da observância desses critérios é que a tabela de instanciação é formada. Desta forma, é definida a “memória” do quantificador existencial durante o processo de prova.

No exemplo apresentado na figura 23 é dada ênfase à necessidade da tabela de instanciação para o processo de busca de quantificadores idênticos, pois, desta forma, não há necessidade de verificar o quantificador em vários ramos; apenas é verificada a tabela de instanciação. Encontrado um quantificador com o mesmo identificador, a variável que estiver no processo de instanciação deverá ser substituída pela mesma constante que está na tabela, referente ao quantificador encontrado.

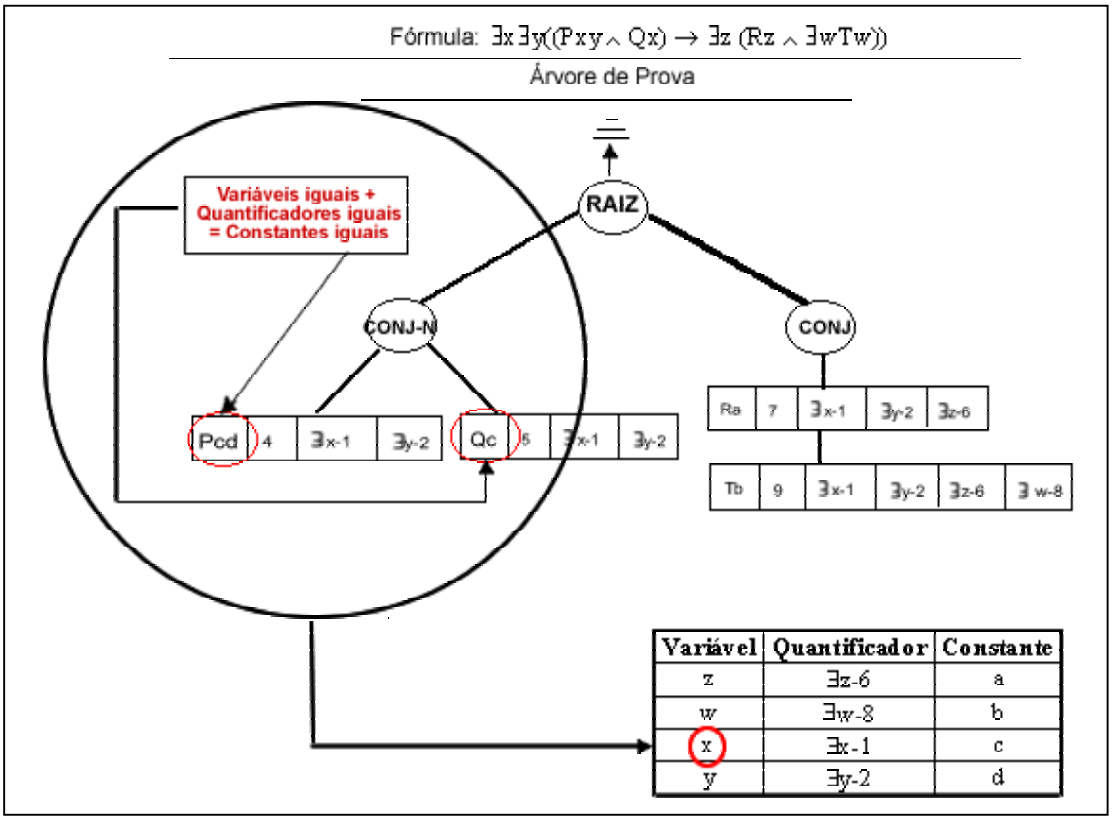

Figura 23: A relação entre a tabela de instanciação e a árvore de prova.

Para os casos em que não forem encontrados quantificadores idênticos, é necessário realizar uma busca nos ramos em que será incluído o novo elemento, para verificar qual constante utilizar, ou seja, para verificar a próxima letra da seqüência de constantes existente nos ramos.

Um fator que deve ser observado são as variáveis existenciais que estão internas a mais de um quantificador. Neste caso, além da análise do ramo e da tabela de instanciação, são consideradas, também, as constantes já instanciadas para o predicado que está agregado à variável que será tratada. Ou seja, a primeira análise é feita no próprio histórico do elemento cuja variável será instanciada. Depois, são realizadas as análises na tabela e no ramo.

O essencial nesta proposta de instanciação é que as constantes sejam substituídas com coerência na árvore de prova. Isso significa seguir a regra de instanciação das variáveis agregadas aos quantificadores existenciais, ou seja, cada variável distinta deve ter uma constante distinta num dado ramo. No entanto, são as variáveis relacionadas aos quantificadores universais que requerem um maior tratamento no que tange a instanciação. São essas instanciações que poderão definir o correto desempenho do provador. Na próxima seção será apresentado o conceito de “instanciação inteligente” como forma de diminuir a quantidade de nós na árvore de prova.

4.2.2.2 Instanciação das variáveis relacionadas ao Quantificador Universal

Na instanciação de variáveis relacionadas ao quantificador universal são verificados os pontos mais complexos da implementação do provador. Isso porque, contrariamente às variáveis existenciais, as variáveis universais necessitam ser instanciadas de tal forma que possibilitem a verificação de contradição nos ramos da árvore de prova ou pelo menos a constatação de que um dado ramo não poderá ser refutado.

Nesse trabalho, é definida “instanciação inteligente” como a possibilidade do algoritmo de unificação realizar a substituição das variáveis universais pelas constantes que mais se adequarem ao contexto do(s) ramo(s) vigente(s) numa dada instanciação, ou seja, pela constante capaz de provocar uma contradição no(s) ramo(s). Esse conceito pode também ser entendido como a execução da instanciação necessária para, na menor quantidade de nós, inferir a refutação ou a constatação da impossibilidade de refutação em um dado ramo. Mas, pelo caráter

indecidível de problemas que envolvem a lógica quantificacional, o algoritmo de unificação não pode garantir uma certeza na resolução de todos os problemas. No entanto, a partir do algoritmo proposto, pretende-se resolver uma grande quantidade de problemas passíveis de serem resolvidos computacionalmente.

Como os quantificadores universais podem ser instanciados com quaisquer constantes, é necessária a definição de estratégias que possam delimitar a forma como se dará a instanciação. Assim, para a unificação de uma constante que satisfaça da melhor forma possível um maior número de nós, é verificado em todos os ramos, a partir dos nós folhas em que serão inseridos os elementos, qual a constante que satisfaz um maior número de ramos. Satisfazer um ramo, neste caso, é possibilitar uma contradição no ramo. Assim, antes de fazer a instanciação, é necessário buscar nos ramos qual a letra mais adequada para a instanciação. Do conjunto de letras observadas no ramo que pode resultar numa contradição, é retornada a letra que possibilita um maior número de ramos com elementos contraditórios. No caso de várias letras possibilitarem a mesma quantidade de contradição, é escolhida a primeira letra como a constante. No caso de nenhuma letra retornar uma contradição, é utilizada a letra “a”.

Na figura 24 são apresentados os detalhes do processo de instanciação universal. É observado que o elemento a ser inserido (∀xPx), pode ser instanciado pelas letras “a” e “c”, pois ambas provocam uma contradição em algum ramo. Mas, no momento da definição da constante é utilizada a letra “c”, pois esta provoca uma quantidade maior de ramos contraditórios. Assim, não apenas são definidas instanciações inteligentes para um ramo, mas a instanciação que melhor se adequa para toda a árvore.

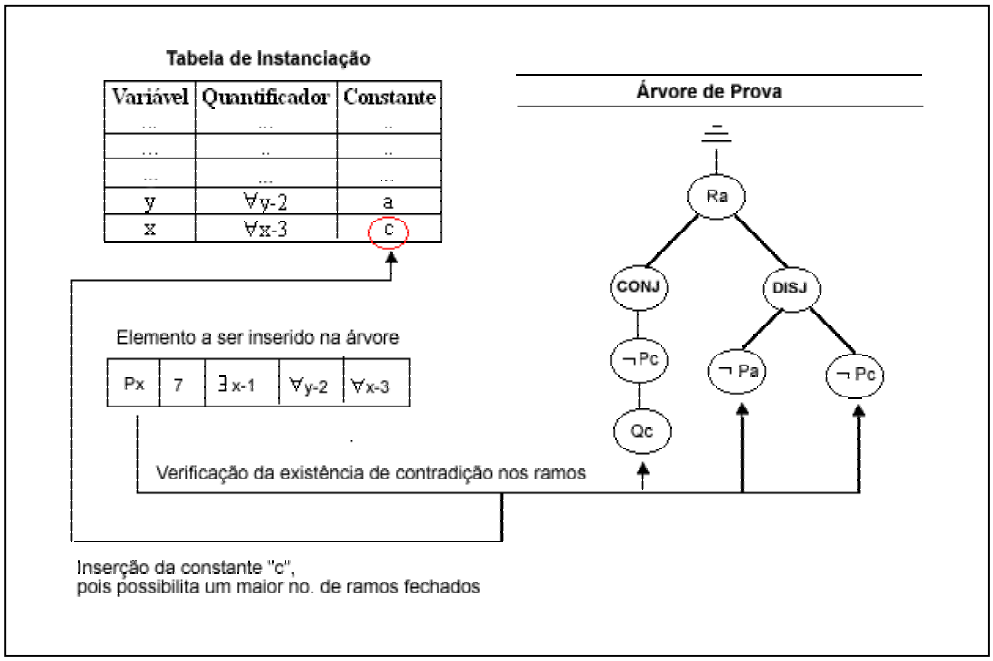

Figura 24: Processo de instanciação de variáveis universais

Na figura 24 é possível verificar, também, que para a instanciação das variáveis universais é utilizada uma tabela de instanciação. A partir do momento em que é definida a letra que melhor satisfaz o conjunto, esta letra é inserida na tabela. As outras fórmulas que possuírem quantificadores com o mesmo identificador deverão instanciar suas variáveis com a mesma letra (neste exemplo, a letra “c”).

As etapas para a instanciação das variáveis universais são sistematizadas a seguir:

- busca de quantificador idêntico na tabela de instanciação;
- caso a busca seja satisfeita, então é realizada a substituição da variável pela constante referente ao quantificador encontrado na tabela;
- caso não encontre quantificador idêntico na tabela, é realizada uma busca em todos os ramos da árvore de prova referentes aos nós folhas atuais. Essa busca procura identificar letras que possibilitariam contradições nos ramos e é sistematizada a seguir:

- verificação da letra que retorna o maior número de contradições;
- caso exista pelo menos uma letra que possibilita contradição, então deverá ser realizada a inclusão do quantificador, da variável e da letra que possibilitou o maior número de contradições na tabela de instanciação;
    - substituição da variável pela constante supracitada e inclusão do elemento na árvore de prova.
- caso não seja encontrada possibilidade de contradições no ramo, então será incluído na tabela de instanciação o referido quantificador, sua variável e a constante “a”.

Na figura 25 é apresentada a instanciação de um elemento que possui uma variável livre. Esta instanciação segue o mesmo padrão da instanciação de variáveis universais, ou seja, é verificado no ramo se há uma possibilidade de contradição e qual instanciação satisfaz o maior número de ramos, caso nenhuma satisfaça a variável livre é substituída pela constante “a”. O diferencial é a não necessidade de busca na tabela de instanciação, dado o fato desta variável não ser agregada a quantificadores.

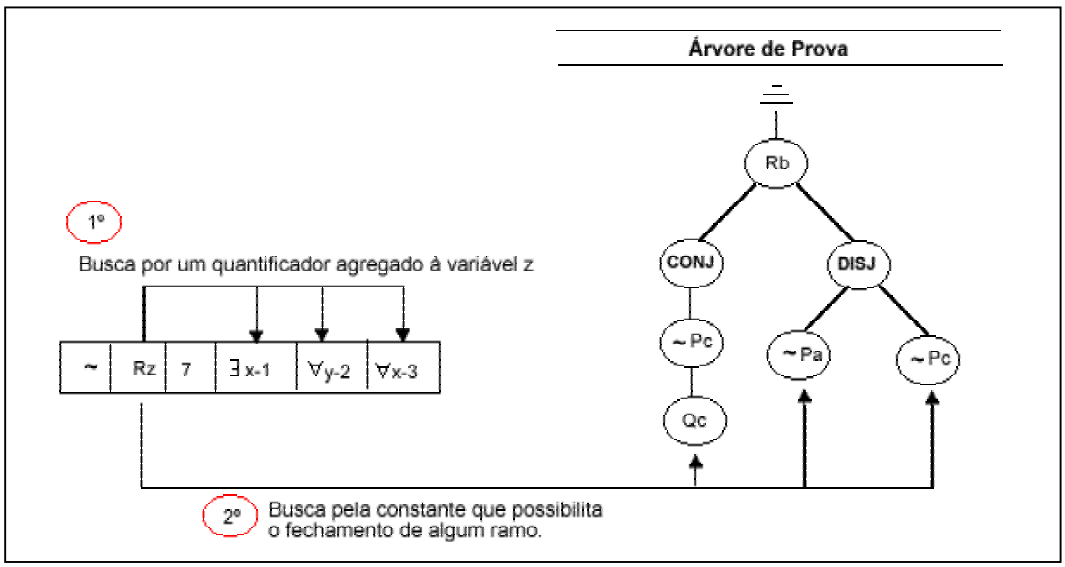

Figura 25: Processo de instanciação de variáveis livres

Um outro ponto considerado durante a verificação de que letra deve ser instanciada numa fórmula quantificada foi a equivalência das fórmulas apresentada na seção 4.2.1 (Figura 19). Assim, a instanciação de um existencial, por exemplo, que é o elemento antecedente de um condicional, deve ser realizada com os critérios de um quantificador universal, pois o condicional é equivalente a uma disjunção de dois elementos, sendo que o primeiro é negado. Desta forma, o existencial negado torna-se um universal com o seu conteúdo negado.

4.3 Etapas de retorno a ramos abertos

A partir da aplicação dos métodos refinados do tableau, é verificada a validade ou invalidade de uma fórmula. No entanto, é necessário definir, em alguns casos, quando não há mais a necessidade de continuar instanciando variáveis universais para a busca de uma contradição. A sistematização desses passos envolve o entendimento de situações genéricas que podem ser usadas para a inferência da interrupção do algoritmo a partir do entendimento da invalidade da fórmula que está sendo analisada.

Depois da inserção de todas as fórmulas que compõem o argumento na árvore de prova, é realizada a verificação da contradição geral, ou seja, da contradição em todos os ramos da árvore. Assim, são relacionados apenas os ramos que não tiverem sido fechados. A partir desses ramos, são observadas as situações descritas a seguir.

Situações para a interrupção imediata do algoritmo e entendimento da invalidade da fórmula:

- A não existência de sinais predicativos contraditórios em algum ramo aberto. Se não há possibilidade de um sinal predicativo (ex. P) ter uma contradição, então não há a necessidade de verificação de quais constantes estão agregadas ao sinal predicativo, nem mesmo a que quantificador ele está agregado, pois será impossível, sem a contradição do sinal predicativo, alcançar uma contradição da fórmula. Para esse ponto, serão considerados que elementos com quantificadores negados já

são vistos com o seu equivalente sem negação e com o conteúdo interno negado (ex.: $\sim\forall xPx \equiv \exists x\sim Px$).

- Sinais predicativos contraditórios agregados a quantidade de constantes diferentes também não podem ser usados para buscar a contradição. Sendo assim, caso esta seja a única possibilidade de elementos contraditórios no ramo, é inferida a invalidade da fórmula.
- A constatação de que algum ramo aberto apresenta somente fórmulas quantificadas existencialmente é suficiente para a inferência da invalidade da fórmula. Pois, não há possibilidade de buscar a modificação de constantes advindas da substituição por variáveis existenciais. Isto porque, qualquer nova instanciação de variáveis existenciais acarreta mudança de constante.

As próximas inferências para o retorno no ramo em busca de elementos contraditórios consideram que as três etapas gerais de interrupção não foram satisfeitas. Assim, é necessário verificar os ramos não fechados e tratá-los particularmente. A invalidade de um dos ramos interrompe o processo e infere a invalidade do argumento.

Em cada ramo é realizada a seleção das fórmulas que apresentam em seus históricos pelo menos um quantificador universal. Encontradas fórmulas com essa característica, devem ser identificadas aquelas que possuem elementos contraditórios, ou seja, que possuem o mesmo sinal predicativo, e que em uma delas é localizado o elemento da negação. O fato do sinal predicativo não ser contraditório para o fechamento da árvore está relacionado às constantes que o acompanham, pois apesar de serem considerados, para esta situação, somente sinais predicativos agregados à mesma quantidade de constantes, estas constantes não sendo semelhantes, provocaram o não fechamento do ramo. Assim, uma nova instanciação pode acarretar em uma situação de contradição, pois, a nova escolha das constantes será realizada de acordo com o ramo em que a fórmula está inserida. Caso exista mais de uma fórmula que apresenta esta situação, é realizado o processo de instanciação em cada uma na ordem em que aparecem na árvore de prova, a partir dos nós folhas. O processo termina com o fechamento do ramo se a

partir da instanciação de alguma das fórmulas ocorrer uma contradição. A contradição, desta forma, interrompe a verificação em outras fórmulas (se houver) no mesmo ramo.

Na figura 26 é apresentado o processo de retorno em um ramo aberto. É verificado que duas fórmulas apresentam em seu histórico apenas quantificadores universais. Então, o segundo ponto é observado: a possibilidade de contradição. A primeira fórmula (~Abb) é descartada, pois não é possível fazer uma instanciação que seja coerente com o elemento que lhe é contraditório, visto que qualquer instanciação nas variáveis agregadas ao elemento retornaria a constantes idênticas, isso porque as duas variáveis estão ligadas ao mesmo quantificador. Portanto, não há possibilidade de contradição na primeira situação. É observado, neste ponto, que a equivalência já está sendo tratada, ou seja, o existencial negado é tratado como um universal com o seu conteúdo negado. Busca-se, então, o próximo elemento que atendeu ao requisito do histórico com quantificadores universais (~ Aab). Assim, a partir da nova instanciação do condicional ao qual este elemento está agregado, é possível obter uma contradição, pois como o condicional está ligado a dois quantificadores universais, é possível instanciar o elemento com duas constantes distintas.

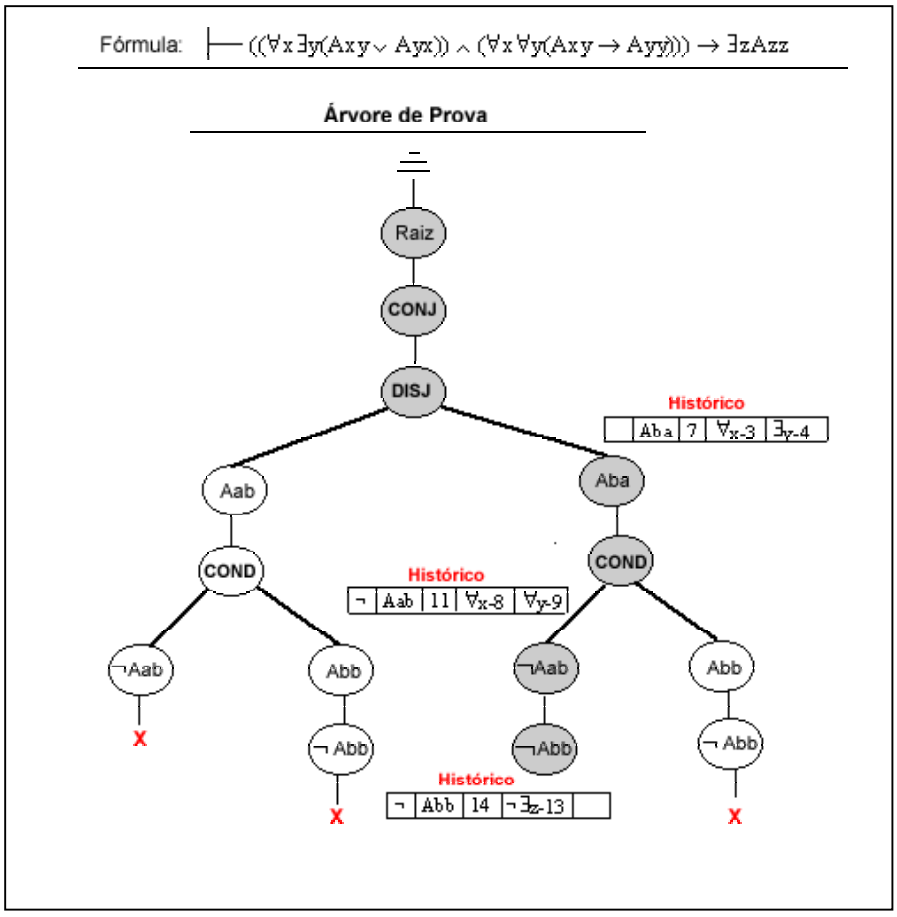

Figura 26: Retorno às variáveis quantificadas universalmente

Deve ser considerada a diferença em realizar novas instanciações em fórmulas que produzam bifurcações e em fórmulas que tratam caminhos seqüenciais, assim, são priorizadas, de acordo com os critérios de ordenação de fórmulas vistos anteriormente (seção 4.2.1), as fórmulas que não bifurcam o caminho. Mas, tendo apenas essas fórmulas que provocam bifurcações para realizar o retorno às instanciações, deve ser analisado se existe contradição para ambos os ramos. Assim, só serão realizadas instanciações que possibilitarem contradição, ou que sejam o caminho para contradições futuras.

O entendimento de que um dado elemento é caminho para uma futura contradição se dá pelo histórico que cada nó da árvore de prova carrega consigo. Então, ainda que a instanciação de um dado elemento num dos ramos da bifurcação não provoque uma contradição de imediato, pode ser verificada a existência de uma

fórmula quantificada universalmente no ramo que venha a ser contraditória de alguma fórmula que se encontra no ramo que ficará em aberto.

Para fazer a nova substituição, todo o processo de instanciação de variáveis é realizado. Ou seja, é criada a tabela de instanciação, verificado qual elemento é contraditório no ramo, substituída a variável pela mesma constante da fórmula contraditória encontrada e, ao final, verificado no ramo se ocorreu a contradição (Figura 27).

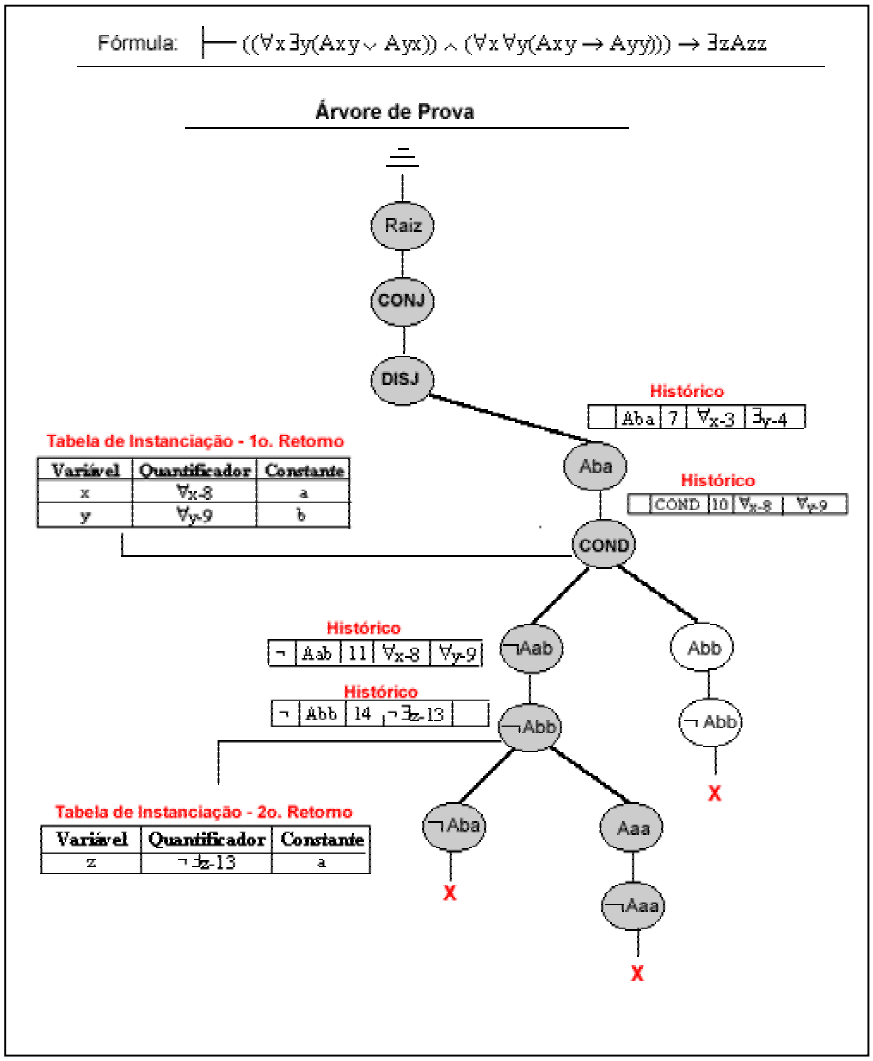

Figura 27: Etapas para o retorno às variáveis quantificadas universalmente

4.4 Considerações

Neste capítulo foram desenvolvidas as etapas necessárias para a definição lógica do provador de teoremas. A ordenação das fórmulas para a inserção na árvore de prova levando em consideração os elementos que a compõe foi fundamental para a instanciação correta das variáveis. E, além da ordenação das fórmulas do tableau inicial, todas as partes de uma fórmula, como os componentes de uma disjunção, por exemplo, foram analisados e ordenados para posterior inserção na árvore. Essa ordenação dos elementos que estão ligados aos conectivos binários e a prioridade dada para as fórmulas que apresentassem em sua composição o maior número de variáveis quantificadas existencialmente reduziu os problemas de instanciação e possibilitou uma maior eficiência na árvore de prova, no que tange a quantidade de nós gerados.

A partir dos elementos ordenados, foi possível trabalhar com uma "instanciação inteligente" das variáveis quantificadas. Então, ao invés de instanciar inicialmente toda a fórmula, a instanciação ocorreu por etapas, ou seja, na medida em que os elementos eram selecionados para a inserção na árvore de prova. Para isso ser realizado sem ocorrer discrepâncias de constantes na árvore, ou seja, variáveis referentes a um mesmo quantificador num mesmo momento de instanciação serem instanciadas por constantes distintas, foi inferido o conceito de tabela de instanciação. A partir desta tabela, a lógica para a instanciação foi aplicada com êxito, pois no momento da substituição de uma variável por uma letra nominal, além de ter o ramo para o entendimento de que letra deveria ser usada para o processo de instanciação, o elemento também possuía um histórico resumido dos elementos que já tinham sido instanciados em outros ramos.

Um ponto que foi verificado para evitar o eterno retorno do algoritmo em ramos abertos que apresentassem variáveis quantificadas universalmente foi o estabelecimento dos critérios para a constatação de que os elementos universais poderiam provocar contradições no ramo. Ou, ainda, para a constatação de que, apesar da existência de elementos ligados aos quantificadores universais no ramo, tais elementos não teriam possibilidade de provocar contradições. Além disso, foram

definidos os critérios gerais de interrupção do algoritmo e da inferência de que uma dada fórmula é inválida.

Na próxima seção serão apresentadas as etapas da implementação do provador. Desde a definição das estruturas necessárias para o armazenamento dos dados utilizados para a verificação da validade ou invalidade das fórmulas até os métodos definidos para tornar possível a ordenação e a instanciação.

Capítulo 5

Etapas da Implementação do Provador para Fórmulas Quantificacionais

Para um melhor entendimento das etapas envolvidas na implementação do provador é apresentada, na Figura 28, a estrutura de funcionamento do sistema. A classe **documentoClass** é a responsável por abrir o documento XML (Figura 28a) e por utilizar o *parser* DOM para gerar a árvore na memória (Figura 28b), permitindo, assim, que essa árvore seja manipulada na classe **provadorClass** (Figura 28c). Desta forma, são realizadas as operações nos elementos através do exemplar que a classe **provadorClass** faz das classes **ordenacaoClass** e **arvoreClass**. E, com os métodos contidos nessa última é possível apresentar a árvore de prova refinada e verificar a validade ou invalidade da fórmula (Figura 28d).

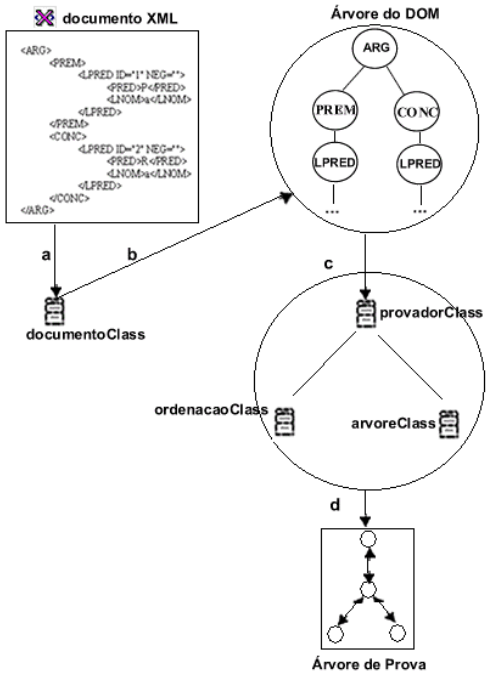

Figura 28: Estrutura do Sistema

Nas seções seguintes serão analisadas as etapas de prova, desde a ordenação dos elementos no vetor base até a instanciação das variáveis

quantificadas existencialmente e universalmente. O bicondicional, por sua estrutura diferenciada, será analisado numa seção à parte, assim como o processo de retorno às fórmulas, nos casos em que uma única instanciação dos elementos não foi suficiente para a prova da fórmula.

5.1 Etapa de Ordenação

Para fazer a ordenação de uma fórmula é passada como parâmetro a árvore gerada pela API DOM (figura 29) por um exemplar da classe **ordenacaoClass**. Assim, é possível acessar todos os elementos da fórmula, executando as operações de ordenação definidas na seção 4.2.1.

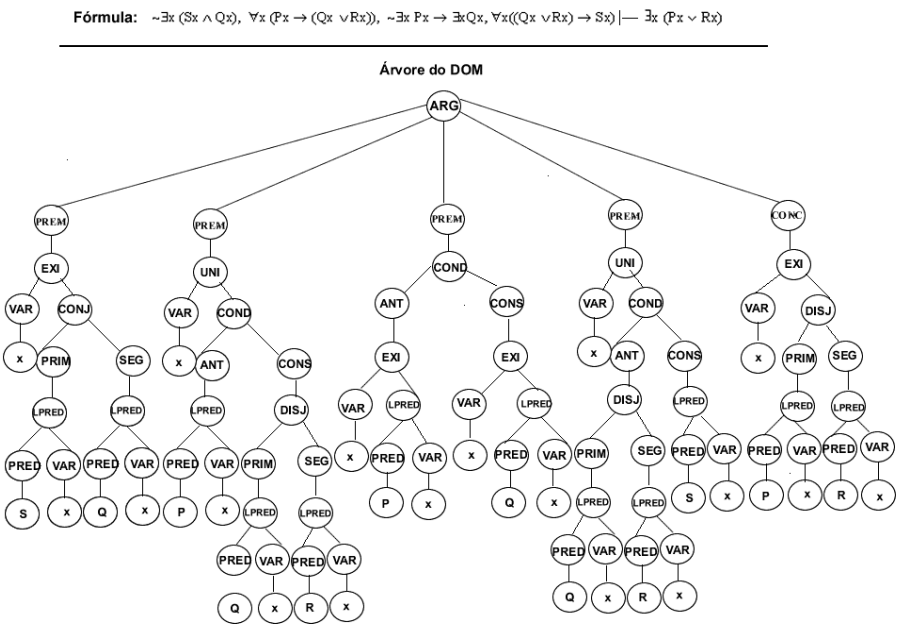

Figura 29: Fórmula gerada a partir da utilização da API DOM

Na classe **ordenacaoClass** estão implementados os métodos responsáveis pela aplicação lógica da seção 4.2.1. Na ordenação inicial, cada situação definida como critério de ordenação é verificada e os elementos são armazenados em vetores específicos. A junção desses vetores forma o **vetorBase** que, como resultado da agregação dos vetores referentes a cada situação de ordenação, já é criado ordenado. Esta situação ocorre quando o argumento que está sendo

analisado é composto por uma ou mais premissas e uma conclusão. Quando o argumento for um teorema, ou seja, composto apenas pela conclusão, essa ordenação inicial não é necessária, visto que se trata apenas de uma única fórmula. Assim, a ordenação será realizada somente entre os elementos internos do teorema.

Na figura 30 é apresentado o método **ordenacaoClass** que recebe como parâmetro a árvore gerada pela API DOM (Document doc) e o vetor base. Assim, é criado um atributo lista do tipo **NodeList** (capaz de armazenar uma coleção de nós a partir de um nó especificado) identificando que a ordenação será iniciada pelos elementos **PREM** (premissa) e **CONC** (conclusão). Desta forma, são enviadas para o método **ordenacao** duas situações distintas, ou seja, a lista e o parâmetro referente a negação vazio e a lista e o parâmetro referente a negação com uma negação "~" (no método dos Tableaux, para a verificação da validade de uma fórmula, nega-se a conclusão e busca-se encontrar a contradição em todos os ramos que compõem a árvore).

```
1  public ordenacaoClass(Document doc,Vector vetorBase) throws Exception
2  {
3          NodeList lista = doc.getElementsByTagName("PREM");
4          ordenacao(lista,"");
5          lista = doc.getElementsByTagName("CONC");
6          ordenacao(lista,"~");
7          montar_vetorBase(vetorBase);
8  }
```

Figura 30: Código do método ordenacaoClass

No método **ordenacao** (figura 31), são extraídos dos elementos da lista iniciada por **PREM** e/ou por **CONC** a negação (armazenada em **strNeg** – essa situação ocorre apenas para a conclusão), o nome do elemento (ex. conjunção – CONJ, condicional – COND, armazenado em **nomeElemento**) e o identificador do elemento (armazenado em **id**). Para a separação de cada um dos elementos, exceto a negação da conclusão, são utilizados métodos da API DOM, como o método para capturar o nome do filho de um determinado nó (figura 31, linha 8) e o método

utilizado para capturar o valor armazenado num determinado atributo do filho de um elemento (figura 31, linha 9).

É necessário estar definida, também, a quantidade de quantificadores existenciais externos e internos presentes em cada fórmula, pois este é o principal critério da ordenação, capaz de diminuir consideravelmente o número de instanciações dos elementos. Para verificar a quantidade dos quantificadores existenciais, são utilizados os métodos **contador_exi** e **contador_exi_interno**, respectivamente. Caso a fórmula não tenha quantificadores e o seu elemento mais externo tenha o atributo da negação e não seja conclusão do argumento, então é necessário verificar quantas negações ocorrem no elemento; caso esse número seja par, não é atribuído valor para **strNeg**; caso seja ímpar, uma negação é atribuída a **strNeg** (Figura 31, linha 18). Isso acontece porque quando há ocorrência de negações em quantidades pares, os elementos voltam a ter o mesmo estado que tinham antes das negações (ex.: ~ ~ P ≡ P). A implementação desta forma de ordenação obteve um resultado mais eficiente, num curto período de tempo, pela existência de métodos para manipulação da árvore gerada a partir do *parser* da API DOM. A ordenação, assim, não levou em consideração somente o primeiro elemento externo de uma fórmula, mas trabalhou com todos os elementos que a compõem.

```
1  protected void ordenacao(NodeList lista, String negConc)
2  {
3          for (int i = 0; i <  lista.getLength(); i++)
4          {
5                  if (lista.item(i).getChildNodes().item(1).getNodeType() == Node.ELEMENT_NODE)
6                  {
7                          String strNeg = negConc;
8                          String nomeElemento = lista.item(i).getChildNodes().item(1).getNodeName();
9                          String id = lista.item(i).getChildNodes().item(1).getAttributes().getNamedItem("ID").getNodeValue();
10
11                         int exiE = contador_exi(lista.item(i).getChildNodes(), negConc,"", 0);
12                         int exiI = contador_exi_interno(lista.item(i).getChildNodes(), negConc);
13
14
15                         if (lista.item(i).getChildNodes().item(1).hasAttributes() && negConc == "" && exiI == 0 && exiE == 0)
16                                 if ((lista.item(i).getChildNodes().item(1).getAttributes().item(1).toString().length()>=7) &&
17                                    (lista.item(i).getChildNodes().item(1).getAttributes().item(1).toString().length() % 2 != 0))
18                                        strNeg="~";
19                         Vector vHist = new Vector(3, 1);
20                         ordenacao_distribuicao(lista.item(i).getChildNodes(), nomeElemento, strNeg, id, exiE, exiI, vHist);
21                 }
22         }
23 }
```

Figura 31: Implementação do método ordenação

Um dos pontos observados para a ordenação é a verificação das equivalências das fórmulas citadas na seção 4.2.1. É necessário considerar os elementos como eles serão trabalhados a partir do método dos tableaux. Assim, se o elemento é um condicional, por exemplo, e tem como antecedente um elemento agregado ao quantificador universal, este quantificador universal deve ser considerado no somatório dos quantificadores existenciais, pois o condicional é equivalente a uma disjunção com o primeiro elemento negado.

Na figura 32 é apresentada a implementação do método **contador_exi**, responsável por verificar a quantidade de quantificadores existenciais externos a uma fórmula. Este método recebe como parâmetro uma lista de elementos a partir de um determinado nó, a negação da conclusão, o tipo do elemento (ex.: CONJ, BIC) e o atributo que guardará a quantidade de quantificadores existenciais (**cont**). Desta forma, é possível executar quatro ações neste método:

- verificar que tipo de elemento está sendo tratado – caso seja um condicional, por exemplo, é acrescentada uma negação ao termo antecedente do elemento (linhas 8-19);
- verificar se o elemento que está sendo tratado tem um atributo; caso tenha, chamar o método **verificar_neg** para efetuar a adição das negações do elemento, verificando, em seguida, se a partir desta adição é formado um número ímpar ou par de negações. Assim, é retornada a negação, caso o número seja ímpar, ou é retornado espaço em branco, caso contrário (linhas 25-27);
- verificar se o elemento é um quantificador universal (UNI); caso seja e a negação existir, então o contador da quantidade de quantificadores existenciais da fórmula é incrementado, caso contrário, nada é adicionado (linhas 28-34).
- verificar se o elemento é um quantificador existencial (EXI): caso seja, faz uma situação inversa à anterior, ou seja, adiciona mais um ao contador se não existir negação na existência (linhas 35-41).

```
1   protected int contador_exi(NodeList lista, String negConclusao, String tipo, int cont)
2   {
3               int i = 0;
4               String neg = negConclusao;
5               String elemento = "";
6
7               // Verificando as negações para fazer as equivalências: ~E = U e ~U = E
8               if (tipo != "" && tipo.indexOf("BIC") < 0)
9                       elemento = tipo.substring(1,tipo.length());
10              if (elemento.compareTo("COND") == 0)
11                      if (tipo.substring(0,1).compareTo("1") == 0)
12                              neg = neg + "~";
13              if (elemento.compareTo("~COND") == 0)
14                      if (tipo.substring(0,1).compareTo("2") == 0)
15                              neg = neg + "~";
16              if (elemento.compareTo("~DISJ") == 0 || elemento.compareTo("~CONJ") == 0)
17                      neg = neg + "~";
18              if (tipo.indexOf("BIC") > 0 && tipo.substring(0,1).compareTo("2")==0)
19                      neg = neg + "~";
20
21              while (i < lista.getLength())
22              {
23                      if (lista.item(i).getNodeType() == Node.ELEMENT_NODE)
24                      {
25                              if (lista.item(i).hasAttributes())
26                                      neg = verificar_neg(neg +
27                                              lista.item(i).getAttributes().getNamedItem("NEG").getNodeValue());
28                                      if (lista.item(i).getNodeName().compareTo("UNI") == 0)
29                                      {
30                                              if (neg.compareTo("") != 0)
31                                                 cont = cont + 1;
32                                              lista = lista.item(i).getChildNodes();
33                                              i = 0;
34                                      }
35                                      if (lista.item(i).getNodeName().compareTo("EXI") == 0)
36                                      {
37                                              if (neg.compareTo("") == 0)
38                                              cont = cont + 1;
39                                              lista = lista.item(i).getChildNodes();
40                                              i = 0;
41                                      }
42                      }
43                      i = i + 1;
44              }
45              return cont;
46  }
```

Figura 32: Implementação do método contador_exi

O método **contador_exi_interno** segue a mesma lógica do método **contador_exi**, no entanto, busca especificamente os elementos internos da fórmula. Neste sentido, enquanto o elemento da lista for um quantificador externo, a lista passa a apontar para o seu elemento filho e a comparação é novamente realizada (Figura 33, linhas 1-6). Assim, quando não mais existirem quantificadores externos é verificado qual operador está sendo tratado. Para cada operador é chamado o método **contador_exi**, sendo enviados como parâmetros: os elementos filhos do operador que está sendo tratado (Ex: os elementos filhos de ANT, CONS, PRIM, SEG); o espaço vazio referente a negação da conclusão; o valor “1” ou “2” indicando se tratar do elemento PRIM ou SEG (ou ANT e CONS), respectivamente, a negação agregada ao tipo do elemento (BIC, CONJ, COND etc) e o contador (**cont** – responsável por armazenar a quantidade de quantificadores existenciais internos) (linhas 13-19).

```
1     if (lista.item(i).getNodeName().compareTo("EXI") == 0 || lista.item(i).getNodeName().compareTo("UNI") == 0)
2     {
3                 if (lista.item(i).hasAttributes())
4                         neg = verificar_neg(neg + lista.item(i).getAttributes().getNamedItem("NEG").getNodeValue());
5                 lista = lista.item(i).getChildNodes();
6                 i = 0;
7     }
8     else
9     {
10                String tipo = lista.item(i).getNodeName();
11                if (lista.item(i).hasAttributes())
12                        neg = verificar_neg(neg + lista.item(i).getAttributes().getNamedItem("NEG").getNodeValue());
13                if (tipo.compareTo("COND") == 0 || tipo.compareTo("CONJ") == 0 || tipo.compareTo("DISJ") == 0)
14                {
15                        // Conta os existênciais internos do ANT ou PRIM
16                        cont = contador_exi(lista.item(i).getChildNodes().item(1).getChildNodes(),"","1"+neg+tipo,0);
17                        // Conta os existênciais do CONS ou SEG
18                        cont = contador_exi(lista.item(i).getChildNodes().item(3).getChildNodes(),"","2"+neg+tipo,cont);
19                }
```

Figura 33: Parte do código do método contador_exi_interno

A partir da agregação dessas informações, as fórmulas que compõem o argumento são distribuídas em vetores específicos para cada situação de ordenação. Conforme pode ser observado na figura 34, as fórmulas atômicas com aridade zero são inseridas no vetor **p1** (linhas 6-10) e os elementos que possuírem uma maior quantidade de quantificadores existenciais externos são inseridos no vetor **p2** (linhas 12-19). Essas situações são verificadas, também, para a questão dos conectivos lógicos. Assim, quando ocorrer a agregação desses vetores para a formação do vetor base, este já é formado ordenado.

```
1     protected void ordenacao_distribuicao(NodeList listaTemp, String nomeElemento, String strNeg, String id, int exiE, int exiI, Vector
2     vHist)
3     {
4       boolean tratou = false;
5        // Primeiro Critério
6       if (nomeElemento.compareTo("LPRED") == 0 && !tratou && vHist.size() == 0)
7       {
8                 p1.addElement(montar_dados(strNeg, nomeElemento, id, "0", "0", vHist));
9                 tratou = true;
10       }
11      // Segundo Critério
12      if (exiE >= exiI && exiE != 0 && exiI != 0 && !tratou)
13      {
14                if (p2.size() == 0)
15                   p2.addElement(montar_dados(strNeg, nomeElemento, id, Integer.toString(exiE), Integer.toString(exiI), vHist));
16                else
17                   ordenar_vetor(p2,montar_dados(strNeg, nomeElemento, id, Integer.toString(exiE), Integer.toString(exiI), vHist), 3);
18                tratou = true;
19       }
```

Figura 34: Parte do código do método ordenacao_distribuicao

Desta forma, a ordenação inicial dos elementos é estabelecida e em cada situação que tiver a ocorrência de um dos conectivos binários que não utilizam a bifurcação de ramo, o método de **ordenacao_distribuicao** é chamado e a ordenação realizada. Evita-se, desta forma, as repetições que ocorrem em ramos bifurcados e utiliza-se menos letras nominais no ramo. Com isso, é possível executar

a resolução de uma fórmula utilizando uma quantidade inferior de nós na árvore de prova.

5.2 Instanciação das variáveis

Para o entendimento do processo de instanciação das variáveis é necessário ter uma visão das etapas pelas quais uma fórmula passa até o momento da instanciação. Na figura 35 é apresentada uma estrutura com a demonstração das etapas de prova. Conforme pode ser verificado, a partir da árvore gerada no DOM é realizado o processo de ordenação dos elementos. Estando, então, os elementos no vetor base, ocorre a ordenação interna de elementos ligados a conectivos binários e o processo de instanciação é realizado em cada fórmula, na ordem em que estas foram armazenadas.

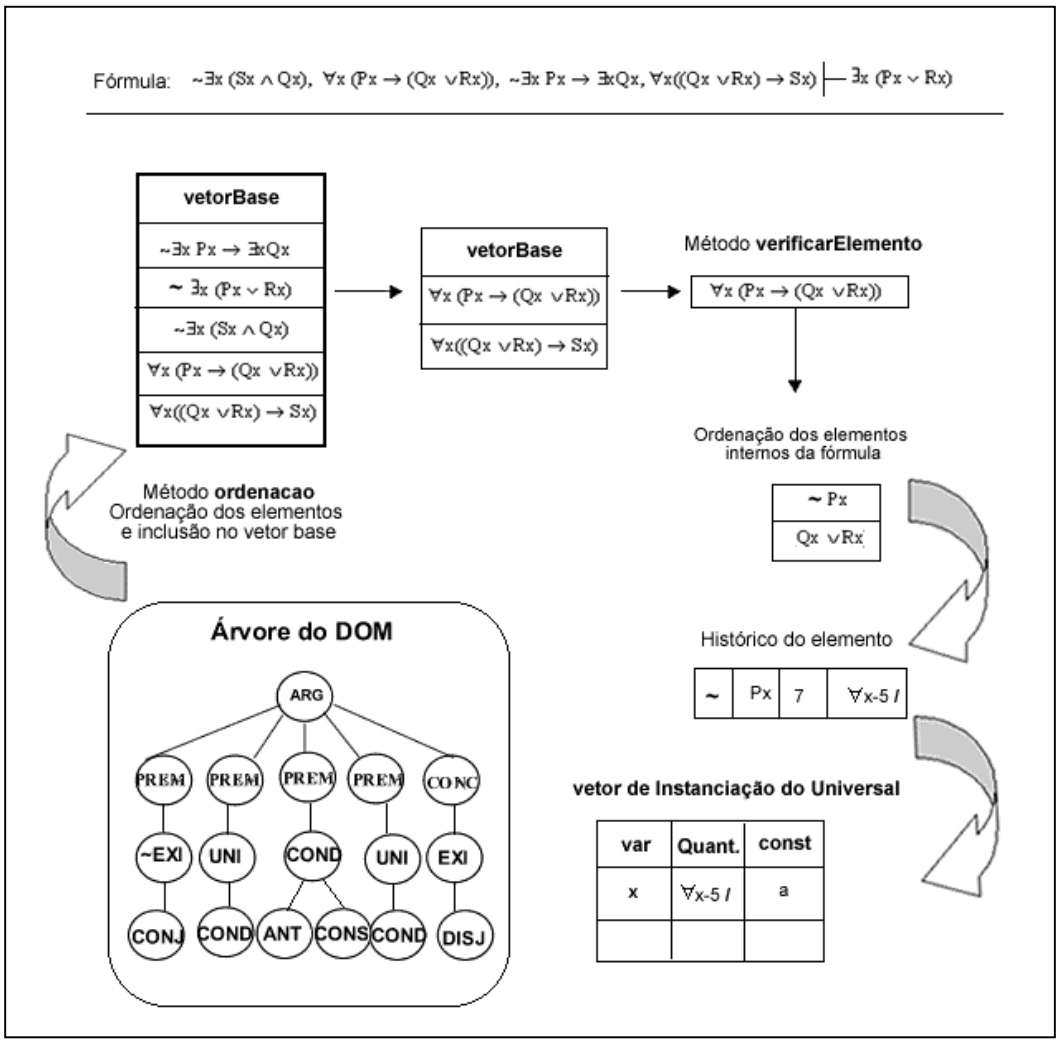

Figura 35: Processo de tratamento das fórmulas

Um dos pontos básicos que devem ser tratados no processo supracitado é a criação do histórico do elemento. Assim, no momento em que é invocado o método **verificarElemento**, é iniciado o processo de identificação do elemento. Primeiro, é verificado se o elemento é um sinal predicativo, um bicondicional ou um dos conectivos binários. Para cada item, um conjunto de situações é realizado, ou seja, sendo um sinal predicativo não há necessidade de executar nova ordenação, são apenas armazenadas as informações que compõem o histórico, realizada a instanciação e inserção na árvore de prova. Caso seja um dos conectivos binários (exceto o bicondicional), os elementos são ordenados antes de ser realizada a instanciação e a conseqüente inserção na árvore de prova. Para o bicondicional, devido a sua estrutura distinta, foram definidos alguns métodos específicos para o tratamento dos seus elementos. Estes métodos serão vistos com mais detalhes na seção 5.4.

A inserção dos elementos no vetor do histórico é realizada a partir da invocação de alguns métodos (Ex1.: **pegar_filhosEU** - os elementos filhos de fórmulas que tem os quantificadores como elemento mais externos. Ex.2.: **pegar_filhos** – os elementos filhos de fórmulas cujo operadores mais externos é um dos conectivos binários). A questão básica é a inclusão da negação (se houver), do sinal predicativo (agregado a sua variável – isso se não for uma fórmula atômica de aridade zero) e do seu identificador, do quantificador referente à variável que acompanha o sinal predicativo, do identificador do quantificador (pois, podem haver quantificadores agregados a variáveis com o mesmo nome, mas que representam situações distintas). Pode ser visto na figura 36 parte do código do método **pegar_filhosEU**. Nas linhas 9 a 12 pode ser observada a formação de parte do histórico de um determinado elemento, ou seja: na linha 9 é incluído no atributo **neg**, a negação (se existir); nas linhas 10 a 12 são invocados os métodos da API DOM responsáveis por buscar o quantificador na árvore DOM, a variável agregada a ele e o seu identificador.

```
1     while (j < listaTemp.getLength() && !parar)
2     {
3        if (listaTemp.item(j).getNodeType() == Node.ELEMENT_NODE)
4        {
5           if (listaTemp.item(j).getNodeName().compareTo("EXI") == 0 || listaTemp.item(j).getNodeName().compareTo("UNI") == 0)
6           {
7              if (listaTemp.item(j).hasAttributes())
8                 {
9                    neg = verificar_neg(neg + listaTemp.item(j).getAttributes().getNamedItem("NEG").getNodeValue());
10                   vHist.addElement(neg + listaTemp.item(j).getNodeName().toString().substring(0,1) +
11                   listaTemp.item(j).getChildNodes().item(1).getFirstChild().getNodeValue() + "-" +
12                   istaTemp.item(j).getAttributes().getNamedItem("ID").getNodeValue().toString());
13                }
```

Figura 36: Parte do código do método pegar_filhosEU

Na instanciação das variáveis, é necessário verificar a qual quantificador o elemento está agregado, para indicar que tipo de instanciação deverá ser realizada. Para isso, o método **verificarElemento** invoca o método **fazer_instanciacao** enviando como parâmetro os elementos da fórmula que devem ser instanciados, os nós folhas, e a informação para a definição de que a instanciação da fórmula está sendo realizada pela primeira vez, ou se está ocorrendo a partir de uma operação de retorno à fórmula.

No método **fazer_instanciacao** é realizada a chamada dos métodos responsáveis pela instanciação específica de cada quantificador. Desta forma, todos os elementos necessários para a instanciação são armazenados e enviados aos métodos. Nas seções seguintes será explicado o método de instanciação do quantificador universal e existencial.

5.2.1 Instanciação do quantificador existencial

Para o método de instanciação das variáveis agregadas ao quantificador existencial são enviados os seguintes elementos:

- **nosFolhas:** armazena os nós folhas da árvore de prova;
- **vInstanciacao_exi:** armazena os elementos instanciados, seguindo a estrutura: variável, quantificador ligado a variável e ao identificador, constante utilizada no processo de instanciação;
- **vTemp:** armazena o elemento que possui a variável que será instanciada, pois, é instanciada uma variável por vez;

- **vHist:** armazena todos os quantificadores existenciais presentes na fórmula, juntamente com a variável e o identificador;
- **vHist.elementAt(j).toString( ):** armazena o quantificador que está sendo tratado;
- **strElemento:** armazena o sinal predicativo que está sendo tratado juntamente com as variáveis agregadas a ele;
- **var:** armazena a variável do elemento que está sendo tratado;
- **strElemento.substring(strElemento.indexOf("~")+2, strElemento.length( )):** armazena as variáveis que estão sendo tratadas, pois, de acordo com o método, serão enviados os elementos que estão na *substring* de **strElemento**. Um conteúdo possível para **strElemento** seria **~Mxxx**, assim, as variáveis sempre iniciam a partir da segunda posição da *string* e terminam na última posição de **strElemento**.

```
1   String strLN = objTemp.buscar_LN(doc);
2   String strConst2 = obj.analisar_ramoExi(nosFolhas);
3   if (strLN != "")
4    {
5      if (strConst2 == "")
6           strConst2 = strLN;
7      else
8           if (Character.getNumericValue(strConst2.charAt(0)) < Character.getNumericValue(strLN.charAt(0)))
9                strConst2 = strLN;
10   }
11   if (strConst2 != "")
12   {
13      if (strConst2.indexOf("s") >= 0)
14        {
15           if (strConst2.length() > 1)
16                strConst2 = "s" + String.valueOf(Integer.parseInt(strConst2.substring(1, strConst2.length())) + 1);
17           else
18                strConst2 = "s1";
19         }
20        else
21             strConst2 = String.valueOf(Character.forDigit(Character.getNumericValue(strConst2.charAt(0)) + 1,20));
22   }
23   else
24      if (strConst.length() > 1 && !ver_limite_variaveis(strConst.charAt(0)))
25           strConst2 = String.valueOf(Character.forDigit(Character.getNumericValue(strConst.charAt(0)) + 1,20));
```

Figura 37: Parte do código do método fazer_instanciacao_exi (parte 1)

A partir dos elementos enviados é possível, no método **fazer_instanciacao_exi**, substituir as variáveis por constantes seguindo os critérios estabelecidos para a instanciação do existencial. Como pode ser observado na figura 37, inicialmente é necessário verificar na árvore gerada pelo *parser* da API DOM se há a ocorrência de uma constante na fórmula (linha 1). Depois é verificado nos ramos da árvore de prova, em que a instanciação irá ocorrer, qual a última constante trabalhada (linha 2). Com essas duas informações, pode-se armazenar no atributo **strConst2**, a constante que deverá ser utilizada na instanciação.

Inicialmente, faz-se uma comparação para constatar se existe constante previamente na fórmula que será instanciada; caso exista e **strConst2** esteja vazia, esta passa a armazenar o valor de **strLN**, ou seja, a constante já existente na fórmula geral do argumento passa a ser a maior constante considerada para a realização da instanciação. Se já existir informação armazenada em **strConst2** então é analisada qual constante é a maior letra e armazenada aquela que retornar a maior letra nominal (linhas 3-10). Nas linhas 11 a 22 é analisado se existe elemento em **strConst2**; caso exista, são verificadas as situações em que é usado um grande número de constantes, ultrapassando o limite estabelecido para as letras nominais (a-s); assim, a partir do momento em que é alcançada a letra s, as variáveis passam a ser substituídas por letras acompanhadas por um número (está sendo utilizada a letra "s", seguindo a seqüência "s_1, s_2, ..., s_n"). Caso o limite não tenha sido alcançado, a variável é substituída pela próxima letra a partir da constante armazenada em **strConst2** (linha 21). Na linha 22, é realizada uma comparação que tem como condições o fato de existir mais de um elemento armazenado em **strConst** e deste atributo ainda conter variáveis, assim, se não existir valor em **strConst2**, este atributo passa a armazenar a próxima letra a partir da constante armazenada em **strConst**. Esta é uma forma de manter a instanciação correta em casos em que há mais de uma variável agregada ao mesmo sinal predicativo.

Na figura 38 é apresentado parte do código para a instanciação final das variáveis quantificadas existencialmente. Com a constante correta para a instanciação armazenada em **strConst2**, os passos seguintes podem ser executados:

- Verificação da existência de elementos no vetor de instanciação (linha 3); caso não existam elementos, existem dois caminhos a seguir: adicionar ao vetor temporário (**vTemp**) a constante armazenada em **strConst2** (caso exista algum elemento), ou adicionar ao **vTemp** a constante "a" (linhas 4-10).
- Caso existam elementos no vetor de instanciação do existencial, deve ser analisado cada elemento do vetor até que seja encontrado um elemento igual ao elemento que está armazenado no atributo **vHist** (responsável por armazenar o

quantificador cuja variável sofrerá a instanciação) ou até chegar ao final do vetor (linhas 16-24).

- Se encontrar um elemento no vetor de instanciação que coincida com o elemento que está sendo tratado, então é armazenada em **strConst2** a constante de instanciação do elemento e adicionada em **vTemp** (linhas 26-34).
- Não encontrando um elemento idêntico é verificado se o último elemento do vetor de instanciação está contido no vetor de históricos (**Hist**) do elemento, ou seja, aquele que contém todos os quantificadores que estão ligados ao elemento em questão (que apresenta não apenas o quantificador com o mesmo histórico do elemento, mas, aqueles que são externos ao elemento) (linha 37). Encontrando essa situação, é analisado se o quantificador do elemento que está sendo tratado tem a barra (/) indicadora de que o quantificador é externo ao elemento e também é verificado se o último elemento que está no vetor de instanciação do existencial é externo ao elemento (linhas 39, 40). Caso seja positiva essa verificação, então é inserida no vetor temporário (**vTemp**) a próxima letra da constante do último elemento armazenado no vetor de instanciação do existencial (linha 42). Caso a comparação não seja válida, isso significa que o elemento que está sendo tratado não tem ligação alguma com os elementos do vetor de instanciação, sendo assim, é acrescida a constante armazenada em **strVar**. Se não houver nenhuma constante armazenada, é atribuído a **vTemp** a letra nominal “a” (indicando, assim, que este é o primeiro elemento instanciado no(s) ramo(s) que sofrerá(ão) a inclusão do elemento) (linhas 45-50).
- O método é encerrado com o retorno do elemento que está no atributo **strElemento**, que é a junção do sinal predicativo que está sendo tratado com as suas variáveis já instanciadas e com aquelas que ainda serão instanciadas numa nova etapa (se existirem) (linha 57).
- O valor de **strElemento** é retornado ao método **fazer_instanciacao** que foi invocado pelo método **verificar_elemento**. Logo, é este método que analisa o tipo do elemento e invoca o método correto para a inserção deste elemento na árvore de prova. Para a execução deste processo, é invocado o método **distribuir_elementos**, que a partir do tipo de elemento que foi enviado como parâmetro (DISJ, CONJ etc) utiliza um dos métodos responsáveis por invocar o método **inserir_elementos** da classe **arvoreClass**. Esses métodos estabelecem

o tipo de inserção correta na árvore de prova, pois se o elemento for uma disjunção, por exemplo, é chamado o método **bifucar_caminho**, se for uma conjunção, o método em questão seria **caminho_sequencial**.

```
1    if (strElemento.indexOf(var) >= 0)
2    {
3       if (vInstanciacao.isEmpty())
4         {
5                if (strConst2 != "")
6                   vTemp.addElement(strConst2);
7                else
8                   vTemp.addElement("a");
9                vInstanciacao.addElement(vTemp);
10        }
11        else
12        {
13               Vector vUlt = new Vector(3,1);
14               int x = 0;
15               boolean achou = false;
16               while(x < vInstanciacao.size() && !achou)
17               {
18                    vUlt.addAll((Vector)vInstanciacao.elementAt(x));
19                    if (vUlt.elementAt(1).toString().compareTo(vHist) == 0)
20                       achou = true;
21                    else
22                       ++x;
23                    vUlt.clear();
24                }
25                if (achou)
26               {
27                   vUlt.addAll((Vector)vInstanciacao.elementAt(x));
28                    strConst2 = vUlt.elementAt(2).toString();
29                    if (Character.getNumericValue(vUlt.elementAt(2).toString().charAt(0)) >
30                        Character.getNumericValue(strConst2.charAt(0)))
31                            vTemp.addElement(vUlt.elementAt(2).toString());
32                    else
33                        vTemp.addElement(strConst2);
34                }
35               else
36               {
37                    if (Hist.indexOf(((Vector)vInstanciacao.lastElement()).elementAt(1).toString()) >= 0)
38                    {
39                        if (vHist.indexOf("/") >=0 &&
40                           ((Vector)vInstanciacao.lastElement()).elementAt(1).toString().indexOf("/") >= 0)
41                              vTemp.addElement(String.valueOf(Character.forDigit(Character.getNumericValue
42                                (((Vector)vInstanciacao.lastElement()).elementAt(2).toString().charAt(0)) + 1, 20)));
43                    }
44                    else
45                    {
46                        if (strConst2 != "")
47                            vTemp.addElement(strConst2);
48                       else
49                            vTemp.addElement("a");
50                    }
51                            vInstanciacao.addElement(vTemp);
52               }
53        }
54               strElemento = strElemento.substring(0,strElemento.indexOf(var)) + vTemp.elementAt(2).toString() +
55                                strElemento.substring(strElemento.indexOf(var)+1,strElemento.length());
56   }
57   return strElemento;
```

Figura 38: parte do código do método fazer_instanciacao_exi (parte 2)

Na figura 39 é apresentado um resumo do processo de instanciação. O início do processo se dá com a ordenação dos elementos e inclusão no **vetorBase**, assim é possível tratar os elementos e enviar os dados necessários para que seja executada a instanciação. No método **fazer_instanciacao** é observado a existência de duas possibilidades, neste caso está sendo apresentada a instanciação de uma variável agregada a um quantificador existencial; assim, seguindo os critérios desta forma de instanciação, são realizadas as instanciações em todas as variáveis

agregadas ao elemento. E apenas depois de finalizada a instanciação é que os elementos podem ser inseridos na árvore de prova.

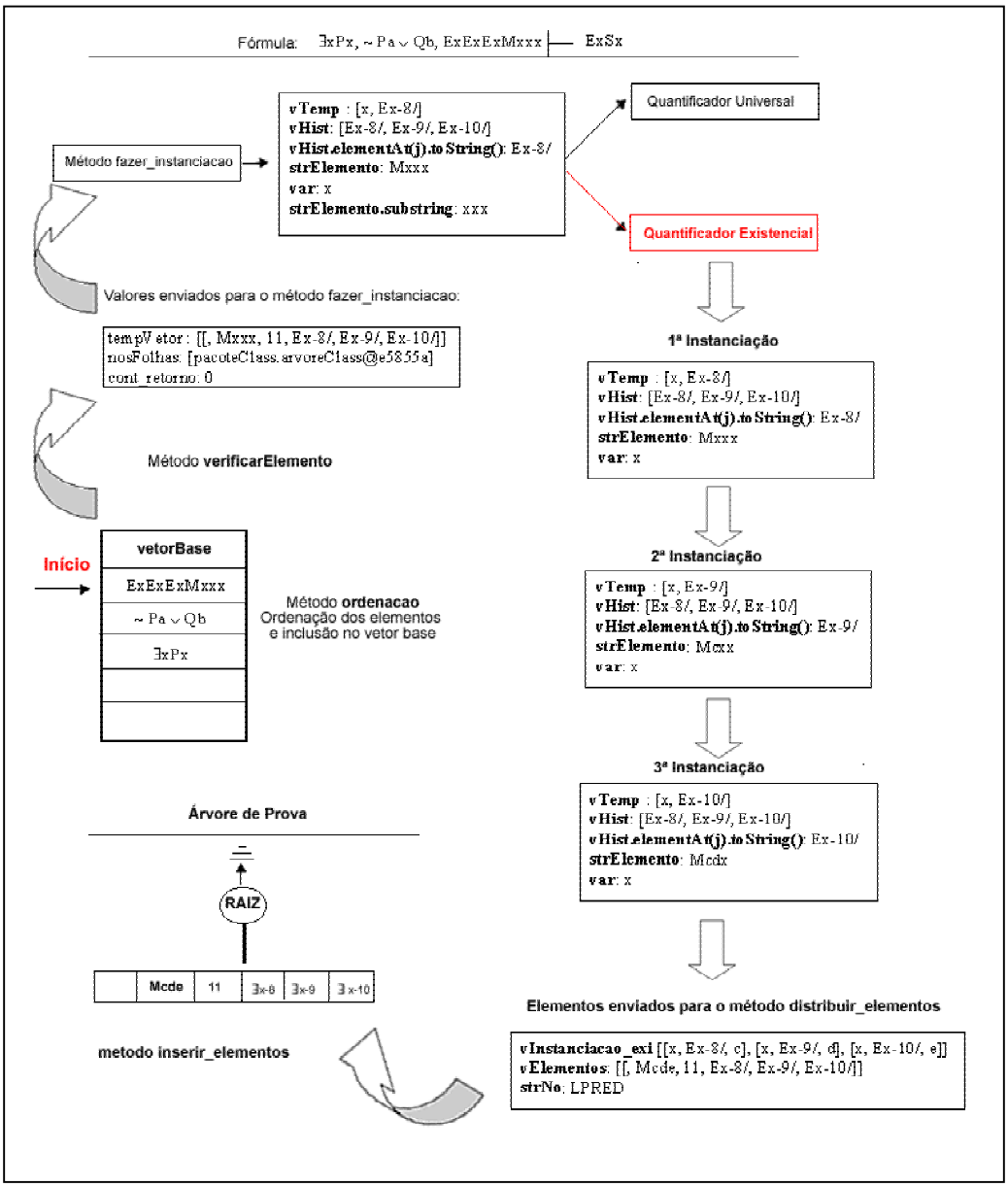

Figura 39: Esquema de instanciação das variáveis existenciais

5.2.2 Instanciação do quantificador universal

Para o estabelecimento da instanciação “inteligente” das variáveis universais foi implementado o método **fazer_instanciacao_uni** considerando a importância da adaptação da constante para o fechamento dos ramos. Para isso, neste método, antes de qualquer análise, é invocado o método **analisar_ramoUni** da classe

arvoreClass. Neste método é realizada a análise dos elementos do(s) ramo(s) em que o elemento será inserido.

Conforme pode ser visto no código apresentado na figura 40, é criado o vetor **vContrad** (linha 3) em que serão armazenadas as letras que tornam possível a contradição entre elementos num ramo. Assim, é verificado a partir de cada nó folha da árvore da árvore de prova se há um elemento contraditório ao elemento enviado como parâmetro (**str**). Para isso, é necessário analisar o elemento a partir de cada nó folha e verificar se existe um elemento que tenha o mesmo sinal predicativo, mas, que exista a negação em, necessariamente, apenas um deles. Esta verificação é possível, pois quando é invocado o método **analisar_neg**, o elemento enviado em **strPred** retorna o sinal predicativo sem a negação (se for negado) ou o contrário. Desta forma, é correto afirmar que se os elementos comparados forem iguais, eles, na verdade, são contraditórios (linha 16). Caso tenham a mesma quantidade de variável, é invocado o método **retornar_var** para armazenar em **var** a constante que está sendo utilizado no predicado contraditório ao elemento tratado (linhas 27 ou 38). Depois é armazenado no vetor **vContrad** a letra nominal encontrada e quantas vezes essa letra retorna uma contradição (linha 29 ou 40). Com o método **vetor_contradicoes_maior** é retornado ao método **intanciacao_uni** a letra nominal que melhor se adequa a árvore de prova, ou seja, aquela que permite um maior número de contradições (linha 49).

```
1       public String analisar_ramoUni(Vector nosFolhas, String str, int pos)
2       {
3           Vector vContrad = new Vector(3,1);
4           String strPred = "";
5           String strVar = "";
6           String var = "";
7           strPred = str.substring(0, str.indexOf("~") + 2);
8           strVar = str.substring(str.indexOf("~") + 2, str.length());
9           for (int i = 0; i < nosFolhas.size(); i++)
10          {
11              arvoreClass no = (arvoreClass)nosFolhas.elementAt(i);
12              boolean achou = false;
13             while (no != null)
14             {
15                 String strElemento = no.vHist.elementAt(0).toString() + no.vHist.elementAt(1).toString();
16                  if (strElemento.substring(0, strElemento.indexOf("~") + 2).compareTo(analisar_neg(strPred)) == 0)
17                  {
18                      if (strElemento.indexOf("CONJ") < 0 && strElemento.indexOf("DISJ") < 0 &&
19                          strElemento.indexOf("COND") < 0 && strElemento.indexOf("BIC") < 0)
20                      {
21                          if (strElemento.indexOf("s") >= 0)
22                          {
24                              If (contar_var(strElemento.substring(strElemento.indexOf("~") + 2, strElemento.length()), 's') ==
25                                  contar_var(strVar, 's'))
26                              {
27                                  var  = retornar_var(strElemento.substring(strElemento.indexOf("~") + 2,
28                                                      strElemento.length()), 's', pos);
29                                  vetor_contradicoes(vContrad, var);
30                                  achou = true;
31                               }
32                          }
33                          else
34                          {
35                             if (strElemento.substring(strElemento.indexOf("~") + 2, strElemento.length()).length() ==
36                                contar_var(strVar,'z'))
37                             {
38                                  var = strElemento.substring(strElemento.indexOf("~") + 2 + pos,
39                                          strElemento.indexOf("~") + 3 + pos);
40                                  vetor_contradicoes(vContrad, var);
41                                 achou = true;
42                              }
43                           }
44                          }
45                      }
46                  no = no.noPai;
47                 }
48             }
49          return vetor_contradicoes_maior(vContrad);
50      }
```

Figura 40: Código do método analisar_ramoUni (arvoreClass)

Com o retorno da letra nominal adequada à instanciação ou o retorno do espaço indicando que nenhuma letra foi encontrada que provocasse uma contradição no ramo, são realizadas as comparações no método **fazer_instanciacao_uni**:

- Caso o vetor de instanciação do universal esteja vazio, é verificado se alguma constante foi retornada do método **analisar_ramoUni**. Caso exista uma letra, então é adicionada esta letra ao vetor **vTemp** (enviado pelo método **fazer_instanciacao**). Caso contrário é adicionada a letra “a”.
- Se o vetor de instanciação do universal não estiver vazio, é verificado se há algum elemento que seja idêntico ao elemento que está sendo tratado.

Se tiver, é adicionada a letra que foi usada na instanciação anterior, evitando, assim, discrepâncias na instanciação.

- A última situação considera o vetor de instanciação com elementos, mas sem encontrar semelhança com o elemento tratado. Desta forma, é inserido o elemento que foi retornado do método **analisar_ramoUni**, ou a letra “a”.

Assim, é construído o vetor de instanciação do universal, considerando que para fazer a instanciação é necessário verificar o sinal predicativo que pode resultar numa contradição e a qual constante tal sinal predicativo está agregado. Apenas depois de ter certeza da não existência de sinal predicativo que permita a contradição, é considerado utilizar a letra “a” como constante de instanciação.

Na figura 41 é apresentado o resumo do processo de instanciação de variáveis universais. O vetor é ordenado e em seguida é realizada a instanciação dos elementos. Nesse exemplo, é realizada a instanciação do segundo elemento da fórmula que é tratado como um universal (~∃P ≡ ∀~P). Desta forma, primeiro é verificado se há elemento na árvore de prova que possa ser contraditório ao elemento que está sendo tratado. Como no exemplo há um sinal predicativo contraditório ao sinal predicativo agregado às variáveis que serão instanciadas, então tais variáveis são instanciadas pela mesma constante deste elemento. Assim, o elemento é introduzido na árvore de prova e no ramo é estabelecida uma contradição.

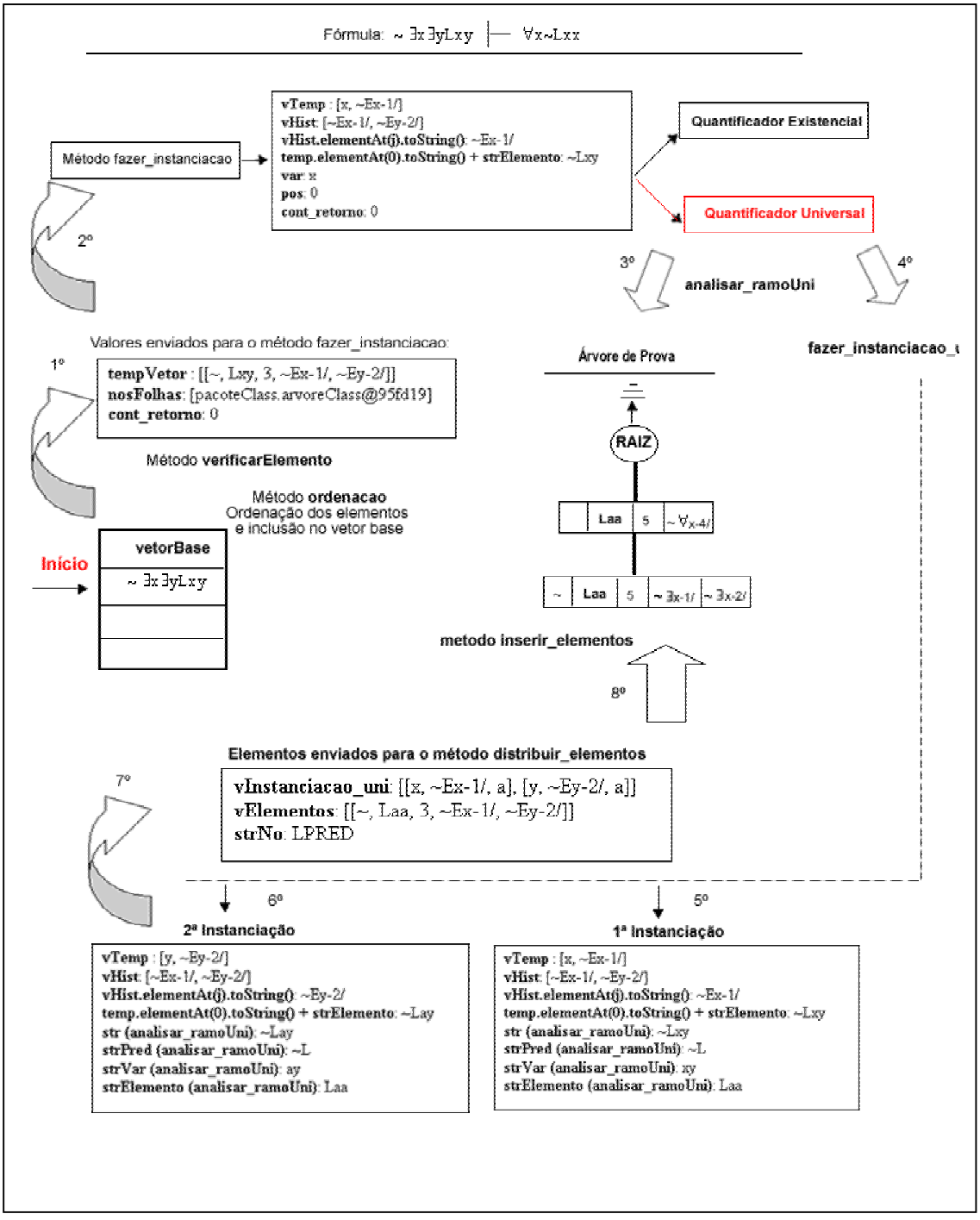

Figura 41: Esquema de instanciação das variáveis universais

5.3 Retorno às fórmulas

Depois de instanciadas todas as variáveis das fórmulas é verificado se na árvore de prova ficou algum ramo aberto, ou seja, um ramo sem elementos contraditórios. Para isso, no método **analisar_formula**, cada ramo é percorrido da raiz até os nós folhas. Se não forem encontrados elementos contraditórios no ramo, este é inserido no vetor **ramosFalhos** (Figura 42).

```
1     public void analisar_formula(arvoreClass obj, Vector vetorProva, Vector vHist, Vector ramosFalhos)
2     {
3         if (obj.centro == null && obj.esquerdo == null && obj.direito == null)
4         {
5             int i = 0;
6             boolean achou = false;
7             vetorProva.addElement(obj.vHist.elementAt(0).toString() + obj.vHist.elementAt(1).toString());
8              vHist.addElement(obj);
9              while (i < vetorProva.size() && !achou)
10             {
11                 if ((i+1) <= vetorProva.size())
12                 {
13                       if (vetorProva.elementAt(i).toString().indexOf("BIC") < 0 &&
14                          vetorProva.elementAt(i).toString().indexOf("COND") < 0 &&
15
16                          vetorProva.elementAt(i).toString().indexOf("DISJ") < 0 &&
17                          vetorProva.elementAt(i).toString().indexOf("CONJ") < 0)
18                          if ((vetorProva.indexOf(analisar_neg(vetorProva.elementAt(i).toString()), i+1)) >= 0)
19                              achou = true;
20                  }
21                  ++i;
22              }
24              if (!achou)
25                  ramosFalhos.add(vHist);
          }
```

Figura 42: Parte do código do método analisar_formula

O passo seguinte é o tratamento dos ramos falhos (ou seja, ramos abertos), realizando, inicialmente, a etapa de ordenação dos elementos. Serão instanciados apenas elementos que possuem em seu histórico quantificadores universais. Isso não implica, necessariamente, que um determinado sinal predicativo tenha que estar agregado a um quantificador universal, mas, este tem que ser interno a alguma fórmula que tenha em sua composição o quantificador universal. Como cada elemento traz consigo um histórico é possível identificar a relação do elemento com os quantificadores que pertencem à fórmula na qual o elemento está inserido.

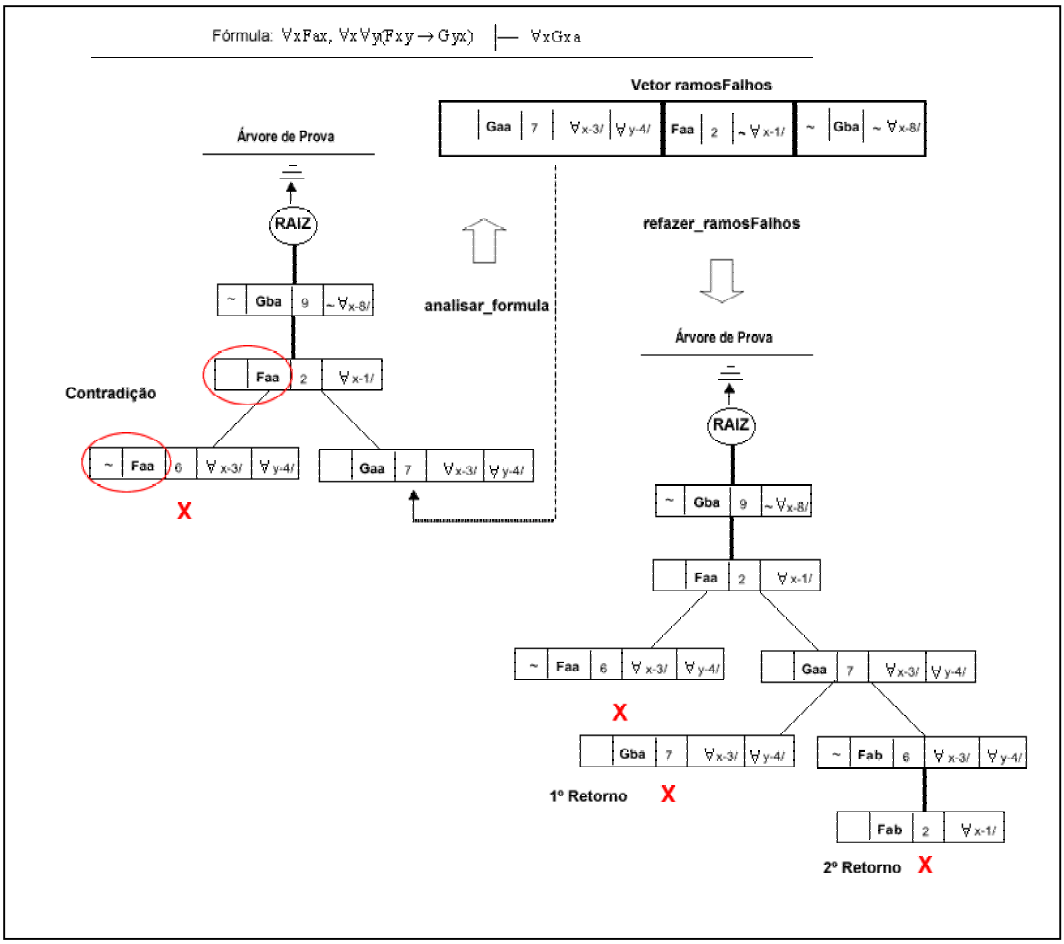

Figura 43: Esquema de retorno a ramos abertos

O tratamento do retorno às fórmulas é essencial para elementos cujas variáveis estavam instanciadas universalmente, pois são a partir destas variáveis que novas substituições podem ser realizadas e novas contradições encontradas. Assim, quando um elemento ligado a um conectivo binário é novamente instanciado, deve ser considerada a combinação para as duas partes que compõem o elemento, pois se isso não for realizado o programa pode repetir sempre a mesma instanciação e, desta forma, não ser possível a verificação da validade ou invalidade da fórmula. No exemplo apresentado na figura 43, pode ser verificada a ocorrência desta situação e só foi possível encontrar a contradição em todos os ramos da fórmula, pois no primeiro retorno o programa modificou a estratégia de substituição, ou seja, utilizou o conseqüente do condicional como fator de busca da contradição. Assim, o elemento antecedente (que está agregado aos mesmos quantificadores) teve que se adequar à instanciação ocorrida no termo conseqüente do condicional.

Para a situação de elementos que estão agregados a variáveis existenciais, não há necessidade de buscar adequação de constantes, apenas as variáveis são instanciadas pela próxima constante da maior constante encontrada no(s) ramo(s) que a instanciação ocorrer.

5.4 Etapas de tratamento do Bicondicional

Para o tratamento do bicondicional, podem ser utilizados dois métodos:

- Trabalhar com a sua equivalência, ou seja, vê-lo como a conjunção de dois condicionais. Sendo assim, seriam tratados 4 elementos distintos (o termo antecedente e conseqüente de dois condicionais).
- Ou tratá-lo diretamente: a relação entre um conectivo lógico binário e dois elementos. Isso dificulta a implementação, pois um mesmo elemento deve ser analisado de duas formas distintas. Como o bicondicional é formado por duas partes, positiva e negativa, a inclusão da negação nos termos provoca uma mudança na interpretação do vetor de instanciação: dois elementos com o mesmo identificador devem ser manipulados de forma distinta.

O bicondicional é formado por dois elementos (PRIM e SEG), estes elementos se tornam quatro na utilização da regra do bicondicional, ou seja, segundo o método dos tableaux, o caminho é bifurcado e o primeiro e segundo elementos são inseridos no ramo esquerdo e suas negações são inseridas seqüencialmente no ramo direito. Há uma variação nessa regra se o bicondicional estiver negado, quando isso ocorrer, a negação ocorre no segundo elemento do ramo esquerdo e no primeiro elemento do ramo direito.

Um ponto que deve ser considerado na implementação do bicondicional é a sua relação com os quantificadores:

- Quantificador externo ao bicondicional: as variáveis que estiverem agregadas aos sinais predicativos armazenados no primeiro ou segundo elemento do bicondicional devem ser instanciadas pela mesma constante.

Isso porque estas variáveis são relacionadas a um mesmo quantificador (com um mesmo identificador).

- Quantificador interno ao bicondicional: devem ser realizadas a equivalência dos quantificadores, nos casos dos termos que recebem a negação. Isso faz com que haja alteração na ordenação interna dos elementos, pois se o primeiro elemento for agregado a um universal e o segundo elemento tiver agregado a um universal negado (elemento negado depois da aplicação das regras do bicondicional), o segundo elemento deve ser considerado antes do primeiro na instanciação (respeitando a regra de economia de constantes).

O método **verificar_elemento** não trata da instanciação das variáveis referentes aos elementos do bicondicional, desta forma, são enviados como parâmetro para o método **distribuir_elementos** tanto os termos que compõem o bicondicional, como o conteúdo da *tag* que indica se tratar de um bicondicional (BIC, ~BIC) e os vetores de instanciação dos quantificadores existencial e universal. Assim, a partir do método distribuir_elemento, essas informações são direcionadas ao método **caminho_sequencial_BIC** (Figura 44). Neste método são realizadas as chamadas que tratarão os elementos do bicondicional, iniciando com a separação dos elementos para o estabelecimento da inserção correta na árvore de prova, ou seja, o primeiro elemento e sua cópia negada dão início a bifurcação dos elementos na árvore de prova. Assim, o segundo elemento e sua cópia negada seguem, respectivamente, o caminho definido pelo primeiro termo. Nos casos que ainda existem elementos complexos dentre os elementos enviados para o método **caminho_sequencial_BIC**, estes elementos são novamente tratados em **verificar_elemento** (linhas 22 a 29).

```
1	protected void caminho_sequencial_BIC(Vector nosFolhas, Vector vElementos, int tipo, int cont_retorno)
2	{
3		Vector tempVetor = new Vector(5,1);
4		Vector tempFolhas = new Vector(5,1);
5		Vector tempEsq = new Vector(1,1);
6		Vector tempDir = new Vector(1,1);
7		Vector temp = new Vector(1,1);
8		Vector vTemp = new Vector(1,1);
9		Vector vElementos_temp = new Vector(1,1);
10		montarVetor_BIC(vElementos, tipo);
11		vElementos_temp.addElement((Vector)vElementos.elementAt(0));
12		vElementos_temp.addElement((Vector)vElementos.elementAt(1));
13			vTemp.addAll(fazer_instanciacao(vElementos_temp, nosFolhas, cont_retorno));
14		eliminar_vl(nosFolhas, vTemp, cont_retorno);
15		for(int i = 0; i< nosFolhas.size(); i++)
16			obj.inserir_nos((arvoreClass)nosFolhas.elementAt(i), (Vector)vTemp.elementAt(0),
17				(Vector)vTemp.elementAt(1), 1);
18		obj.construir_nosFolhas(nosFolhas);
19		eliminar_nosFolhas(nosFolhas);
20			obj.separar_nos(nosFolhas, tempEsq, tempDir, vTemp);
21			obj.analisar_nosFolhas(nosFolhas, tempFolhas, tempVetor);
22		for(int i = 0; i < tempVetor.size(); i++)
23		{
24			Vector tempF = new Vector(1,1);
25			Vector tempV = new Vector(1,1);
26			tempF.addElement(tempFolhas.elementAt(i));
27			tempV.addElement((Vector)tempVetor.elementAt(i));
28			verificar_elemento(tempF, tempV, cont_retorno);
29		}
30		obj.construir_nosFolhas(nosFolhas);
31			for (int i = 0; i < tempEsq.size(); i++)
32		obj.separar_nos((arvoreClass)tempEsq.elementAt(i), temp);
33		vTemp.clear();
34		vElementos_temp.clear();
35		vElementos_temp.addElement((Vector)vElementos.elementAt(2));
36		vTemp.addAll(fazer_instanciacao(vElementos_temp, temp, cont_retorno));
37		eliminar_vl(temp, vTemp, cont_retorno);
38		for(int i = 0; i< temp.size(); i++)
39			if (!obj.inserir_nos((arvoreClass)temp.elementAt(i), vTemp, 0))
40				temp.remove(i);
41		nosFolhas.addAll(temp);
42			temp.clear();
43		for (int i = 0; i < tempDir.size(); i++)
44			obj.separar_nos((arvoreClass)tempDir.elementAt(i), temp);
45		vTemp.clear();
46		vElementos_temp.clear();
47		vElementos_temp.addElement((Vector)vElementos.elementAt(3));
48		vTemp.addAll(fazer_instanciacao(vElementos_temp, temp, cont_retorno));
49		eliminar_vl(temp, vTemp, cont_retorno);
50		for(int i = 0; i< temp.size(); i++)
51		{
52			if (!obj.inserir_nos((arvoreClass)temp.elementAt(i), vTemp, 0))
53				temp.remove(i);
54		}
55		nosFolhas.addAll(temp);
56	}
```

Figura 44: Método caminho_sequencial_BIC

Em resumo, no tratamento do bicondicional deve ser criado um vetor com o primeiro e o segundo termos duplicados e apesar destes terem o mesmo identificador serão tratados como elementos distintos. Para a realização da instanciação dos elementos deve ser observado se o quantificador é externo ou interno ao bicondicional, pois isso causará modificações substanciais na instanciação das variáveis e na forma como será preenchido os vetores de instanciações; no exemplo em questão, com a negação do elemento (∀x-8), este passa a ser incluído no vetor do Existencial e sua variável será instanciada por um elemento que não ocorrer no(s) ramo(s) da inserção do elemento. Em contrapartida,

o elemento da fórmula que não foi negado, não tem na tabela de instanciação do universal outro termo semelhante (que apresenta o mesmo quantificador, a mesma variável e o mesmo identificador), poderia, assim, substituir a variável pela letra que fosse melhor adequada no(s) ramo(s) que o elemento estivesse sendo inserido (Figura 45).

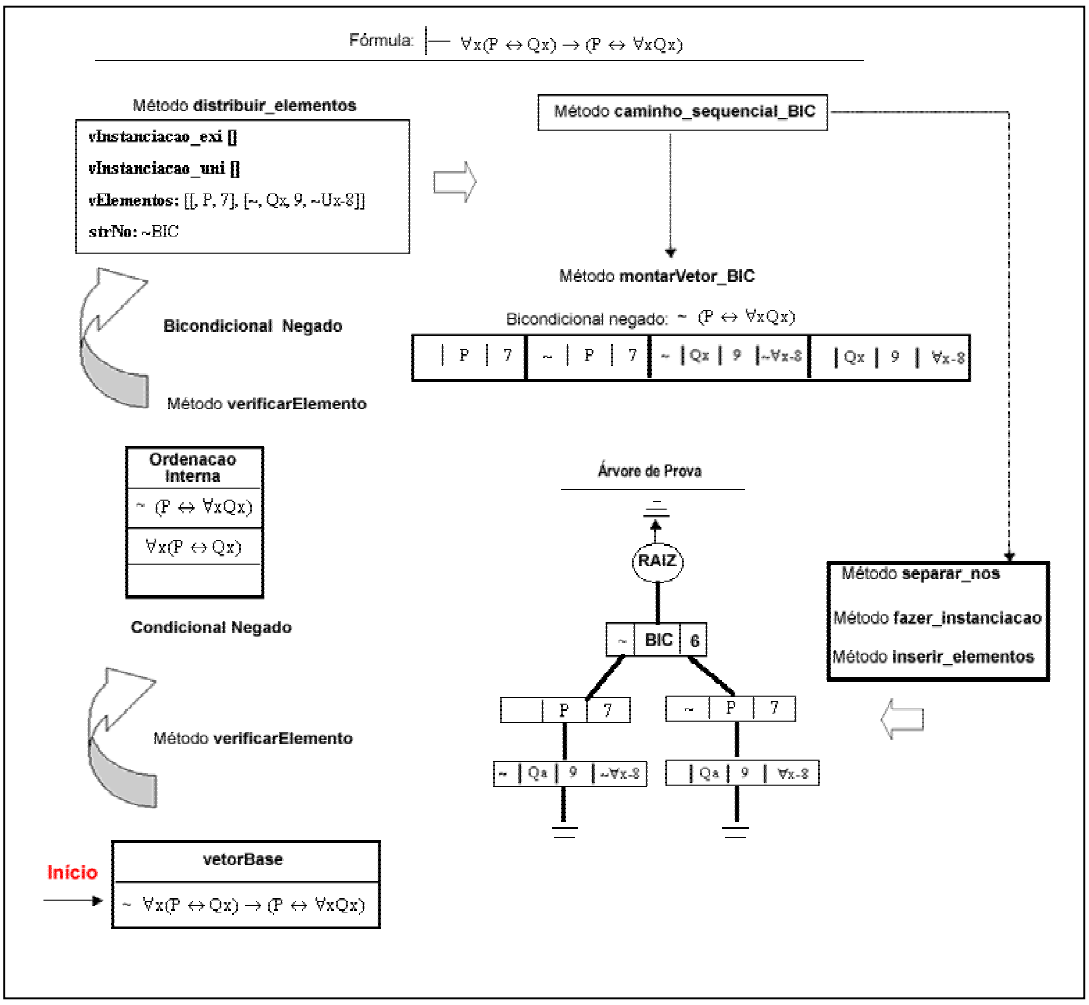

Figura 45: Esquema de tratamento dos elementos de um bicondicional

5.5 Considerações

Neste capítulo foram apresentadas explicações sobre a implementação do provador, desde a geração da árvore a partir do *parser* da API DOM até a inserção dos elementos instanciados na árvore de prova e a verificação da validade ou invalidade de um argumento. O código completo do provador está apresentado no capítulo 8 (anexos) desta monografia.

A implementação do primeiro refinamento apresentado nesse trabalho na seção 5.1 foi realizada a partir da possibilidade de trabalhar com todos os elementos que compõem a fórmula. Assim, utilizando a árvore gerada, os elementos internos de cada nó eram verificados e a ordenação realizada.

Para o segundo refinamento, foi necessário considerar a definição de métodos que pudessem trabalhar as especificidades das instanciações dos dois quantificadores. Desta forma, para cada instanciação, além dos vetores criados para guardar o histórico dos elementos, foi necessário ter acesso aos nós folhas que indicassem o caminho em que ocorreria a inclusão do elemento e a árvore completa gerada pela API DOM, para a verificação da existência de constantes na fórmula original (ou seja, a fórmula antes de sofrer qualquer instanciação).

Com a definição e implementação dos métodos e das estruturas a partir da lógica proposta nos refinamentos, foi possível gerar a árvore de prova e verificar a sua validade. Caso a fórmula não fosse válida, era realizado o retorno aos elementos que pudessem gerar uma contradição nos ramos abertos. Assim, foi encontrada, em uma quantidade de nós reduzida, a solução para vários problemas, inferindo a validade ou invalidade da fórmula.

Capítulo 6

Considerações Finais

Este trabalho apresenta a implementação de um provador de teoremas tendo como base o sistema de resolução de provas por refutação denominado método dos tableaux. A partir deste método foi possível desenvolver duas propostas de refinamentos para tornar o provador mais eficiente:

- a ordenação das fórmulas a partir da observância de uma economia na utilização de constantes e a priorização de instanciações de fórmulas que não bifurcassem o caminho na árvore de prova.
- a instanciação “inteligente” das variáveis quantificadas universalmente.

Para a realização das situações supracitadas, foi necessária a definição de estruturas que armazenassem informações sobre os elementos componentes de uma fórmula. Pois, para que as instanciações pudessem ser realizadas era necessário entender a origem de cada elemento, os quantificadores a que ele estava agregado, bem como suas variáveis e constantes. A definição destas estruturas (denominadas tabelas de instanciações dos quantificadores e histórico dos elementos) possibilitou a implementação dos métodos necessários para a validação dos refinamentos.

Em relação ao retorno às fórmulas dos ramos abertos, vale ressaltar que para o provador atingir uma maior eficiência, a ordenação deve tratar os elementos de forma diferenciada, ou seja, deve dar prioridades para a instanciação de elementos ligados apenas a quantificadores universais, pois há uma maior probabilidade desses elementos alcançarem uma adequação a outros elementos semelhantes e gerar a contradição. Então, nesse sentido, a ordenação de elementos ligados também a quantificadores existenciais só iria ser realizada, nos casos, em que a situação anterior não fosse suficiente.

Ao desenvolver os refinamentos para o método dos Tableaux, buscou-se trabalhar com todos os conectivos lógicos básicos, sem a necessidade do lançamento de equivalências de fórmulas para se adequar à estrutura. Assim, quando o bicondicional é tratado, este é incluído no documento XML como um operador que possui dois elementos. Apenas no tratamento desses elementos é que ocorre a duplicação dos seus termos e a adição do sinal da negação. Assim, teve-se a preocupação em buscar um tratamento individualizado para cada conectivo, considerando suas particularidades e especificidades.

Tabela 4: Comparação entre provadores

FÓRMULA	(FENDT, 2000)				**Provador Java/Xml**
	Sist. Tableaux Tradicional	Primeiro Refinamento	Segundo Refinamento	Terceiro Refinamento	Refinamento Ordenação/ Instanciação
PL1	20	20	20	20	13
PL2	12	12	12	12	5
PL3	6	6	6	6	6
PL4	19	19	19	19	13
PL5	16	15	15	15	13
PL6	3	3	3	3	3
PL7	4	4	4	4	3
PL8	6	6	6	6	6
PL9	24	18	18	18	24
PL10	32	28	28	28	30
PL11	9	9	9	9	5
PL12	174	144	144	144	109
PL13	39	39	39	35	39
PL14	54	48	48	48	35
PL15	18	16	16	16	13
PL16	6	6	6	6	6
PL17	60	60	60	60	83
PL18	17	17	17	14	5
PL19	139	139	139	121	28
PL20	-	-	-	91	17
PL21	60	48	42	49	23
PL22	51	48	43	43	14
PL23	31	31	30	31	13
PL24	284	173	111	145	41
PL25	66	66	52	59	30
PL26	-	-	-	6818	108
PL27	234	95	74	81	53
PL28	-	-	172	142	35
PL29	6985	4534	2660	110	74
PL30	72	72	57	57	16
PL31	37	37	26	30	11
PL32	597	596	426	583	25
PL33	879	226	182	184	53
PL34	-	-	2290	971	473
PL35	43	38	38	35	5
PL36	-	-	-	-	186
PL37	13	13	10	11	5

Para a verificação da eficiência do provador foram utilizados os resultados obtidos na verificação de fórmulas em (FENDT, 2000). No trabalho supracitado (Tabela 4) foram realizados três refinamentos no método dos tableaux. O primeiro refinamento tem como objetivo eliminar as repetições desnecessárias de nós na árvore de prova. O segundo refinamento preocupa-se com a repetição de fórmulas quantificadas universalmente, assim, considera-se o fato de que não existe a necessidade de que se repita uma fórmula universal, instanciada para todos os termos fechados (ou seja, cuja variável já foi substituída por uma constante) de um determinado ramo, se não houver a possibilidade de que novos termos surjam no ramo. E o terceiro refinamento tenta trazer o método um pouco mais próximo à intuição humana.

No presente trabalho, optou-se por fazer um único refinamento tendo como objetivo básico dar ao provador estratégias de prova que simulassem a intuição humana. Desta forma, o diferencial foi ter sido realizada a ordenação dos elementos que compõem a fórmula, pois entender qual o melhor próximo passo numa etapa de prova, é um fator primordial para dar ao provador um caráter mais intuitivo. Juntou-se, a esse fato, a instanciação a partir de um contexto bem definido das variáveis quantificadas universalmente, então foi possível instanciar variáveis a partir de uma verificação das necessidades do ramo para alcançar a contradição.

Pode ser verificado na tabela 4 que o provador desenvolvido neste trabalho teve nós superiores apenas em 10% das fórmulas verificadas e isso aconteceu em fórmulas proposicionais. Neste caso, na implementação do provador foi priorizada a adequação da estrutura às formulas quantificadas. No entanto, o problema identificado nestes itens é a questão da repetição das letras proposicionais que já ocorreram no ramo, pois nesse trabalho em alguns casos é aceita a duplicação dos elementos:

- Caso o elemento que esteja para ser inserido já exista na árvore, porém faça parte de uma fórmula que possui um conectivo lógico que causa bifurcação, então, este elemento não pode ser desconsiderado. Deve ser inserido na árvore para provocar a bifurcação no ramo.

- Um outro ponto observado é a forma como foi implementado o bicondicional. Seus elementos são ordenados e depois é realizada a instanciação e posterior inserção na árvore de prova. Como, a princípio, não há como identificar qual será o elemento que bifurcará o ramo, não há como evitar a inserção dos elementos, pois pode ocorrer que o elemento em questão seria aquele necessário para bifurcar o ramo. No entanto, em termos gerais, conforme pode ser observado em fórmulas quantificadas que tratam o bicondicional, os resultados em números de nós foram bastante inferiores ao outro provador analisado.

O que pode ser inferido a partir da análise da tabela supracitada é que a questão do entendimento da utilização, em primeiro lugar, dos elementos que não bifurcassem o caminho na árvore e que não provocasse a utilização de constante diferenciada gerou uma melhoria significativa nos resultados e até mesmo a solução de fórmulas que no provador anterior havia provocado repetição contínua.

Na figura 46 é apresentada uma fórmula extraída do artigo do (REEVES,1983). Nesse artigo, é observado que a prova desta fórmula não pode ser demonstrada numa quantidade de nós específica, pois causa uma repetição contínua no provador. No entanto, a partir das regras estabelecidas nesse trabalho, é possível verificar quais nós ficam abertos para posterior entendimento da validade ou invalidade da fórmula. Podemos verificar que os ramos abertos (que possuem ao final o símbolo da interrogação) têm grandes possibilidades de ficarem abertos sob quaisquer condições (tendo como base a fórmula), pois não há a priori a possibilidade de encontrar elementos contraditórios nos ramos. Assim, nesses casos, é estabelecido, que seja interrompido o processo de busca e mostrado a árvore de prova completa e seus ramos abertos. Para o estabelecimento da impossibilidade de encontrar uma prova da validade ou invalidade da fórmula foram utilizados, neste exemplo, 54 nós.

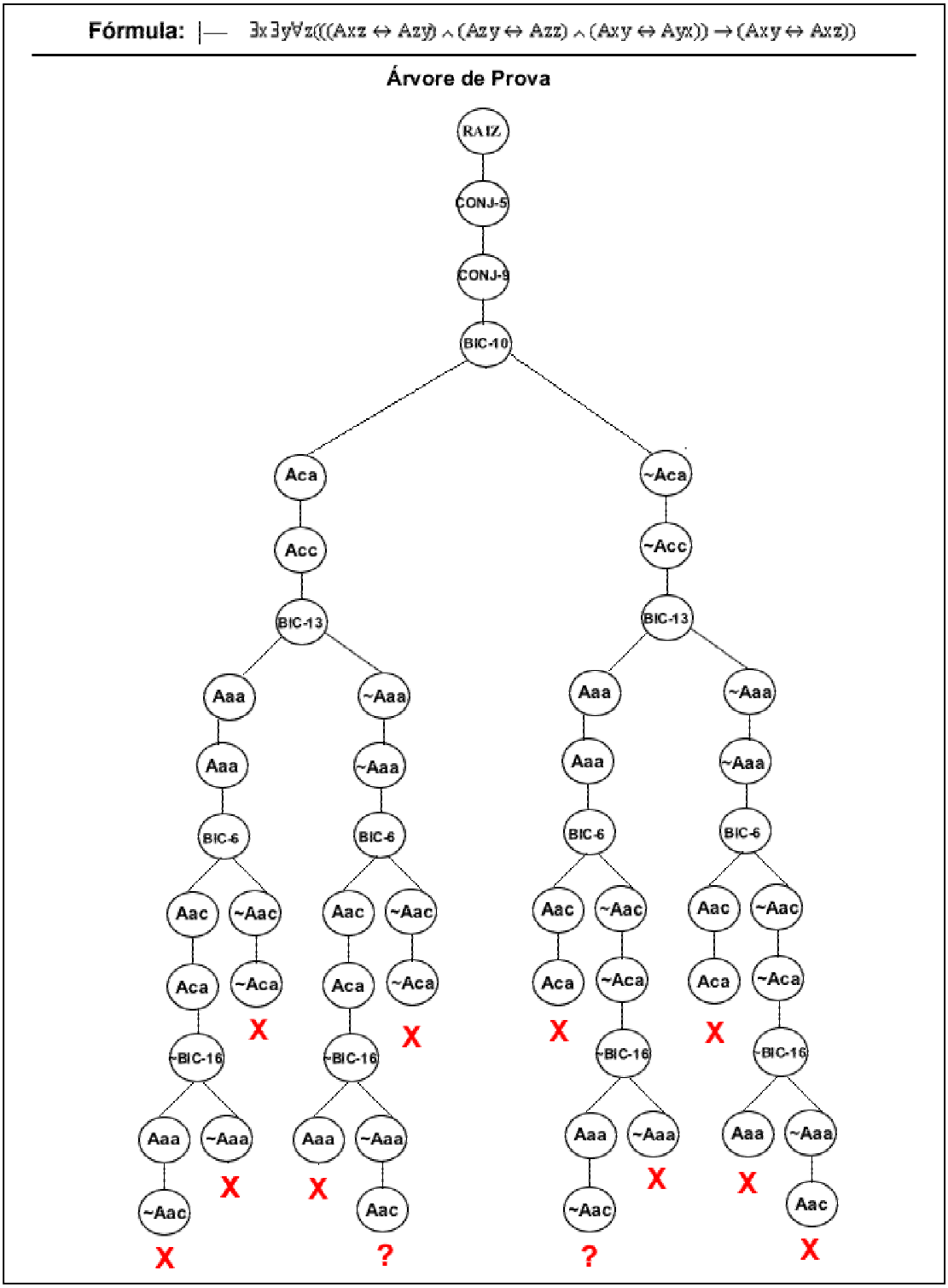

Figura 46: Árvore de prova gerada a partir do provador desenvolvido nesse trabalho

Para a fase de testes foram utilizadas 100 fórmulas. Alguns dos documentos XML destas fórmulas estão disponíveis no anexo (cap. 8) deste trabalho.

Com os resultados apresentados, pode-se verificar a possibilidade de se desenvolver alguns trabalhos futuros:

- Criação de uma base de teoremas a partir da utilização da estrutura das fórmulas definida em XML. Assim, o provador poderia gerar um conjunto de fórmulas válidas a partir de uma determinada estrutura. Ou seja, dada a estrutura de uma fórmula $\exists x \forall y P \rightarrow \forall y \exists x P$, é possível ao provador inferir qualquer P e ainda, assim, produzir uma prova válida.
- Realização de um editor para inclusão das fórmulas e apresentação dos resultados. Como cada nó da árvore de prova carrega consigo um histórico do elemento e a partir das tabelas de instanciação pode-se entender o processo de substituição das variáveis, é possível apresentar não somente a árvore, como também as estratégias que estão sendo utilizadas em cada etapa da prova.
- Expansão do provador para o tratamento da lógica equacional.
- Expansão do provador para o tratamento das lógicas não-clássicas.

Capítulo 7

Referências Bibliográficas

(ANDERSON, 2001) ANDERSON, R. et al. **Professional XML.** Rio de Janeiro: Editora Ciência Moderna LTDA., 2001

(BELL E MACHOVER, 1977) BELL, J.L.; MACHOVER, M.. **A Course in Mathematical Logic.** North-Holland Publishing Company. 1977.

(BESSIE e GLENNAN, 2000) BESSIE, Joseph; GLENNAN, Stuart. **Elements of Deductive Inference.: An Introduction To Symbolic Logic.** Wadsworth Publishing Company. 2000.

(BOYER at al, 1995) B. Boyer, M. Kaufmann and J.S. Moore. **The Boyer-Moore Theorem Provers and It's Interactive Enhancement.** Computers and Mathematics with Applications, Vol. 29, No. 2, pp. 27-62.

(BUCHSBAUM e PEQUENO, 1990) Buchsbaum, Arthur e 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 Fisolofia da Pontificia Universidade Católica do Rio de Janeiro, nº 3, setembro de 1990.

(COPI, 1978) Copi, Irving M. **Introdução à Lógica.** São Paulo: Ed. Mestre Jou, 1978.

(COSTA, 1980) COSTA, Newton C. A. **Ensaio sobre os fundamentos da lógica.** Hucitec: Ed. da Universidade de São Paulo, 1980.

(FENDT, 2000) Fendt, Leticia Carvalho Pivetta. **Refinamentos para o Método dos Tableaux.** Dissertação de Mestrado. Universidade Federal de Santa Catarina – UFSC. 2000.

(FITTING, 1990) FITTING, Melvin. **First-Order Logic and Automated Theorem Proving.** Springer Verlag, 1990.

(HEGENBERG, 1995) Hegenberg, Leônidas. **Dicionário de Lógica.** São Paulo: EPU, 1995.

(HERWIJNEN, 1994) van HERWIJNEN, E. **Practical SGML.** Kluwer Academic Publishers, 2. Ed., 1994

(JOHNSON, 1996) Johnson, Robert . **Parallel Analytic Tableaux Systems.** Tese de Doutorado. Department of Computer Science, Queen Mary and Westfield College, University of London. Londres, 1996.

(KIRK e PITTS-MOULTIS, 2000) KIRK, Cheryl, PITTS-MOULTIS, Natanya. **XML – Black Book.** São Paulo: Makron Books, 2000.a

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

(MACGRATH, 1999) MACGRATH, Sean. **XML – Aplicações práticas.** Rio de Janeiro: Campus, 1999.

(NOLT, 1988) Nolt, John, Rohatyn, Dennis. **Theory and Problems of Logic.** McGraw-Hill Book Co., Schaum's Outline Series, New York, 1988.

(REEVES, 1983) REEVES. S. V. **An Introduction to Semantic Tableaux.** Departament of Computer Science. CMS-55, University of Essex, 1983.

(ROBINSON, 1965) J. A. Robinson. **A machine-oriented logic based on the resolution principle.** *Journal of the ACM*, 1(12):23-41, 1965.

(SOUZA, 2002) João Nunes de. **Lógica para Ciência da Computação: fundamentos da linguagem, semântica e sistemas de dedução.** Editora Campus Ltda. 2002.

(SMULLYAN, 1995) Smullyan, R. M. **First-Order Logic.** Dover Publications, Inc. City University of New York and Indiana University. New York.

(THIRY, 1996) THIRY, Philippe. **Noções de Lógica.** De Boeck & Larcier S.A.. 1996.

(W3C, 1998) W3C – World Wide Web Consortium. **Document Object Model (DOM) Level 1 Specification.** Disponível em < http://www.w3.org/TR/REC-DOM-Level-1/>, 2002. Acesso em 20/11/2002.

(W3C, 1998a) **W3C – World Wide Web Consortium. eXtensible Markup Language (XML).** Disponível em <http://www.w3.org/XML/>, 1998. Acesso em 25/06/2002.

Capítulo 8

Anexos

8.1 Problemas utilizados para a comparação entre os provadores.

As fórmulas abaixo foram extraídas de (PELLETIER, 1986).

PL1	|-- (P → Q) ↔ (~Q → ~P)
PL2	|-- ~~P ↔ P
PL3	|-- ~(P → Q) → (Q → P)
PL4	|-- (~P → Q) ↔ (~Q → P)
PL5	|-- ((P ∨ Q) → (P ∨ R)) → (P ∨ (Q → R))
PL6	|-- P ∨ ~P
PL7	|-- P ∨ ~~~P
PL8	|-- ((P → Q) → P) → P
PL9	|-- ((P ∨ Q) ∧ ((~P ∨ Q) ∧ (P ∨ ~Q))) → ~ (~ P ∨ ~ Q)
PL10	P → R, R →(P ∧ Q), P → (Q ∨ R) |-- P ↔ Q
PL11	|-- P ↔ P
PL12	|-- ((P ↔ Q) ↔ R) ↔ (P ↔ (Q ↔ R))
PL13	|-- (P ∨ (Q ∧ R)) ↔ (P ∨ Q) ∧ (P ∨ R)
PL14	|-- (P ↔ Q) ↔ ((P ∨ ~Q) ∧ (~Q ∨ P))
PL15	|-- (P → R) ↔ (~Q ∨ Q)
PL16	|-- (P → Q) ∨ (Q → P)
PL17	|-- ((P ∧ (Q → R)) → S) ↔ ((~P ∨ (Q ∨ S)) ∧(~P ∨ (~R ∨ S)))
PL18	|-- ∃x∀y(Py → Px)
PL19	|-- ∃x∀y∀z((Py → Qz) → (Px → Qx))
PL20	|-- (∀x∀y∃z∀w((Px ∧ Py) → (Rz ∧ Sw))) → (∃x∃y(Px ∧ Qy) → ∃zRz)
PL21	∃x(P → Qx), ∃x(Qx → P) |-- ∃x(P ↔ Qx)
PL22	|-- ∀x(P ↔ Qx) → (P ↔ ∀xQx)
PL23	|-- ∀x(P ∨ Qx) ↔ (P ∨ ∀xQx)
PL24	~∃x(Sx ∧ Qx), ∀x(Px → (Qx ∨ Rx)), ~∃xPx → ∃xQx, ∀x((Qx ∨ Rx) → Sx) |-- ∃x(Px ∧ Rx)
PL25	∃xPx, ∀x(Sx → (~Tx ∧ Rx)), ∀x(Px → (Tx ∧ Sx)), ∀x(Px →Qx) ∨ ∃x(Px ∧ Rx) |-- ∃x(Qx ∧ Px)
PL26	∃xPx ↔ ∃xQx, ∀x∀y((Px ∧ Qy) → (Rx ↔ Sy)) |-- ∀x(Px → Rx) ↔ ∀x(Qx → Sx)
PL27	∃x(Px ∧ ~Qx), ∀x(Px → Rx), ∀x((Tx ∧ Sx) → Px), ∃x(Rx ∧ ~Qx) → ∀x(Tx → ~Rx) |-- ∀x (Tx → ~Sx)
PL28	∀xPx → ∀xQx, ∀x(Qx ∨ Rx) → ∃x(Qx ∧ Sx), ∃xSx → ∀x(Tx →Mx) |-- ∀x((Px ∧ Tx) → Mx)
PL29	∃xPx ∧ ∃xQx |-- (∀x(Px → Rx) ∧ ∀x(Qx → Sx)) ↔ ∀x∀y((Px ∧ Qy) → (Rx ∧ Sy))
PL30	∀x((Px ∨ Qx) → ~Rx), ∀x((Qx → ~Sx) → (Px ∧ Rx)) |-- ∀xSx
PL31	~∃x((Px ∧ (Qx ∨ Rx)), ∃x(Sx ∧ Px), ∀x(~Rx → Tx) |-- ∃x(Sx ∧ Tx)
PL32	∀x((Px ∧ (Qx ∨ Rx)) → Sx), ∀x((Sx ∧ Rx) → Tx), ∀x(Mx → Rx) |-- ∀x((Px ∧ Mx) → Tx)

PL33	|-- ∀x((Pa ∧ (Px → Pb)) → Pc) ↔ ∀x((~Pa ∨ (Px ∨ Pc)) ∧ (~Pa ∨ (Pb ∨ Pc)))
PL34	|-- (∃x∀y(Px ↔ Py) ↔ (∃xQx ↔ ∀yPy)) ↔ (∃x∀y(Qx ↔ Qy) ↔ (∃xPx ↔ ∀yPy))
PL35	|-- ∃x∃y(Pxy → ∀x∀yPxy)
PL36	∀x∃yPxy, ∀x∃yQxy, ∀x∀y((Pxy ∨ Qxy) → ∀z((Pyz ∨ Qyz) → Rxz) |-- ∀x∃yRxy
PL37	∀z∃w∀x∃y((Pxz → Pyw) ∧ (Pyz ∧ (Pyw → ∃uQuw))), ∀x∀z(~Pxz → ∃yQyz), ∃x∃yQxy → ∀xRxx |-- ∀x∃yRxy
PL39	|-- ∃x∀y(Pxy ↔ ~Pyy)

8.2 DTD para a validação das fórmulas proposicionais e quantificacionais

```
<?xml version="1.0"?>
<!DOCTYPE ARG[
<!ELEMENT ARG (PREM*, CONC)>
<!ELEMENT PREM (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT CONC (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT VAR (#PCDATA)>
<!ELEMENT LNOM (#PCDATA)>
<!ELEMENT PRED (#PCDATA)>
<!ELEMENT LPRED (PRED, (VAR*|LNOM*), (VAR*|LNOM*))>
<!ELEMENT COND (ANT,CONS)>
<!ELEMENT BIC (PRIM,SEG)>
<!ELEMENT DISJ (PRIM,SEG)>
<!ELEMENT CONJ (PRIM,SEG)>
<!ELEMENT ANT (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT CONS (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT PRIM (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT SEG (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI)>
<!ELEMENT UNI (VAR, (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI))>
<!ELEMENT EXI (VAR, (LPRED|COND|DISJ|CONJ|BIC|UNI|EXI))>
<!ATTLIST LPRED ID CDATA #IMPLIED>
<!ATTLIST LPRED NEG CDATA #IMPLIED>
<!ATTLIST COND ID CDATA #IMPLIED>
<!ATTLIST COND NEG CDATA #IMPLIED>
<!ATTLIST BIC ID CDATA #IMPLIED>
<!ATTLIST BIC NEG CDATA #IMPLIED>
<!ATTLIST CONJ ID CDATA #IMPLIED>
<!ATTLIST CONJ NEG CDATA #IMPLIED>
<!ATTLIST DISJ ID CDATA #IMPLIED>
<!ATTLIST DISJ NEG CDATA #IMPLIED>
<!ATTLIST UNI ID CDATA #IMPLIED>
<!ATTLIST UNI NEG CDATA #IMPLIED>
<!ATTLIST EXI ID CDATA #IMPLIED>
<!ATTLIST EXI NEG CDATA #IMPLIED>
]>
```

8.2 Documentos XML das fórmulas extraídas de (PELLETIER, 1986)

```
<!-- |- (P -> Q) <-> (~Q -> ~P) -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <COND ID="2" NEG="">
                                        <ANT>
                                                <LPRED ID="3" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </PRIM>
                        <SEG>
                                <COND ID="5" NEG="">
                                        <ANT>
                                                <LPRED ID="6" NEG="~">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="7" NEG="~">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- ~~P <-> P -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <LPRED ID="2" NEG="~~">
                                        <PRED>P</PRED>
                                </LPRED>
                        </PRIM>
                        <SEG>
                                <LPRED ID="3" NEG="">
                                        <PRED>P</PRED>
                                </LPRED>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- ~(P -> Q) -> (Q -> P) -->

<ARG>
        <CONC>
                <COND ID="1" NEG="">
                        <ANT>
                                <COND ID="2" NEG="~">
                                        <ANT>
                                                <LPRED ID="3" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </ANT>
                        <CONS>
                                <COND ID="5" NEG="">
                                        <ANT>
                                                <LPRED ID="6" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="7" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </CONS>
                </COND>
        </CONC>
</ARG>
```

```
<!-- |- (~P -> Q) <-> (~Q -> P) -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <COND ID="2" NEG="">
                                        <ANT>
                                                <LPRED ID="3" NEG="~">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </PRIM>
                        <SEG>
                                <COND ID="5" NEG="">
                                        <ANT>
                                                <LPRED ID="6" NEG="~">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="7" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- ((P ou Q) -> (P ou R)) -> (P ou (Q -> R))-->

<ARG>
        <CONC>
                <COND ID="1" NEG="">
                        <ANT>
                                <COND ID="2" NEG="">
                                        <ANT>
                                                <DISJ ID="3" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="4" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="5" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </ANT>
                                        <CONS>
                                                <DISJ ID="6" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="7" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="8" NEG="">
                                                                        <PRED>R</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </CONS>
                                </COND>
                        </ANT>
                        <CONS>
                                <DISJ ID="9" NEG="">
                                        <PRIM>
                                                <LPRED ID="10" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <COND ID="11" NEG="">
                                                        <ANT>
                                                                <LPRED ID="12" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </ANT>
                                                        <CONS>
                                                                <LPRED ID="13" NEG="">
                                                                        <PRED>R</PRED>
                                                                </LPRED>
                                                        </CONS>
                                                </COND>
                                        </SEG>
                                </DISJ>
                        </CONS>
                </COND>
        </CONC>
</ARG>
```

```
<!-- |- (P ou ~P)-->

<ARG>
        <CONC>
                <DISJ ID="1" NEG="">
                        <PRIM>
                                <LPRED ID="2" NEG="">
                                        <PRED>P</PRED>
                                </LPRED>
                        </PRIM>
                        <SEG>
                                <LPRED ID="3" NEG="~">
                                        <PRED>P</PRED>
                                </LPRED>
                        </SEG>
                </DISJ>
        </CONC>
</ARG>
```

```
<!-- |- (P ou ~~~P)-->

<ARG>
        <CONC>
                <DISJ ID="1" NEG="">
                        <PRIM>
                                <LPRED ID="2" NEG="">
                                        <PRED>P</PRED>
                                </LPRED>
                        </PRIM>
                        <SEG>
                                <LPRED ID="3" NEG="~~~">
                                        <PRED>P</PRED>
                                </LPRED>
                        </SEG>
                </DISJ>
        </CONC>
</ARG>
```

```
<!-- |- ((P -> Q) -> P) -> P-->

<ARG>
        <CONC>
                <COND ID="1" NEG="">
                        <ANT>
                                <COND ID="2" NEG="">
                                        <ANT>
                                                <COND ID="3" NEG="">
                                                        <ANT>
                                                                <LPRED ID="4" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </ANT>
                                                        <CONS>
                                                                <LPRED ID="5" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </CONS>
                                                </COND>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="6" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </ANT>
                        <CONS>
                                <LPRED ID="7" NEG="">
                                        <PRED>P</PRED>
                                </LPRED>
                        </CONS>
                </COND>
        </CONC>
</ARG>
```

```
<!-- |- ((P ou Q) e ((~P ou Q) e (P ou ~Q))) -> ~(~P ou ~Q) -->

<ARG>
    <CONC>
        <COND ID="1" NEG="">
            <ANT>
                <CONJ ID="2" NEG="">
                    <PRIM>
                        <DISJ ID="3" NEG="">
                            <PRIM>
                                <LPRED ID="4" NEG="">
                                    <PRED>P</PRED>
                                </LPRED>
                            </PRIM>
                            <SEG>
                                <LPRED ID="5" NEG="">
                                    <PRED>Q</PRED>
                                </LPRED>
                            </SEG>
                        </DISJ>
                    </PRIM>
                    <SEG>
                        <CONJ ID="6" NEG="">
                            <PRIM>
                                <DISJ ID="7" NEG="">
                                    <PRIM>
                                        <LPRED ID="8" NEG="~">
                                            <PRED>P</PRED>
                                        </LPRED>
                                    </PRIM>
                                    <SEG>
                                        <LPRED ID="9" NEG="">
                                            <PRED>Q</PRED>
                                        </LPRED>
                                    </SEG>
                                </DISJ>
                            </PRIM>
                            <SEG>
                                <DISJ ID="10" NEG="">
                                    <PRIM>
                                        <LPRED ID="11" NEG="">
                                            <PRED>P</PRED>
                                        </LPRED>
                                    </PRIM>
                                    <SEG>
                                        <LPRED ID="12" NEG="~">
                                            <PRED>Q</PRED>
                                        </LPRED>
                                    </SEG>
                                </DISJ>
                            </SEG>
                        </CONJ>
                    </SEG>
                </CONJ>
            </ANT>
            <CONS>
                <DISJ ID="13" NEG="~">
                    <PRIM>
                        <LPRED ID="14" NEG="~">
                            <PRED>P</PRED>
                        </LPRED>
                    </PRIM>
                    <SEG>
                        <LPRED ID="15" NEG="~">
                            <PRED>Q</PRED>
                        </LPRED>
                    </SEG>
                </DISJ>
            </CONS>
        </COND>
    </CONC>
</ARG>
```

```
<!-- P -> R, R -> (P e Q), P -> (Q ou R) |- P <-> Q -->

<ARG>
	<PREM>
		<COND ID="1" NEG="">
			<ANT>
				<LPRED ID="2" NEG="">
					<PRED>P</PRED>
				</LPRED>
			</ANT>
			<CONS>
				<LPRED ID="3" NEG="">
					<PRED>R</PRED>
				</LPRED>
			</CONS>
		</COND>
	</PREM>
	<PREM>
		<COND ID="4" NEG="">
			<ANT>
				<LPRED ID="5" NEG="">
					<PRED>R</PRED>
				</LPRED>
			</ANT>
			<CONS>
				<CONJ ID="6" NEG="">
					<PRIM>
						<LPRED ID="7" NEG="">
							<PRED>P</PRED>
						</LPRED>
					</PRIM>
					<SEG>
						<LPRED ID="8" NEG="">
							<PRED>Q</PRED>
						</LPRED>
					</SEG>
				</CONJ>
			</CONS>
		</COND>
	</PREM>
	<PREM>
		<COND ID="9" NEG="">
			<ANT>
				<LPRED ID="10" NEG="">
					<PRED>P</PRED>
				</LPRED>
			</ANT>
			<CONS>
				<DISJ ID="11" NEG="">
					<PRIM>
						<LPRED ID="12" NEG="">
							<PRED>Q</PRED>
						</LPRED>
					</PRIM>
					<SEG>
						<LPRED ID="13" NEG="">
							<PRED>R</PRED>
						</LPRED>
					</SEG>
				</DISJ>
			</CONS>
		</COND>
	</PREM>
	<CONC>
		<BIC ID="14" NEG="">
			<PRIM>
				<LPRED ID="15" NEG="">
					<PRED>P</PRED>
				</LPRED>
			</PRIM>
			<SEG>
				<LPRED ID="16" NEG="">
					<PRED>Q</PRED>
				</LPRED>
			</SEG>
		</BIC>
	</CONC>
</ARG>
```

```
<!-- |- P <-> P -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <LPRED ID="2" NEG="">
                                        <PRED>P</PRED>
                                </LPRED>
                        </PRIM>
                        <SEG>
                                <LPRED ID="3" NEG="">
                                        <PRED>P</PRED>
                                </LPRED>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- ((P <-> Q) <-> R) <-> (P <-> (Q <-> R)) -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <BIC ID="2" NEG="">
                                        <PRIM>
                                                <BIC ID="3" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="4" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="5" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </BIC>
                                        </PRIM>
                                        <SEG>
                                                <LPRED ID="6" NEG="">
                                                        <PRED>R</PRED>
                                                </LPRED>
                                        </SEG>
                                </BIC>
                        </PRIM>
                        <SEG>
                                <BIC ID="7" NEG="">
                                        <PRIM>
                                                <LPRED ID="8" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <BIC ID="9" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="10" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="11" NEG="">
                                                                        <PRED>R</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </BIC>
                                        </SEG>
                                </BIC>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- (P ou (Q e R)) <-> ((P ou Q) e (P ou R)) -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <DISJ ID="2" NEG="">
                                        <PRIM>
                                                <LPRED ID="3" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <CONJ ID="4" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="5" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="6" NEG="">
                                                                        <PRED>R</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </CONJ>
                                        </SEG>
                                </DISJ>
                        </PRIM>
                        <SEG>
                                <CONJ ID="7" NEG="">
                                        <PRIM>
                                                <DISJ ID="8" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="9" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="10" NEG="">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </PRIM>
                                        <SEG>
                                                <DISJ ID="11" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="12" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="13" NEG="">
                                                                        <PRED>R</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </SEG>
                                </CONJ>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- (P <-> Q) <-> ((P ou ~Q) e (~Q ou P)) -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <BIC ID="2" NEG="">
                                        <PRIM>
                                                <LPRED ID="3" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </SEG>
                                </BIC>
                        </PRIM>
                        <SEG>
                                <CONJ ID="5" NEG="">
                                        <PRIM>
                                                <DISJ ID="6" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="7" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="8" NEG="~">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </PRIM>
                                        <SEG>
                                                <DISJ ID="9" NEG="">
                                                        <PRIM>
                                                                <LPRED ID="10" NEG="~">
                                                                        <PRED>Q</PRED>
                                                                </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                                <LPRED ID="11" NEG="">
                                                                        <PRED>P</PRED>
                                                                </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </SEG>
                                </CONJ>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- (P->R) <-> (~Q ou Q) -->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <COND ID="2" NEG="">
                                        <ANT>
                                                <LPRED ID="3" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>R</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </PRIM>
                        <SEG>
                                <DISJ ID="5" NEG="">
                                        <PRIM>
                                                <LPRED ID="6" NEG="~">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <LPRED ID="7" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </SEG>
                                </DISJ>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

```
<!-- |- (P -> Q) ou (Q -> P)  -->

<ARG>
        <CONC>
                <DISJ ID="1" NEG="">
                        <PRIM>
                                <COND ID="2" NEG="">
                                        <ANT>
                                                <LPRED ID="3" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </PRIM>
                        <SEG>
                                <COND ID="5" NEG="">
                                        <ANT>
                                                <LPRED ID="6" NEG="">
                                                        <PRED>Q</PRED>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="7" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </SEG>
                </DISJ>
        </CONC>
</ARG>
```

```
<!-- |- ((P e (Q -> R)) -> S) <-> ((~P ou (Q ou S)) e (~P ou (~R ou S)))  -->

<ARG>
    <CONC>
        <BIC ID="1" NEG="">
            <PRIM>
                <COND ID="2" NEG="">
                    <ANT>
                        <CONJ ID="3" NEG="">
                            <PRIM>
                                <LPRED ID="4" NEG="">
                                    <PRED>P</PRED>
                                </LPRED>
                            </PRIM>
                            <SEG>
                                <COND ID="5" NEG="">
                                    <ANT>
                                        <LPRED ID="6" NEG="">
                                            <PRED>Q</PRED>
                                        </LPRED>
                                    </ANT>
                                    <CONS>
                                        <LPRED ID="7" NEG="">
                                            <PRED>R</PRED>
                                        </LPRED>
                                    </CONS>
                                </COND>
                            </SEG>
                        </CONJ>
                    </ANT>
                    <CONS>
                        <LPRED ID="8" NEG="">
                            <PRED>S</PRED>
                        </LPRED>
                    </CONS>
                </COND>
            </PRIM>
            <SEG>
                <CONJ ID="9" NEG="">
                    <PRIM>
                        <DISJ ID="10" NEG="">
                            <PRIM>
                                <LPRED ID="11" NEG="~">
                                    <PRED>P</PRED>
                                </LPRED>
                            </PRIM>
                            <SEG>
                                <DISJ ID="12" NEG="">
                                    <PRIM>
                                        <LPRED ID="13" NEG="">
                                            <PRED>Q</PRED>
                                        </LPRED>
                                    </PRIM>
                                    <SEG>
                                        <LPRED ID="14" NEG="">
                                            <PRED>S</PRED>
                                        </LPRED>
                                    </SEG>
                                </DISJ>
                            </SEG>
                        </DISJ>
                    </PRIM>
                    <SEG>
                        <DISJ ID="15" NEG="">
                            <PRIM>
                                <LPRED ID="16" NEG="~">
                                    <PRED>P</PRED>
                                </LPRED>
                            </PRIM>
                            <SEG>
                                <DISJ ID="17" NEG="">
                                    <PRIM>
                                        <LPRED ID="18" NEG="~">
                                            <PRED>R</PRED>
                                        </LPRED>
                                    </PRIM>
                                    <SEG>
                                        <LPRED ID="19" NEG="">
                                            <PRED>S</PRED>
                                        </LPRED>
                                    </SEG>
                                </DISJ>
                            </SEG>
                        </DISJ>
                    </SEG>
                </CONJ>
            </SEG>
        </BIC>
    </CONC>
</ARG>
```

```
<!-- |- ExUy(Py->Px)-->

<ARG>
        <CONC>
                <EXI ID="1" NEG="">
                        <VAR>x</VAR>
                        <UNI ID="2" NEG="">
                                <VAR>y</VAR>
                                <COND ID="3" NEG="">
                                        <ANT>
                                                <LPRED ID="4" NEG="">
                                                        <PRED>P</PRED>
                                                        <VAR>y</VAR>
                                                </LPRED>
                                        </ANT>
                                        <CONS>
                                                <LPRED ID="5" NEG="">
                                                        <PRED>P</PRED>
                                                        <VAR>x</VAR>
                                                </LPRED>
                                        </CONS>
                                </COND>
                        </UNI>
                </EXI>
        </CONC>
</ARG>
```

```
<!-- |- ExUyUz((Py->Qz)->(Px->Qx))  -->

<ARG>
    <CONC>
        <EXI ID="1" NEG="">
                <VAR>x</VAR>
                <UNI ID="2" NEG="">
                        <VAR>y</VAR>
                        <UNI ID="3" NEG="">
                                <VAR>z</VAR>
                                <COND ID="4" NEG="">
                                        <ANT>
                                                <COND ID="5" NEG="">
                                                        <ANT>
                                                                <LPRED ID="6" NEG="">
                                                                        <PRED>P</PRED>
                                                                        <VAR>y</VAR>
                                                                </LPRED>
                                                        </ANT>
                                                        <CONS>
                                                                <LPRED ID="7" NEG="">
                                                                        <PRED>Q</PRED>
                                                                        <VAR>z</VAR>
                                                                </LPRED>
                                                        </CONS>
                                                </COND>
                                        </ANT>
                                        <CONS>
                                                <COND ID="8" NEG="">
                                                        <ANT>
                                                                <LPRED ID="9" NEG="">
                                                                        <PRED>P</PRED>
                                                                        <VAR>x</VAR>
                                                                </LPRED>
                                                        </ANT>
                                                        <CONS>
                                                                <LPRED ID="10" NEG="">
                                                                        <PRED>Q</PRED>
                                                                        <VAR>x</VAR>
                                                                </LPRED>
                                                        </CONS>
                                                </COND>
                                        </CONS>
                                </COND>
                        </UNI>
                </UNI>
        </EXI>
    </CONC>
</ARG>
```

```
<!-- |-(UxUyEzUw((Px e Qy)->(Rz e Sw)))->(ExEy(Px e Qy)->EzRz)-->

<ARG>
        <CONC>
                <COND ID="1" NEG="">
                        <ANT>
                                <UNI ID="3" NEG="">
                                        <VAR>x</VAR>
                                        <UNI ID="4" NEG="">
                                                <VAR>y</VAR>
                                                <EXI ID="5" NEG="">
                                                        <VAR>z</VAR>
                                                        <UNI ID="6" NEG="">
                                                                <VAR>w</VAR>
                                                                <COND ID="7" NEG="">
                                                                        <ANT>
                                                                                <CONJ ID="8" NEG="">
                                                                                        <PRIM>
                                                                                                <LPRED ID="9" NEG="">
                                                                                                        <PRED>P</PRED>
                                                                                                        <VAR>x</VAR>
                                                                                                </LPRED>
                                                                                        </PRIM>
                                                                                        <SEG>
                                                                                                <LPRED ID="10" NEG="">
                                                                                                        <PRED>Q</PRED>
                                                                                                        <VAR>y</VAR>
                                                                                                </LPRED>
                                                                                        </SEG>
                                                                                </CONJ>
                                                                        </ANT>
                                                                        <CONS>
                                                                                <CONJ ID="11" NEG="">
                                                                                        <PRIM>
                                                                                                <LPRED ID="12" NEG="">
                                                                                                        <PRED>R</PRED>
                                                                                                        <VAR>z</VAR>
                                                                                                </LPRED>
                                                                                        </PRIM>
                                                                                        <SEG>
                                                                                                <LPRED ID="13" NEG="">
                                                                                                        <PRED>S</PRED>
                                                                                                        <VAR>w</VAR>
                                                                                                </LPRED>
                                                                                        </SEG>
                                                                                </CONJ>
                                                                        </CONS>
                                                                </COND>
                                                        </UNI>
                                                </EXI>
                                        </UNI>
                                </UNI>
                        </ANT>
                        <CONS>
                                <COND ID="14" NEG="">
                                        <ANT>
                                                <EXI ID="15" NEG="">
                                                        <VAR>x</VAR>
                                                        <EXI ID="16" NEG="">
                                                                <VAR>y</VAR>
                                                                <CONJ ID="17" NEG="">
                                                                        <PRIM>
                                                                                <LPRED ID="18" NEG="">
                                                                                        <PRED>P</PRED>
                                                                                        <VAR>x</VAR>
                                                                                </LPRED>
                                                                        </PRIM>
                                                                        <SEG>
                                                                                <LPRED ID="19" NEG="">
                                                                                        <PRED>Q</PRED>
                                                                                        <VAR>y</VAR>
                                                                                </LPRED>
                                                                        </SEG>
                                                                </CONJ>
                                                        </EXI>
                                                </EXI>
                                        </ANT>
                                        <CONS>
                                                <EXI ID="20" NEG="">
                                                        <VAR>z</VAR>
                                                        <LPRED ID="21" NEG="">
                                                                <PRED>R</PRED>
                                                                <VAR>z</VAR>
                                                        </LPRED>
                                                </EXI>
                                        </CONS>
                                </COND>
                        </CONS>
                </COND>
        </CONC>
</ARG>
```

```
<!-- Ex(P->Qx), Ex(Qx->P) |- Ex(P<->Qx)-->

<ARG>
        <PREM>
                <EXI ID="1" NEG="">
                        <VAR>x</VAR>
                        <COND ID="2" NEG="">
                                <ANT>
                                        <LPRED ID="3" NEG="">
                                                <PRED>P</PRED>
                                        </LPRED>
                                </ANT>
                                <CONS>
                                        <LPRED ID="4" NEG="">
                                                <PRED>Q</PRED>
                                                <VAR>x</VAR>
                                        </LPRED>
                                </CONS>
                        </COND>
                </EXI>
        </PREM>
        <PREM>
                <EXI ID="5" NEG="">
                        <VAR>x</VAR>
                        <COND ID="6" NEG="">
                                <ANT>
                                        <LPRED ID="7" NEG="">
                                                <PRED>Q</PRED>
                                                <VAR>x</VAR>
                                        </LPRED>
                                </ANT>
                                <CONS>
                                        <LPRED ID="8" NEG="">
                                                <PRED>P</PRED>
                                        </LPRED>
                                </CONS>
                        </COND>
                </EXI>
        </PREM>
        <CONC>
                <EXI ID="9" NEG="">
                        <VAR>x</VAR>
                        <BIC ID="10" NEG="">
                                <PRIM>
                                        <LPRED ID="11" NEG="">
                                                <PRED>P</PRED>
                                        </LPRED>
                                </PRIM>
                                <SEG>
                                        <LPRED ID="12" NEG="">
                                                <PRED>Q</PRED>
                                                <VAR>x</VAR>
                                        </LPRED>
                                </SEG>
                        </BIC>
                </EXI>
        </CONC>
</ARG>
```

```
<!-- |-Ux(P<->Qx)->(P<->UxQx) -->

<ARG>
        <CONC>
                <COND ID="1" NEG="">
                        <ANT>
                                <UNI ID="2" NEG="">
                                        <VAR>x</VAR>
                                        <BIC ID="3" NEG="">
                                                <PRIM>
                                                        <LPRED ID="4" NEG="">
                                                                <PRED>P</PRED>
                                                        </LPRED>
                                                </PRIM>
                                                <SEG>
                                                        <LPRED ID="5" NEG="">
                                                                <PRED>Q</PRED>
                                                                <VAR>x</VAR>
                                                        </LPRED>
                                                </SEG>
                                        </BIC>
                                </UNI>
                        </ANT>
                        <CONS>
                                <BIC ID="6" NEG="">
                                        <PRIM>
                                                <LPRED ID="7" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <UNI ID="8" NEG="">
                                                        <VAR>x</VAR>
                                                        <LPRED ID="9" NEG="">
                                                                <PRED>Q</PRED>
                                                                <VAR>x</VAR>
                                                        </LPRED>
                                                </UNI>
                                        </SEG>
                                </BIC>
                        </CONS>
                </COND>
        </CONC>
</ARG>
```

```
<!-- |-Ux(P ou Qx) <-> (P ou UxQx)-->

<ARG>
        <CONC>
                <BIC ID="1" NEG="">
                        <PRIM>
                                <UNI ID="2" NEG="">
                                        <VAR>x</VAR>
                                        <DISJ ID="3" NEG="">
                                                <PRIM>
                                                        <LPRED ID="4" NEG="">
                                                                <PRED>P</PRED>
                                                        </LPRED>
                                                </PRIM>
                                                <SEG>
                                                        <LPRED ID="5" NEG="">
                                                                <PRED>Q</PRED>
                                                                <VAR>x</VAR>
                                                        </LPRED>
                                                </SEG>
                                        </DISJ>
                                </UNI>
                        </PRIM>
                        <SEG>
                                <DISJ ID="6" NEG="">
                                        <PRIM>
                                                <LPRED ID="7" NEG="">
                                                        <PRED>P</PRED>
                                                </LPRED>
                                        </PRIM>
                                        <SEG>
                                                <UNI ID="8" NEG="">
                                                        <VAR>x</VAR>
                                                        <LPRED ID="9" NEG="">
                                                                <PRED>Q</PRED>
                                                                <VAR>x</VAR>
                                                        </LPRED>
                                                </UNI>
                                        </SEG>
                                </DISJ>
                        </SEG>
                </BIC>
        </CONC>
</ARG>
```

8.3 Documentos XML das fórmulas extraídas de (REEVES, 1983)

```
<!-- |-(((UxEy(Axy ou Ayx)) e (UxUy(Axy -> Ayy))) -> (Ez(Azz))-->

<ARG>
    <CONC>
        <COND ID="1" NEG="">
          <ANT>
                <CONJ ID="2" NEG="">
                        <PRIM>
                                <UNI ID="3" NEG="">
                                        <VAR>x</VAR>
                                        <EXI ID="4" NEG="">
                                                <VAR>y</VAR>
                                                <DISJ ID="5" NEG="">
                                                        <PRIM>
                                                            <LPRED ID="6" NEG="">
                                                                  <PRED>A</PRED>
                                                                  <VAR>xy</VAR>
                                                            </LPRED>
                                                        </PRIM>
                                                        <SEG>
                                                            <LPRED ID="7" NEG="">
                                                                  <PRED>A</PRED>
                                                                  <VAR>yx</VAR>
                                                            </LPRED>
                                                        </SEG>
                                                </DISJ>
                                        </EXI>
                                </UNI>
                        </PRIM>
                        <SEG>
                                <UNI ID="8" NEG="">
                                        <VAR>x</VAR>
                                        <UNI ID="9" NEG="">
                                                <VAR>y</VAR>
                                                <COND ID="10" NEG="">
                                                        <ANT>
                                                            <LPRED ID="11" NEG="">
                                                                  <PRED>A</PRED>
                                                                  <VAR>xy</VAR>
                                                            </LPRED>
                                                        </ANT>
                                                        <CONS>
                                                            <LPRED ID="12" NEG="">
                                                                  <PRED>A</PRED>
                                                                  <VAR>yy</VAR>
                                                            </LPRED>
                                                        </CONS>
                                                </COND>
                                        </UNI>
                                </UNI>
                        </SEG>
                </CONJ>
          </ANT>
          <CONS>
                <EXI ID="13" NEG="">
                        <VAR>z</VAR>
                        <LPRED ID="14" NEG="">
                                <PRED>A</PRED>
                                <VAR>zz</VAR>
                        </LPRED>
                </EXI>
          </CONS>
        </COND>
    </CONC>
</ARG>
```

```
<!-- |- ExEyUz(((Axz<->Azy) e (Azy<->Azz) e (Axy<->Ayx))-> (Axy<->Axz)) -->

<ARG>
   <CONC>
      <EXI ID="1" NEG="">
          <VAR>x</VAR>
          <EXI ID="2" NEG="">
                   <VAR>y</VAR>
                   <UNI ID="3" NEG="">
                            <VAR>z</VAR>
                            <COND ID="4" NEG="">
                                     <ANT>
                                              <CONJ ID="5" NEG="">
                                                       <PRIM>
                                                           <BIC ID="6" NEG="">
                                                               <PRIM>
                                                                   <LPRED ID="7" NEG="">
                                                                       <PRED>A</PRED>
                                                                       <VAR>xz</VAR>
                                                                   </LPRED>
                                                               </PRIM>
                                                               <SEG>
                                                                   <LPRED ID="8" NEG="">
                                                                       <PRED>A</PRED>
                                                                       <VAR>zy</VAR>
                                                                   </LPRED>
                                                               </SEG>
                                                           </BIC>
                                                       </PRIM>
                                                       <SEG>
                                                           <CONJ ID="9" NEG="">
                                                               <PRIM>
                                                                   <BIC ID="10" NEG="">
                                                                       <PRIM>
                                                                           <LPRED ID="11" NEG="">
                                                                               <PRED>A</PRED>
                                                                               <VAR>zy</VAR>
                                                                           </LPRED>
                                                                       </PRIM>
                                                                       <SEG>
                                                                           <LPRED ID="12" NEG="">
                                                                               <PRED>A</PRED>
                                                                               <VAR>zz</VAR>
                                                                           </LPRED>
                                                                       </SEG>
                                                                   </BIC>
                                                               </PRIM>
                                                               <SEG>
                                                                   <BIC ID="13" NEG="">
                                                                       <PRIM>
                                                                           <LPRED ID="14" NEG="">
                                                                               <PRED>A</PRED>
                                                                               <VAR>xy</VAR>
                                                                           </LPRED>
                                                                       </PRIM>
                                                                       <SEG>
                                                                           <LPRED ID="15" NEG="">
                                                                               <PRED>A</PRED>
                                                                               <VAR>yx</VAR>
                                                                           </LPRED>
                                                                       </SEG>
                                                                   </BIC>
                                                               </SEG>
                                                           </CONJ>
                                                       </SEG>
                                              </CONJ>
                                     </ANT>
                                     <CONS>
                                              <BIC ID="16" NEG="">
                                                       <PRIM>
                                                               <LPRED ID="17" NEG="">
                                                                       <PRED>A</PRED>
                                                                       <VAR>xy</VAR>
                                                               </LPRED>
                                                       </PRIM>
                                                       <SEG>
                                                               <LPRED ID="18" NEG="">
                                                                       <PRED>A</PRED>
                                                                       <VAR>xz</VAR>
                                                               </LPRED>
                                                       </SEG>
                                              </BIC>
                                     </CONS>
                            </COND>
                   </UNI>
          </EXI>
      </EXI>
   </CONC>
</ARG>
```