L-EditXML: Ambiente para Manipulação e Apresentação das Etapas de Prova de Fórmulas da Lógica Quantificacional

Bruno Vilar[1], Parcilene Fernandes de Brito[1]

[1]Curso de Sistemas de Informação – Centro Universitário Luterano de Palmas (CEULP/ULBRA)
Caixa Postal 160 – 77.054-970 – Palmas – TO – Brasil

{brunovilar, pfb}@ulbra-to.br

Resumo. *O presente trabalho tem por objetivo descrever o desenvolvimento do L-EditXML, um ambiente interativo que apresenta o processo de verificação da validade de fórmulas da Lógica Quantificacional e as etapas utilizadas para a sua prova. O ambiente está agregado a um provador de teoremas, que utiliza o método dos Tableaux e um algoritmo de unificação na construção das etapas de prova. Dessa forma, a partir da interação com o ambiente, o aluno-usuário poderá verificar a validade de fórmulas lógicas e, especialmente, compreender como a prova foi realizada.*

1. Introdução

Provadores de teoremas são programas de manipulação de símbolos que tentam provar uma determinada fórmula aplicando as regras de inferência da lógica [1]. Além disso, o comportamento do provador de teoremas é determinado, em grande parte, pela base de dados e pelas centenas de heurísticas de controle de uso das regras de inferência, dos axiomas, das definições e de teoremas previamente provados.

Existem algoritmos de prova automática, como o Método de Resolução e o Método dos *Tableaux* [3], que fornecem uma solução semidecidível para a questão "P é conseqüência de T", em que T é uma coleção finita de fórmulas e P é uma fórmula adicional. Esses algoritmos apresentam, ainda, respostas que informam que algumas fórmulas não são conseqüência lógica das premissas.

O provador Alice desenvolvido em [2] trata da verificação da validade de fórmulas e sua conseqüente prova a partir da utilização do Método dos *Tableaux* e de um algoritmo de unificação. As fórmulas que servem de entrada ao provador devem estar armazenadas em documentos XML, daí a necessidade de um editor de fórmulas que permita o manuseio do provador por qualquer usuário, sem a necessidade da construção manual do documento XML.

O entendimento dos métodos usados nas provas de fórmulas da Lógica Quantificacional, que compreende, dentre outros elementos, os quantificadores universal e existencial, requer uma prática constante por parte dos alunos. Muitas vezes, essa prática, se não for acompanhada intensivamente por um professor, tende a ser frustrante, dado o fato da complexidade de prova de algumas fórmulas. Um ambiente como o desenvolvido nesse trabalho, contribuirá para atenuar esse processo, pois

VILAR, Bruno; BRITO, Parcilene Fernandes de. L-EditXML: Ambiente para Manipulação e Apresentação das Etapas de Prova de Fórmulas da Lógica Quantificacional. In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais...** Palmas: 2005.

apresentará ao aluno, de forma interativa, algumas formas de raciocínio para a compreensão da validade de uma fórmula e suas etapas de prova. O sistema permite ao aluno compreender, dentre outros aspectos, como cada elemento da fórmula foi decomposto e utilizado no provador.

Nesse contexto, o objetivo do L-EditXML é ser um ambiente que permite ao usuário editar fórmulas da Lógica Quantificacional, verificar sua validade e suas etapas de prova, a partir de uma interação com o provador Alice. Assim, anteriormente à apresentação dos resultados, é preciso criar as fórmulas, sob o formato de documentos XML, para servir como entrada para o provador. A necessidade de recebimento de fórmulas, e posterior conversão para documentos XML, determinou a criação de um módulo auxiliar ao Editor, o *parser* de fórmulas. Os passos desenvolvidos na construção deste *parser* são relatados na seção 2.2.

Ao submeter uma fórmula ao provador, são obtidos os resultados da prova e um conjunto de valores, ou passos, que foram significativos na obtenção dos resultados. Como o propósito do provador não é o de apresentar de maneira didática os passos trabalhados para a prova, criou-se um recurso adicional ao módulo auxiliar, o *parser* para a árvore de prova, que gera uma esquematização hierárquica dos resultados, demonstrando a seqüência de passos desenvolvida. A representação, criada em uma imagem SVG, permite, ainda, que os elementos utilizados mantenham-se identificados conforme sua representação na fórmula em XML.

Um dos pontos básicos do trabalho é o desenvolvimento de uma interface que permita ao usuário o entendimento de como acontece a verificação da validade de uma fórmula, e, caso a fórmula seja válida, as etapas de raciocínio que definem sua prova. Para isso, na seção 2.3 é apresentada a interface desenvolvida, que agrega as funcionalidades dos módulos empregados.

2. Desenvolvimento do L-EditXML

A construção do L-EditXML abrangeu diferentes tecnologias que foram adotadas por trabalhos previamente construídos ou pela percepção de que seu emprego atendesse aos objetivos necessários. Conforme o desenvolvimento do Provador Alice, o Editor manteve-se utilizando a linguagem Java [4] para a implementação das classes e sua utilização na Web, através do JSP. Dentre os recursos do Java, foram utilizados a API DOM [9], para a manipulação de documentos XML e SVG, e métodos para manipulação de *String* por Expressões Regulares.

O padrão XML [5] permitiu estruturar fórmulas, delinear os elementos que as compõem e verificar se a relação entre os elementos existentes é permitida. Esta forma de representação é flexível no que se refere à aplicação de diferentes fórmulas, porém, mantém-se rígida quanto às regras impostas pela DTD.

A verificação da conformidade das fórmulas com a DTD pôde ser realizada através da API DOM do Java, a qual foi utilizada também para o manuseio dos

VILAR, Bruno; BRITO, Parcilene Fernandes de. L-EditXML: Ambiente para Manipulação e Apresentação das Etapas de Prova de Fórmulas da Lógica Quantificacional. In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais...** Palmas: 2005.

documentos XML e SVG. O *Scalable Vector Graphics* (SVG) consiste em uma linguagem para a descrição de gráficos bidimensionais em XML [6] e como tal foi utilizada para apresentação dos resultados ao usuário, oferecendo recursos de acessibilidade, como o redimensionamento da imagem, e a exibição de elementos dinâmicos à imagem, que é formada apenas por texto.

Ao trabalhar com SVG e JSP, pôde-se promover uma maior interação entre o usuário e a interface com o emprego de recursos DHTML em ambas as tecnologias. O *Dynamic* HTML [7], que não constitui um padrão, mas sim a combinação do HTML (*Hyper Text Markup Language*), CSS (*Cascading Style Sheet*) [8] e do *JavaScript* [10], permite trabalhar com eventos capturados pelas interações com o usuário e modificar os elementos das páginas e imagens.

2.1. A estrutura do L-EditXML

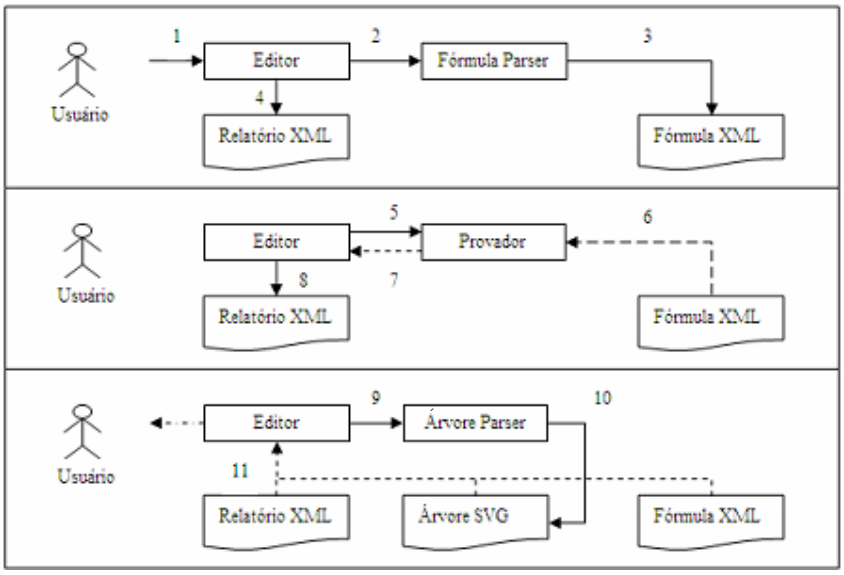

Figura 1. Fluxo de execução apresentado entre os módulos do L-EditXML e o Provador Alice.

A Figura 1 apresenta o fluxo de execução que define a interação entre os módulos do L-EditXML (Editor, Fórmula Parser, Árvore Parser) e o Provador Alice. Através da figura é evidenciada a modularidade dos processos e o compartilhamento de informações através dos arquivos XML. A seqüência apresentada sugere o percurso realizado entre a utilização da interface pelo usuário e a apresentação dos resultados. O processo é iniciado com a entrada dos valores pelo usuário (1). A interface permite, ao usuário, inserir fórmulas através da entrada de caracteres pelo teclado, seguindo as convenções do conjunto de elementos apresentados em uma legenda. Ao ter a fórmula submetida (2), o Editor utiliza o *parser* de fórmulas para verificar se a fórmula está bem formada (do ponto de vista sintático) e, caso seja verificada a boa formação da prova, criar o documento XML (3). Além da criação do documento XML, o *parser* realiza a

reconstrução da fórmula combinando-a com recursos DHTML. Esta reconstrução estabelece um novo recurso de interação entre o usuário e a apresentação, que será explicado em seções posteriores.

Se a criação do documento for realizada com sucesso, o Editor tem a indicação de que a fórmula enviada está corretamente formada, e então é criado um novo documento XML, denominado Relatório (4). Este segundo documento tem como propósito manter, de modo persistente, informações que posteriormente serão apresentadas ao usuário. Após sua criação, o Editor salva a fórmula nas versões original e reconstruída, e passa o fluxo de execução ao Provador Alice (5). A transição entre Editor e Provador pode ocorrer sem que o segundo saiba da existência do primeiro, pois, de acordo com a estrutura criada, basta que seja apresentada uma fórmula bem formada, representada em um documento XML, que a execução do Provador pode ocorrer.

Conforme os passos realizados pelo provador, o documento é lido (6) e, a partir da árvore montada em memória, são aplicados métodos para refinamento e prova. Com a finalização destes procedimentos, a execução retorna ao Editor, que salva as informações referentes à prova no Relatório (8). Tais informações, recuperadas por métodos adicionados ao Alice (7), consistem na validade ou invalidade da fórmula, número de ramos necessários à prova e a representação, por um valor literal, dos passos realizados para se chegar ao resultado. Além de salvar as informações no documento XML, o Editor envia o valor, que representa os passos necessários à prova, ao *parser* da árvore (9), responsável pela transformação das informações provenientes do Provador para um formato que facilite a compreensão do usuário. Dessa forma, o *parser* cria um documento SVG contendo uma apresentação no formato de árvore de prova (10).

Com o término da execução do *parser*, o Editor apresenta, ao usuário, uma nova página, contendo os resultados obtidos (11). Entre estes se encontram um quadro resumo, com os resultados do Provador e a fórmula reconstruída, a fórmula armazenada sob o formato de um documento XML e o documento SVG, exibido ao usuário como uma imagem .

Durante todo o processo, cada etapa possui um conjunto de regras de verificação da conformidade dos dados e do fluxo de execução. Se em uma determinada etapa é verificado que ocorreu uma exceção às regras, a execução do Editor pára e é iniciado o processo de construção da mensagem de erro ao usuário. Caso a exceção tenha ocorrido por erro nos valores fornecidos pelo usuário, a mensagem de erro é construída apresentando o valor fornecido, a descrição do problema ocorrido e, se possível, o ponto específico, da entrada, em que houve o problema.

2.2 Etapas para a Definição do *Parser*

Para permitir que os valores fornecidos pelo usuário fossem aplicados ao Provador Alice, construiu-se um *parser* para as fórmulas. O *parser* tem como finalidade reconhecer os elementos das fórmulas, verificar sua correta formação e então representá-los em um documento XML.

A estrutura do *parser* foi definida de forma que as ações de reconhecimento e extração fossem realizadas por regras. Estas foram criadas pelos recursos oferecidos pelo pacote *java.util.regex*[1], que oferece métodos que trabalham com Expressões Regulares. Como cada elemento é representado por uma regra distinta, novos elementos podem ser inseridos sem alterar a estrutura da ferramenta.

A gramática, que serve como base para a análise léxica, foi definida pelos elementos que compõem a Lógica Quantificacional, que compreendem:

- literais: o conjunto de letras maiúsculas, como "P", "Q", "R" e "S";
- variáveis: as letras minúsculas de "a" a "t", podendo obter variações a partir de números, como "b¹", "b²" etc;
- conectivos lógicos: Negação, Conjunção, Disjunção, Condicional e Bicondicional, respectivamente representados por "~", "^", "\/", "->" e "<->";
- quantificadores: Universal e Existencial, representados por "U" e "E".

O *parser* trabalha de forma independente com os enunciados que compõem a fórmula, que podem ser uma premissa ou a conclusão. Dado um enunciado, o elemento mais externo é reconhecido e então removido da fórmula original, para que sua representação seja inserida em um vetor. Posteriormente o elemento que passou a ser o mais externo é reconhecido e levado ao mesmo procedimento. A forma de representação dos elementos, dentro dos vetores, segue a nomenclatura dos elementos da DTD. A Figura 2 apresenta o processo de análise e construção da matriz de elementos a partir de uma fórmula submetida.

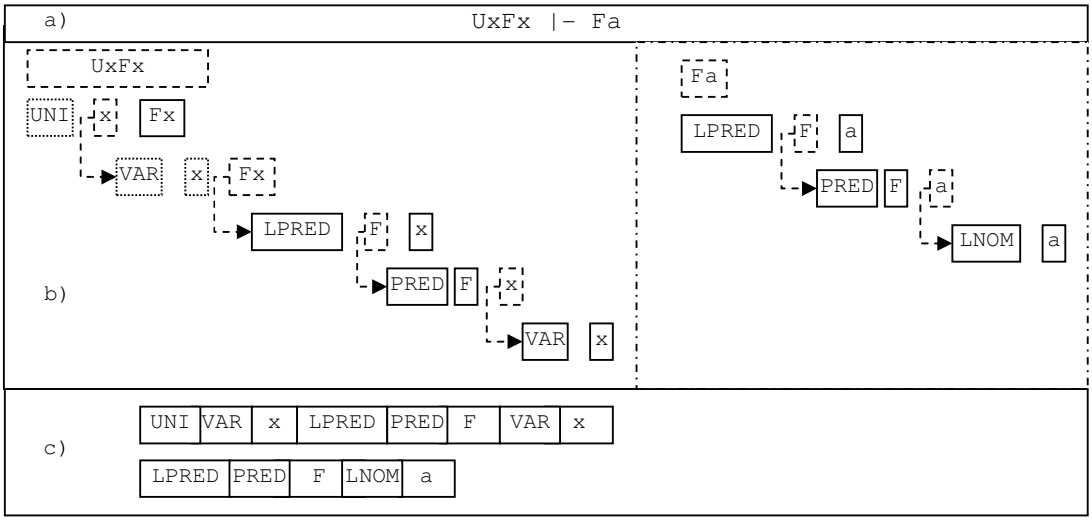

Figura 2. Análise sintática da fórmula.

Conforme apresentado na Figura 2, dada uma fórmula submetida para análise (a), cada enunciado é trabalhado de forma isolada, passando pelo procedimento de

[1] Fornecido pela API do Java em: http://java.sun.com/j2se/1.4.2/download.html

reconhecimento e extração de cada elemento mais externo (b). Deve ser considerado que determinados elementos não são extraídos assim que o primeiro reconhecimento é obtido, como ocorre para o elemento "*Fx*" (parte da premissa "UxFx"), que primeiro é decomposta a premissa para que cada termo seja analisado de forma independente. Após a conclusão da análise de todos os elementos, têm-se os vetores que correspondem a cada enunciado da fórmula, composta por uma premissa e uma conclusão (c). A disposição dos elementos dentro dos vetores corresponde à composição de uma árvore em pré-ordem, quando estes são acessados em seqüência.

Estabelecida a análise dos elementos e obtidos os vetores correspondentes aos enunciados decompostos, realizou-se a síntese da fórmula para o formato de entrada esperado pelo Provador Alice. Para isto foi realizado um percurso seqüencial através dos vetores internos e, a medida em que os elementos foram lidos, foram inseridos os elementos correspondentes em um documento XML. Estas inserções adotaram convenções e elementos pré-determinados pela DTD (Figura 2 – A), sendo que a API DOM do Java proveu os recursos necessários à criação e ao manuseio dos documentos XML. Na Figura 2 (B), é apresentado também o resultado da criação do documento XML .

A

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

B

```
<ARG>
    <PREM>
        <UNI ID="1" NEG="">
            <VAR>x</VAR>
             <LPRED ID="2" NEG="">
                <PRED>F</PRED>
                <VAR>x</VAR>
             </LPRED>
        </UNI>
    </PREM>
    <CONC>
       <LPRED ID="3" NEG="">
          <PRED>F</PRED>
          <VAR>x</VAR>
       </LPRED>
   </CONC>
</ARG>
```

Figura 3. Parte da Representação da fórmula através de um documento XML e seu DTD.

Conforme apresentado na Figura 3, a ordem do aninhamento existente na fórmula original, e posteriormente no conjunto de vetores, é mantida pelo documento XML. Com a criação do documento, pode-se realizar uma análise do documento a partir do *parser* fornecido pela API do DOM. De forma complementar, a API do SAX facilitou o reconhecimento dos erros encontrados no documento, ou seja, a verificação da má formação da fórmula submetida pelo usuário. Com a capacidade de reconhecer os erros e apontar o ponto incorreto, pôde-se auxiliar a criação das fórmulas exibindo mensagens de erro ao usuário com recomendações da utilização do elemento.

Dado o propósito inicial do *parser*, após a criação do documento XML e sua validação de acordo com a DTD especificada, o controle de execução poderia passar ao Provador Alice, obtendo as informações da validade da fórmula. Entretanto, prevendo a possibilidade de uma interação maior entre o usuário e a árvore de apresentação das etapas de prova, criou-se a perspectiva de reconstruir a fórmula incluindo novos elementos.

A reconstrução envolveu os fragmentos da fórmula original, que foram reconhecidos pelo *parser* e armazenados no vetor, e posteriormente foram combinados com recursos DHTML. Tais recursos, que incluem marcações HTML, Java *Script* e CSS, permitem ao usuário ter a percepção de qual termo da fórmula deu origem a um determinado elemento da árvore. A Figura 4 apresenta a fórmula reconstruída com elementos DHTML.

```
1.  <SPAN id='f0'>Ux<SPAN id='f1'>Fx</SPAN></SPAN> |- <SPAN id='f2'>Fa</SPAN>
```

Figura 4. Fórmula reconstruída com a adição de elementos DHTML.

Como demonstrado pela Figura 4, os elementos originais da fórmula são preservados, porém, são envolvidos por marcações HTML (*<SPAN></SPAN>*) que fazem referência a elementos pertencentes ao Documento XML, Folhas de Estilo (CSS) e Java *Script* (Figura 4). Os atributos, cujos números são precedidos da letra 'f', correspondem aos identificadores criados para a fórmula em XML, mantendo relação entre estes elementos, que posteriormente são evidenciados para o usuário. Os recursos DHTML são a base para a exibição de mensagens detalhadas ao usuário ou a adição de elementos e eventos que auxiliem no entendimento das etapas de prova.

2.3 Apresentação

A página de apresentação foi desenvolvida de forma que o usuário possa visualizar as informações necessárias ao entendimento dos símbolos e elementos que compõem as fórmulas trabalhadas e posteriormente possa submeter uma fórmula à prova. A página é apresentada na Figura 5.

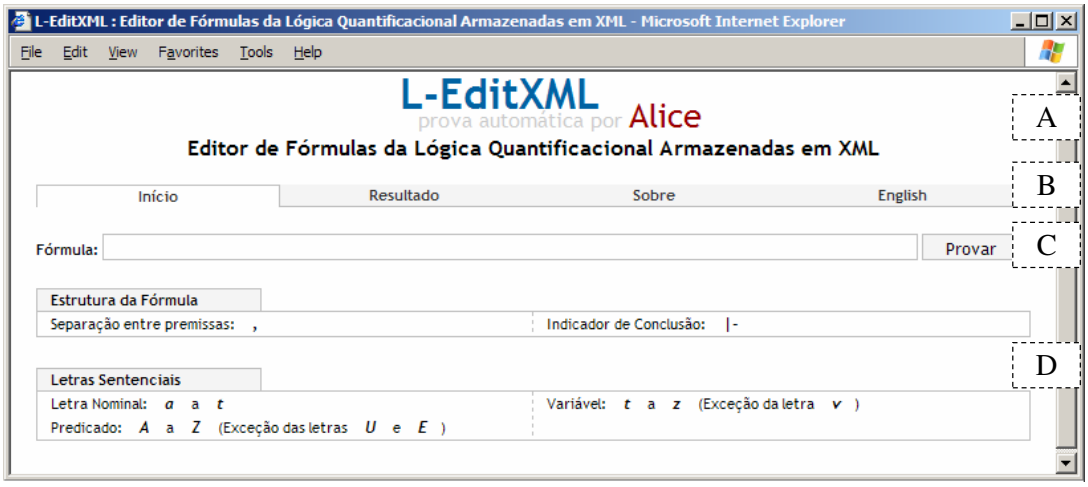

Figura 5. Página para a submissão de fórmula.

De acordo com a Figura 5, a interface estabelece um cabeçalho padrão para as páginas (A) e um menu (B) que permite ao usuário navegar entre as páginas e, caso uma fórmula tenha sido submetida, não perder as informações das demais páginas durante a navegação. A submissão de fórmulas pode ser realizada logo abaixo do menu, em um formulário composto por uma caixa de texto e um botão de confirmação (C). As informações necessárias ao entendimento dos símbolos e elementos que compõem as fórmulas trabalhadas são apresentadas em (D). A seguir, na Figura 6, é apresentado um exemplo de mensagem que é exibida ao usuário caso a fórmula esteja construída incorretamente.

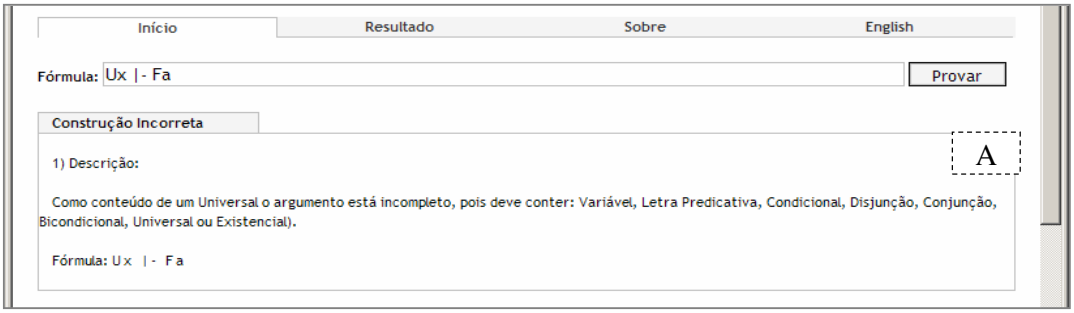

Figura 6. Mensagem de erro correspondente à formula submetida.

Conforme o propósito do ambiente, a apresentação dos erros também se mostra importante durante o aprendizado. Dessa forma, a estrutura da página permite que seja exibida, ao usuário, a mensagem de erro (A) correspondente à fórmula submetida (Figura 6). Assim que a fórmula é validada sintaticamente através do *parser*, o usuário tem acesso à página de apresentação dos resultados (Figura 7).

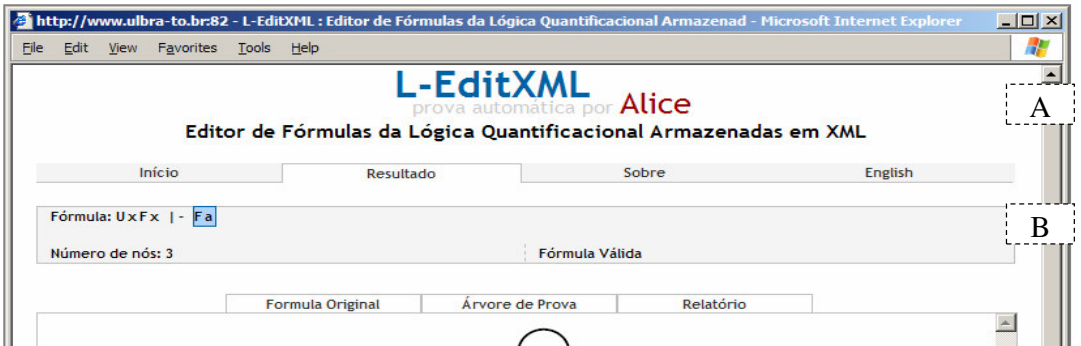

Figura 7. Página de apresentação dos resultados e da árvore de prova.

A página de apresentação é formada pelo cabeçalho e menu padrões (A), por um painel de apresentação (B), contendo um resumo dos resultados da prova, e um conjunto de guias (C), que fornece acesso às informações detalhadas da fórmula e da prova (Figura 7). O quadro de resumo exibe a fórmula reconstruída, o número de nós necessários à prova e, caso o provador tenha efetuado com êxito a verificação da validade da fórmula, a informação se esta é válida ou inválida. O conjunto de guias fornece a possibilidade de que seja exibido o documento XML, pelo qual a fórmula foi armazenada; o relatório, mantendo de forma persistente as informações da prova; e o documento SVG, com a árvore de prova.

A árvore permite ao usuário selecionar os nós para que seus equivalentes sejam evidenciados na fórmula reconstruída. Dessa forma, é facilitado o processo de entendimento dos passos realizados pelo provador para que a prova fosse alcançada. Outro recurso que auxilia o acompanhamento dos passos traçados são as possibilidades de redimensionar, mover e realizar a busca por texto dentro da imagem da árvore. A Figura 8 apresenta parte dos recursos de acessibilidade.

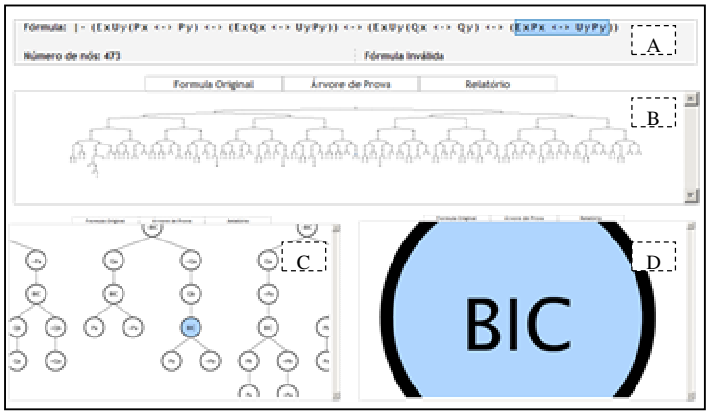

Figura 8. Aplicação de diferentes focos sobre a mesma.

VILAR, Bruno; BRITO, Parcilene Fernandes de. L-EditXML: Ambiente para Manipulação e Apresentação das Etapas de Prova de Fórmulas da Lógica Quantificacional. In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais...** Palmas: 2005.

Como nem todas as fórmulas podem ter sua prova realizada através de uma quantidade reduzida de ramos, a possibilidade de mudar as dimensões das árvores e seus nós mostra-se útil para o melhor acompanhamento da árvore e o seu conseqüente entendimento. A exemplo disto, a Figura 8 apresenta três diferentes perspectivas da mesma árvore, composta por 473 nós. A primeira perspectiva (B) fornece a visão da árvore completa, a partir dela o usuário pode determinar o ponto de onde começará a analisar a árvore. A segunda perspectiva (C) exibe a aproximação de uma área da árvore, permitindo o acompanhamento mais preciso dos nós. A terceira perspectiva (D) apresenta a capacidade máxima de aproximação sobre a árvore criada, trazendo a percepção de que a qualidade da imagem não é afetada por sua aproximação ou distanciamento, que é uma característica inerente às imagens vetoriais como o SVG. Durante as diferentes ações sobre a imagem, os nós que foram selecionados não deixam de manter sua evidência na árvore ou dentro da fórmula, como pode ser percebido nas visões da árvore (B, C e D) e no quadro resumo (A).

3 Conclusão

O trabalho apresentado teve por objetivo o desenvolvimento de um ambiente para auxiliar no estudo da Lógica, mais especificamente no que tange à verificação da validade de fórmulas e prova de teoremas a partir da utilização do método dos *Tableaux*. Para tanto, foi criado o L-EditXML, que através do provador Alice, apresenta um conjunto de informações referentes às fórmulas inseridas pelo usuário, de modo que este possa entender a aplicação das regras de um método automático de prova, como o *Tableaux*, e verificar as etapas de raciocínio usadas para a prova do teorema ou a verificação da invalidade da fórmula. A interação que é possível realizar na árvore de prova faz com que o aluno compreenda como os elementos são decompostos e como podem ser usados nas etapas de prova. Assim, conceitos complexos como a instanciação das variáveis agregadas aos quantificadores Existencial e Universal, por exemplo, podem ser observados de forma interativa.

Mantendo o diferencial do trabalho realizado em [2], para a construção do Alice, sobretudo pela utilização do XML, para o armazenamento de fórmulas, e do Java, para a manipulação dos documentos XML, o L-EditXML trabalhou com tecnologias comuns às utilizadas, como o JSP e o SVG. Por seu uso, o Java forneceu, ainda, recursos que facilitaram a construção do *parser*, como as Expressões Regulares e os *parsers* para XML, como o DOM e o SAX. O emprego do SVG mostrou-se adequado pelo uso comum dos recursos aplicados ao manuseio das fórmulas em XML, além de apresentar, de forma inerente, recursos de acessibilidade, que permitiram a criação de resultados que podem ser melhor explorados pelo usuário.

O desenvolvimento do *parser* para as fórmulas permitiu abranger os diferentes elementos envolvidos nas fórmulas e elaborar mensagens de erro que buscam auxiliar o usuário a reconhecer o problema e encontrar alguma informação relevante à correção. Com o resultado da análise do *parser*, foi possível fornecer valores corretamente estruturados para o provador e sintetizar uma segunda saída, semelhante à entrada do usuário, porém, aproveitando as informações extraídas pelo *parser* para promover

VILAR, Bruno; BRITO, Parcilene Fernandes de. L-EditXML: Ambiente para Manipulação e Apresentação das Etapas de Prova de Fórmulas da Lógica Quantificacional. In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais...** Palmas: 2005.

possíveis interações entre os valores de entrada, o usuário e os demais resultados obtidos.

4 Referências

1. BOYER, M. Kaufmann B.; MOORE, J.S. The Boyer-Moore Theorem Provers and It's Interactive Enhancement. Computers and Mathematics with Applications, Vol. 29, No. 2, pp. 27-62 (1995).
2. BRITO, P. F. "Dedução Automática por Tableaux Estruturada em Xml", Dissertação de Mestrado – UFSC (2003).
3. REEVES, 1983. REEVES. S. V. An Introduction to Semantic Tableaux. Departament of Computer Science. CMS-55, University of Essex, 1983.
4. SUN, 2004. Sun Microsystems. "Java TM 2 Platform, Standard Edition, v 1.5 API Specification 2004", http://java.sun.com/apis/j2se/1.5.0/docs/api/, Julho (2005).
5. W3C. "W3C – World Wide Web Consortium. Extensible Markup Language (XML) 1.1 Specification", http://www.w3.org/XML/, Julho (2005).
6. W3C. "W3C – World Wide Web Consortium. Scalable Vector Graphics (SVG) 1.1 Specification", http://www.w3.org/TR/SVG/, Fevereiro (2003).
7. W3C. "W3C – World Wide Web Consortium. HyperText Markup Language (HTML)", http://www.w3.org/Markup/, Julho (2005).
8. W3C. "W3C – World Wide Web Consortium. Cascading Style Sheet (CSS) level 1", http://www.w3.org/TR/CSS1, Julho (2005).
9. W3C. "W3C – World Wide Web Consortium. Document Object Model (DOM)", http://www.w3.org/dom, Julho (2005).
10. W3C. "W3C – World Wide Web Consortium. JavaScript Tutorial", http://www.w3schools.com/js, Julho (2005).

VILAR, Bruno; BRITO, Parcilene Fernandes de. L-EditXML: Ambiente para Manipulação e Apresentação das Etapas de Prova de Fórmulas da Lógica Quantificacional. In: VII ENCONTRO DE ESTUDANTES DE INFORMÁTICA DO ESTADO DO TOCANTINS, 2005, Palmas. **Anais...** Palmas: 2005.