UNIVERSIDADE FEDERAL FLUMINENSE
INSTITUTO DE COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

LUCAS DE ANDRADE

ASSISTENTE PARA CONSTRUÇÃO DE PROVAS PARA LÓGICA PROPOSICIONAL EM DEDUÇÃO NATURAL

NITERÓI
2008

LUCAS DE ANDRADE

ASSISTENTE PARA CONSTRUÇÃO DE PROVAS PARA LÓGICA PROPOSICIONAL EM DEDUÇÃO NATURAL

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração: Interface Homem-Computador.

Orientador: Prof. Marcelo da Silva Corrêa
Co-Orientador: Prof.ª Teresa Cristina de Aguiar

NITERÓI
2008

LUCAS DE ANDRADE

ASSISTENTE PARA CONSTRUÇÃO DE PROVAS PARA LÓGICA PROPOSICIONAL EM DEDUÇÃO NATURAL

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração: Interface Homem-Computador.

Aprovada em dezembro de 2008.

BANCA EXAMINADORA

Prof. MARCELO DA SILVA CORRÊA – Orientador
UFF

Prof.ª TERESA CRISTINA DE AGUIAR – Co-Orientador
UFF

Prof. LEONARDO GRESTA PAULINO MURTA
UFF

NITERÓI
2008

AGRADECIMENTOS

À Universidade Federal Fluminense,

Ao Professor Marcelo da Silva Corrêa,

À Professora Teresa Cristina de Aguiar,

A todos os professores com quem convivi e de quem tive o privilégio de ser aluno no Curso de Graduação em Ciência da Computação,

Aos colegas e amigos que conheci durante toda a faculdade,

A minha família por sempre ter me apoiado em todos os momentos.

RESUMO

O projeto consiste no desenvolvimento de uma interface para um provador-assistente para a Lógica Proposicional baseado no método de dedução natural. A partir da análise de sistemas atuais e considerando critérios para a elaboração de interfaces para assistentes de construção de provas e provadores, tem-se o objetivo de conceber uma interface amigável para um sistema que viabilize um ambiente de edição de provas. O sistema deve permitir que o usuário consiga provar uma fórmula a partir de um conjunto de fórmulas da lógica proposicional, utilizando as regras de introdução e de eliminação do método de dedução natural. Um dos principais desafios do projeto é permitir a visualização de uma prova no formato de árvore.

Palavras-chave: Assistente para Construção de Provas, Dedução Natural, Lógica Proposicional

ABSTRACT

This project consists in the development of a Graphical User Interface (GUI) for a proof assistant for the Propositional Classical Logic based on Natural Deduction method. The goal is to design a friendly GUI for a proof editor. The system has to let the user prove a formula from a set of formula of the language of the Propositional Logic, using the introduction and elimination rules provided by the natural deduction method. One of the main challenges of this project is to show the proof in a tree form, commonly used with the natural deduction method.

Keywords: Proof Assistant, Natural Deduction, Propositional Logics

SUMÁRIO

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

1 INTRODUÇÃO

A Lógica Proposicional é um sistema formal cujas expressões bem formadas, as fórmulas, representam proposições e que pode ter uma noção de prova (demonstração) ou critérios para a classificação de fórmulas quanto à sua validade ou satisfabilidade.

Nos anos 30, foram apresentados por Gentzen e Jaśkowski, separadamente, sistemas de dedução em que as demonstrações são organizadas em árvores, chamadas de árvores de derivação. Os sistemas propostos por Gentzen para a Lógica Clássica, Proposicional e de 1ª. Ordem, foram chamados de sistemas de Dedução Natural, porque suas regras buscam representar as formas simples de inferência utilizadas pelos matemáticos.

Com o surgimento dos computadores eletrônicos por volta dos anos 50, iniciaram-se as pesquisas sobre a prova automática de teoremas. A tentativa de implementar procedimentos para realizar provas automaticamente não obteve tanto exito durante os primeiros anos pelo fato de não haver estruturas que dessem suporte a complexidade de tais algoritmos.

Alguns anos mais tarde, o paradigma de programação lógica e o desenvolvimento das linguagens funcionais, viabilizou a implementação de programas computacionais que realizassem provas automaticamente. Com a evolução dessas ferramentas, os resultados obtidos se tornaram cada vez melhores.

Atualmente, existem alguns programas computacionais que tratam vários tipos de lógica, possibilitando até mesmo a construção de um novo tipo, em que o usuário define as regras. Alguns desses programas são provadores automáticos, enquanto outros oferecem apenas ambientes de construção de prova, deixando que o usuário indique as regras a serem utilizadas. Alguns desses ambientes também oferecem assistência, como, por exemplo, indicando quais regras que podem ou não podem ser aplicadas em determinado momento.

A existência de programas computacionais que viabilizem apenas um editor de prova é justificada pelo fato de que a prova interativa de problemas, assistida por algum módulo ou programa de verificação tornou-se uma boa solução para os problemas da prova automática de teoremas. Isso se deve ao fato de que muitas vezes, dependendo da complexidade do problema, o provador pode não encontrar uma solução, necessitando de intervenções de um usuário.

O estudo da interface para provadores automáticos e assistentes para construção de provas tornou-se uma necessidade com o passar dos anos, principalmente ao se notar a deficiência de provadores automáticos para encontrar soluções para problemas mais complexos.

Com base em alguns conceitos de Interface Homem-Computador (IHC), novos parâmetros de análise de interfaces foram formalizados especificamente para esse tipo de programa computacional como é proposto por (Merriam & Harrison, Evaluating the Interfaces of Three Theorem Proving Assistants, 1996). Também foram apontados problemas no projeto de interfaces como é possível ver em (Völker, 2003).

A busca por aprimorar as interfaces para provadores automáticos e assistentes para construção de provas ganhou tamanha importância que em julho de 1995 ocorreu o primeiro workshop internacional sobre o tema, denominado *User Interface for Theorem Provers* (UITP). Estes encontros ocorreram ao longo dos anos na Europa, sendo o último realizado em 2005 em Edinburgh na Escócia.

Tendo como base esses conceitos, desenvolveu-se esse trabalho com a finalidade de aprofundar os conhecimentos a respeito da prova interativa e automática de teoremas, analisar e comparar alguns programas computacionais utilizados atualmente e por fim propor uma nova interface para um assistente para construção de provas. A finalidade é de aprimorar o processo de interação do usuário elaborando uma programa computacional e uma interface que viabilize um ambiente de construção de provas.

Após um levantamento dos principais provadores automáticos e assistentes para construção de provas, foi realizado um estudo das interfaces dos provadores. Com uma fundamentação na área de IHC e um estudo específico de interfaces para programas computacionais desse tipo, obteve-se uma tabela comparativa que realçou a falta de recursos disponibilizados pelas interfaces nos ambientes de edição de prova utilizados atualmente.

Iniciou-se, então, a elaboração de uma proposta de um novo programa que atendesse requisitos fundamentais para um ambiente de edição de provas, até então, não explorados pelos provadores e assistentes. O principal desafio era construir uma interface baseada em manipulação gráfica e menus e que também apresentasse a estrutura da prova no formato de árvore de derivação.

Adotando-se um processo de engenharia de software iterativo, foi proposto como primeiro *entregável* a interface do assistente para construção de provas para Lógica Proposicional em Dedução Natural com algumas funcionalidades básicas como

por exemplo: aplicação de regras de Dedução Natural, construção da prova no formato de árvore de derivação e validação da sintaxe de uma conjectura baseado no alfabeto da Lógica Proposicional.

Neste trabalho está descrito todo o processo de pesquisa realizado, bem como a documentação gerada durante o processo de engenharia de software e implementação do primeiro *entregável* do projeto.

As próximas seções tratarão portanto dos quatro tópicos abordados no trabalho: fundamentação em lógica, estudo sobre conceitos de IHC e parâmetros para análise de interfaces de provadores, análise do estado da arte e proposta de um assistente para construção de provas.

2 LÓGICA PROPOSICIONAL E O MÉTODO DE DEDUÇÃO NATURAL

Neste capítulo será feita uma breve descrição sobre a Lógica Proposicional e o método de Dedução Natural. Esses conceitos são fundamentais para o entendimento de como um provador automático ou um assistente para construção de provas funcionam.

2.1 A LÓGICA PROPOSICIONAL

A definição da linguagem da Lógica Proposicional é feita definindo-se um alfabeto e um conjunto de regras para geração de fórmulas. Assim como palavras são formadas por regras específicas na língua portuguesa, as fórmulas também têm suas regras de formação. Posteriormente a semântica ou a interpretação dessas fórmulas será abordada.

Inicia-se então a descrição da sintaxe da Lógica Proposicional:

Para o caso de um programa computacional, deve-se definir um alfabeto considerando as limitações de escrita e, portanto nem todos os símbolos serão idênticos aos utilizados pelos métodos tradicionais de ensino. Segue o alfabeto da Lógica Proposicional definido para o assistente:

- Símbolos de Pontuação: (,).
- Símbolos de Verdade: T, F.
- Símbolos Proposicionais: P,Q,R,S,P1,Q1,R1,S1,..., PN,...,...,SN,....
- Conectivos Proposicionais: $\wedge$, $\vee$, $\sim$, $\rightarrow$, $\leftrightarrow$.
- Símbolo: $\vdash$.

Considerando esse alfabeto, o conjunto de fórmulas da Lógica Proposicional deve ser formado pela concatenação de tais símbolos. Tal formação de fórmulas é determinada pela seguinte definição por indução estrutural:

- Todo Símbolo de Verdade é uma fórmula.
- Todo Símbolo Proposicional é uma fórmula.
- Se α é uma fórmula então $(\sim\alpha)$, a negação de α, é uma fórmula.

- Se α e β são fórmulas, então $(\alpha \vee \beta)$ é uma fórmula. Esta fórmula é a disjunção das fórmulas α e β.
- Se α e β são fórmulas, então $(\alpha \wedge \beta)$ é uma fórmula. Esta fórmula é a conjunção das fórmulas α e β.
- Se α e β são fórmulas, então $(\alpha \rightarrow \beta)$ é uma fórmula. Neste caso, α é o antecedente e β o conseqüente da fórmula $(\alpha \rightarrow \beta)$.
- Se α e β são fórmulas, então $(\alpha \leftrightarrow \beta)$ é uma fórmula. Neste caso, α é o lado esquerdo e β o lado direito da fórmula $(\alpha \leftrightarrow \beta)$.
- Nada mais é formula.

Além disso, assim como os operadores básicos da matemática, podemos adotar uma ordem de precedência para os conectivos lógicos no processo de decomposição estrutural de uma fórmula (por meio da identificação de operadores e seus operandos). Isto é feito a fim de simplificar a escrita pela eliminação de pares de parênteses, mas mantendo uma leitura única (decomposição estrutural única) para as fórmulas:

- Maior precedência: $\rightarrow$, $\leftrightarrow$.
- Precedência intermediária: $\wedge$, $\vee$.
- Menor precedência: ~.

Assim, a fórmula $((P \wedge Q) \rightarrow ((\sim (P \vee (Q \wedge R))) \vee S))$ pode ser escrita como $P \wedge Q \rightarrow \sim (P \vee (Q \wedge R)) \vee S$.

Segue também a definição de alguns elementos sintáticos que podem ser necessários ou úteis durante a implementação do assistente:

- O comprimento de uma fórmula é definido da seguinte maneira:
    - Se α é um símbolo proposicional ou de verdade então $comp[\alpha] = 1$.
    - Se α e β são fórmulas então:
        - $Comp[\sim\alpha] = comp[\alpha] + 1$
        - $Comp[\alpha \vee \beta] = comp[\alpha] + comp[\beta] + 1$
        - $Comp[\alpha \wedge \beta] = comp[\alpha] + comp[\beta] + 1$

        - Comp[$\alpha \rightarrow \beta$] = comp[α] + comp[β] + 1
        - Comp[$\alpha \leftrightarrow \beta$] = comp[$\alpha$] + comp[$\beta$] + 1
- Uma subfórmula de uma fórmula α é definida da seguinte forma:
    - α é uma subfórmula de α
    - Se $\alpha = (\sim\beta)$, então β é uma subfórmula de α.
    - Se α é uma fórmula do tipo: $(\beta \vee \gamma)$, $(\beta \wedge \gamma)$, $(\beta \rightarrow \gamma)$, $(\beta \leftrightarrow \gamma)$, então β e γ são subfórmulas de α.
    - Se β é uma subfórmula de α, então toda subfórmula de β é subfórmula de α.

Tendo sido apresentada a sintaxe da Lógica Proposicional, descrevemos a seguir a sua semântica:

"O significado ou semântica dos elementos sintáticos da Lógica Proposicional clássica é determinado por uma função I denominada Interpretação. Esta função associa a cada fórmula da Lógica Proposicional um valor de verdade "verdadeiro" ou "falso", que é representado respectivamente por T e F." (Souza, 2002).

Considerando isso, dadas uma fórmula α e uma interpretação I, então o significado de α indicado por I[α] é determinado pelas regras a seguir:

- Se α = P, onde P é um símbolo proposicional, então I[α] $\in$ {T, F}.
- Se α = Verdadeiro, então I[α] = I[Verdadeiro] = T.
- Se α = Falso, então I[α] = I[Falso] = F.
- Seja β uma fórmula. Se $\alpha = \sim\beta$, então:
    - I[α] = I[$\sim\beta$] = T, se, e somente se, I[β] = F,
    - I[α] = I[$\sim\beta$] = F, se, e somente se, I[β] = T.
- Sejam β e γ duas fórmulas. Se $\alpha = (\beta \vee \gamma)$, então:
    - I[α] = I[$\beta \vee \gamma$] = T, se, e somente se, I[β] = T ou I[γ] = T,
    - I[α] = I[$\beta \vee \gamma$] = F, se, e somente se, I[β] = F e I[γ] = F,
- Sejam β e γ duas fórmulas. Se $\alpha = (\beta \wedge \gamma)$, então:
    - I[α] = I[$\beta \wedge \gamma$] = T, se, e somente se, I[β] = T e I[γ] = T,
    - I[α] = I[$\beta \wedge \gamma$] = F, se, e somente se, I[β] = F ou I[γ] = F,
- Sejam β e γ duas fórmulas. Se $\alpha = (\beta \rightarrow \gamma)$, então:

- $I[\alpha] = I[\beta \rightarrow \gamma] = T$, se, e somente se, $I[\beta] = F$ ou $I[\gamma] = T$,
- $I[\alpha] = I[\beta \rightarrow \gamma] = F$, se, e somente se, $I[\beta] = T$ e $I[\gamma] = F$,

- Sejam β e γ duas fórmulas. Se $\alpha = (\beta \leftrightarrow \gamma)$, então:
  - $I[\alpha] = I[\beta \leftrightarrow \gamma] = T$, se, e somente se, $I[\beta] = I[\gamma]$,
  - $I[\alpha] = I[\beta \leftrightarrow \gamma] = F$, se, e somente se, $I[\beta] \neq I[\gamma]$,

A seguir estão definidas algumas propriedades semânticas que relacionam os resultados das interpretações de fórmulas:

- Uma fórmula é uma tautologia ou é válida se, e somente se, para toda interpretação I, $I[\alpha] = T$. Representa-se: $\models \alpha$.
- Uma fórmula α é factível ou satisfatível se, e somente se, existe pelo menos uma interpretação I, tal que $I[\alpha] = T$.
- Uma fórmula α é uma contradição se, e somente se, para toda interpretação I, $I[\alpha] = F$.
- Dadas duas fórmulas α e β, α implica β se, e somente se, para toda interpretação I, se $I[\alpha] = T$ então $I[\beta] = T$.
- Dadas duas fórmulas α e β, α equivale a β se, e somente se, para toda interpretação I, $I[\alpha] = I[\beta]$.
- Dada uma fórmula α e uma interpretação I, então I satisfaz α se, e somente se, $I[\alpha] = T$.
- Um conjunto de fórmulas $\Gamma = \{\alpha 1, \alpha 2, ..., \alpha N, ...\}$ é satisfazível se, e somente se, existe uma interpretação I, tal que $I[\alpha 1] = I[\alpha 2] = ... = I[\alpha N] = T$ .... Neste caso, diz-se que I satisfaz o conjunto de fórmulas, o que é indicado por $I[\Gamma] = T$.
- Dado um conjunto de fórmulas vazio, então toda interpretação I satisfaz esse conjunto.
- Dadas uma fórmula α e um conjunto de fórmulas Γ, Γ implica α se, e somente se, para toda interpretação I, se I satisfaz Γ, então $I[\alpha] = T$. Isto é representado por: $\Gamma \models \alpha$

2.2 O MÉTODO DE DEDUÇÃO NATURAL

O sistema de Dedução Natural para Lógica Proposicional é composto pelos seguintes elementos:

- O alfabeto da Lógica Proposicional;
- O conjunto de fórmulas da Lógica Proposicional, e
- Um conjunto de regras de dedução; e
- Uma noção de dedução (prova).

Como o alfabeto e o conjunto de fórmulas da Lógica Proposicional já foram abordados anteriormente, resta apenas a explicação do conjunto de regras de dedução.

Para representar as regras de inferência desse sistema, adotou-se uma notação. As fórmulas acima do traço são as premissas, enquanto as que estão abaixo do traço são as conclusões. Existem ainda as hipóteses, que são assumidas durante a construção da prova, e são consideradas premissas também. Usualmente essas são destacadas na prova utilizando colchetes.

A seguir está o conjunto das regras de dedução:

$\dfrac{\alpha \wedge \beta}{\alpha}$	$\dfrac{\alpha \wedge \beta}{\beta}$	$\dfrac{\alpha \quad \beta}{\alpha \wedge \beta}$
Eliminação - $\wedge$ - 1	Eliminação - $\wedge$ - 2	Introdução - $\wedge$
$\dfrac{\alpha}{\alpha \vee \beta}$	$\dfrac{\beta}{\alpha \vee \beta}$	$\dfrac{\begin{array}{cc}[\alpha] & [\beta] \\ \vdots & \vdots \\ \alpha \vee \beta & \gamma \quad \gamma\end{array}}{\gamma}$
Introdução - $\vee$ - 1	Introdução - $\vee$ - 2	Eliminação - $\vee$
$\dfrac{\alpha \quad \alpha \rightarrow \beta}{\beta}$	$\dfrac{\begin{array}{c}[\alpha] \\ \vdots \\ \beta\end{array}}{\alpha \rightarrow \beta}$	$\dfrac{\sim\sim\alpha}{\alpha}$
Eliminação - $\rightarrow$	Introdução - $\rightarrow$	Eliminação - ~

$\begin{array}{c}[\beta] \\ \vdots \\ \frac{F}{\sim\beta}\end{array}$	$\frac{F}{\alpha}$	$\frac{\alpha \quad \sim\alpha}{F}$
Introdução - ~	Eliminação – F	Introdução - F
$\frac{\alpha \leftrightarrow \beta}{\alpha \rightarrow \beta}$	$\frac{\alpha \leftrightarrow \beta}{\beta \rightarrow \alpha}$	$\frac{\alpha \rightarrow \beta \quad \beta \rightarrow \alpha}{\alpha \leftrightarrow \beta}$
Eliminação - <-> - 1	Eliminação - <-> - 2	Introdução - <->

Tabela 1 - Regras de Inferência do Sistema de Dedução Natural

Com a aplicação das regras listadas acima sobre uma fórmula, é possível obter uma árvore de derivação, ou simplesmente, derivação.

Uma prova no sistema de Dedução Natural pode ser definida formalmente da seguinte maneira:

Seja α uma fórmula e Γ um conjunto de fórmulas denominado por premissas. Uma dedução (prova) de α a partir de β no sistema de Dedução Natural é uma derivação com as seguintes características:

- As regras de derivação são aplicadas tendo como premissas iniciais, fórmulas de Γ ou fórmulas que são consideradas como hipóteses intermediárias, cuja utilização é permitida pela aplicação de certas regras de derivação, como Eliminação - $\vee$, Introdução - $\rightarrow$ e Introdução - ~..
- A última fórmula da derivação, ou seja, a conclusão obtida após a aplicação da última regra de inferência é α.

Considerando a metodologia utilizada para construção de uma prova, é possível verificar se uma conjectura está provada a partir da verificação se ainda resta alguma fórmula pertencente a árvore de derivação que deve ser provada.

Em outras palavras, define-se para cada momento da prova, um conjunto de fórmulas que precisam ser provadas. Quando esse conjunto estiver vazio, significa que não há mais fórmulas para se provar. Com isso, resta apenas realizar uma verificação das hipóteses utilizadas, se elas são premissas iniciais ou hipóteses intermediárias

introduzidas e descartadas a partir de aplicação de algumas das regras destacadas acima..

Caso todas as alternativas de aplicações de regras de inferência tenham sido testadas e o conjunto de conclusões não esteja vazio, significa que não existe uma prova para aquela conjectura.

Essa idéia para verificação do fechamento de uma prova poderá ser utilizada durante a implementação de um procedimento que faça essa verificação, considerando que é uma visão bem mais prática.

Na figura 1 é possível observar um exemplo de prova em Dedução Natural considerando a Lógica Proposicional. A exibição está no formato conhecido como árvore de derivação e logo ao lado, em uma caixa, está a base de hipóteses e premissas utilizadas nessa prova. As hipóteses são diferenciadas das premissas por estarem entre colchetes, serem numeradas de acordo com as regras de inferência que as geraram e também por terem que ser descartadas no final da prova, o que pode ser visto como sendo um traço sobre as respectivas fórmulas.

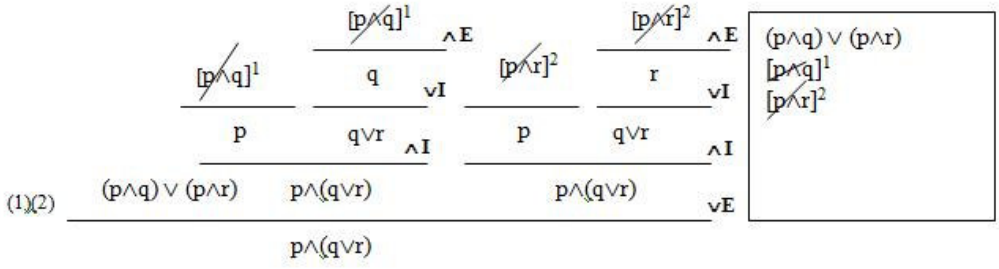

Figura 1 - Exemplo de Prova em Dedução Natural

Além do conceito de prova, existem ainda os conceitos de conseqüência lógica e teorema:

- Dados uma fórmula α e um conjunto de premissas Γ, então α é conseqüência lógica de Γ (ou Γ implica logicamente α) se houver uma prova de α a partir de fórmulas de Γ. Representa-se: $\Gamma \vdash \alpha$.
- Uma fórmula α é considerada um teorema no sistema de Dedução Natural caso exista uma prova de α a partir de um conjunto de premissas que seja vazio.

Prova-se que o sistema de Dedução Natural apresentado é correto e completo, isto é, tem-se a certeza de que uma fórmula é uma tautologia se, e somente se, existe uma prova para ela nesse sistema. Seguem as definições de correção e completude (fracas) respectivamente:

- Seja α uma fórmula da Lógica Proposicional, se α for um teorema no sistema de Dedução Natural, então α é uma tautologia.
- Seja α uma fórmula da Lógica Proposicional. Se α é uma tautologia, então α é um teorema no sistema de Dedução Natural.

Além disso, prova-se também resultados chamados de corretude e completude fortes que relacionam as noções de conseqüência lógica, do ponto de vista semântico e do sistema de dedução considerado. Sejam α uma fórmula e Γ um conjunto de fórmulas da Lógica Proposicional:

- se α for conseqüência lógica de Γ no sistema de Dedução Natural, então Γ implica α (se . $\Gamma \vdash \alpha$, então $\Gamma \models \alpha$).
- se Γ implica α, então α é conseqüência lógica de Γ no sistema de Dedução Natural, então (se . $\Gamma \models \alpha$, então $\Gamma \vdash \alpha$).

Sabendo como funciona a construção de uma prova em lógica proposicional pelo método de dedução natural, passa-se então a abordar aspectos sobre a interação do usuário durante a construção de uma prova. As próximas seções abordam as fases durante a construção de uma prova e quais são as atividades realizadas pelo usuário durante esse processo.

2.3 FASES NA PROVA INTERATIVA DE CONJECTURAS

Segundo (Gray & Aitken), algumas fases podem ser definidas como as mais importantes durante um processo de construção interativa de uma prova. São elas:

- Formalização: Refere-se à definição de constantes e novos tipos lógicos;

- Prova: Estabelecer um objetivo e alcançá-lo através da aplicação de regras de inferência ou regras derivadas;
- Manuseamento da Prova: Inclui todas as atividades envolvidas em estabelecer os parâmetros iniciais do provador como carregar algum módulo que contenha conjecturas já provadas ou axiomas.

Obviamente a maior parte da interação entre o usuário e o provador dá-se na fase de prova, quando o usuário deve estabelecer o objetivo atual e prová-lo através das regras, premissas e hipóteses, além de outras provas disponíveis. Um resultado disso pode ser visto em (Gray & Aitken).

2.4 ATIVIDADES DO USUÁRIO DURANTE A CONSTRUÇÃO DE UMA PROVA

Vários provadores e assistentes de construção de provas já foram desenvolvidos para uma grande variedade de lógicas, sistemas de dedução e estilos de raciocínio. Com isso foi possível notar que um grande problema reside no projeto da interface do programa. Uma interface pobre pode obstruir a interação entre o usuário e o programa computacional, e conseqüentemente reduzir a eficiência com que o usuário poderia obter uma prova.

Mesmo com a grande variedade de estilos de interface já implementados, os princípios para tais projetos são raramente destacados e, quando são, as justificativas para tais implementações não são suficientemente esclarecedoras (Aitken, Gray, Melham, & Thomas, Interactive Theorem Proving: An Empirical Study of User Activity, 1998). Uma possível maneira de obter justificativas seria estudando e identificando critérios para avaliação de interfaces de provadores, como é discutido por (Merriam & Harrison, Evaluating the Interfaces of Three Theorem Proving Assistants, 1996), que apresentam algumas funções a serem suportadas pela interface.

Os aspectos relacionados a interface serão abordados no próximo capítulo. A proposta dessa seção é realizar uma fundamentação sobre a interação do usuário no momento da construção da prova.

A noção mais primitiva de uma prova formal é o método de busca de prova direcionada ao objetivo, em que regras de inferência são aplicadas em seqüência sobre axiomas e teoremas previamente provados até se alcançar o objetivo desejado.

Existem dois modelos a serem adotados: *forward proof* (análise) ou *backward proof* (síntese). O último, mais "natural" e promissor, é o modelo geralmente amparado pelas interfaces conhecidas, dentre elas o HOL.

Independentemente do modelo de interação a ser adotado pelo usuário, sua ação em um determinado momento da prova pode ser caracterizada pela seguinte seqüência:

- Selecionar a fórmula ou objeto a provar
- Selecionar a tática ou regra a ser aplicada

Uma interface pode auxiliar ou não o usuário a seguir essa maneira de refletir. Por exemplo, uma interface que utilize menus e manipulação gráfica obriga o usuário a primeiro selecionar uma fórmula ou objeto e em seguida a regra a ser aplicada, enquanto interfaces orientadas a linhas de comando obrigam o usuário a digitar primeiramente a regra a ser utilizada.

Considerando que a maneira natural e intuitiva do usuário interagir durante a construção de uma prova é: primeiro identificar a fórmula ou o objeto a provar e depois selecionar a regra a ser aplicada, percebe-se a vantagem de uma interface baseada em menus e manipulação gráfica.

3 CONCEITOS DE INTERFACE HOMEM-COMPUTADOR

Neste capítulo são apresentados alguns conceitos de Interface Homem-Computador (IHC) e também alguns estudos específicos sobre interfaces para provadores automáticos e assistentes para construção de provas. Esses conceitos são utilizados na especificação da interface para o assistente para construção de provas para a Lógica Proposicional em Dedução Natural.

3.1 PROJETANDO INTERFACES PARA PROVADORES

De acordo com (Aitken, Gray, Melham, & Thomas, Interactive Theorem Proving: An Empirical Study of User Activity, 1998) três níveis de abstração são suficientes para caracterizar a interação entre o usuário e o sistema em assistentes e provadores. São eles:

- Nível lógico: consiste na descrição em termos de conceitos lógicos;
- Nível de Interação Abstrata: tipicamente são os objetos visíveis e as operações que podem ser aplicadas sobre eles, sem considerar os detalhes da forma física de como isso é realizado. Exemplos são diagramas, textos estruturados e listas.
- Nível de Interação Concreta: neste nível estão as ações nos mecanismos de entrada e as características perceptuais da apresentação dos objetos. Consiste na interação do usuário com a máquina de maneira física, o que o usuário utiliza para entrar com os dados do problema e como ele visualiza a saída.

Grande parte do tempo de desenvolvimento atual de interfaces e pesquisas relacionadas a esse problema tem sido dedicada ao nível da interação concreta e seu relacionamento com o nível de interação abstrata. São usualmente denominados estilos ou técnicas de interação e incluem os objetos de interação tais como estruturas de árvore ou *pop-up* menus, assim como técnicas de *drag-and-drop*.

Um novo aspecto que também deve ser levado em consideração ao se projetar a interface de um assistente para construção de provas é qual a visão de interação mais adequada. Algumas visões com relações as interfaces já existentes foram analisadas por

(Aitken, Gray, Melham, & Thomas, Interactive Theorem Proving: An Empirical Study of User Activity, 1998). São elas:

- Prova como programação – usualmente relacionada às interfaces com linha de comando. Ao final da prova o usuário tem um script com a seqüência de aplicação de regras;
- Prova por manipulação – Foi proposto como maneira de sintetizar os comandos fornecidos por um provador. Para lógicas mais simples a substituição de comandos pelo uso do mouse pode ser feita sem problemas;
- Prova como edição de estruturas – Consiste na construção e edição de objetos de prova. Consiste na manipulação direta dos objetos (associação de fórmulas e regras);

3.2 TEORIA DE GESTALT PARA PROJETO DE INTERFACES

A Teoria de Gestalt é uma família de teorias psicológicas que influenciou várias áreas desde 1924, incluindo problemas que envolvem visualização. Tais teorias são usualmente expressadas por leis que, especificamente para o projeto de interfaces, sugerem como elementos visuais estáticos devem ser apresentados a fim de se obter resultados visuais efetivos.

Foi notado que apenas uma pequena parte da Teoria de Gestalt pode ser aplicada ao projeto visual de interfaces. No trabalho de (Chang, Laurence, & Tuovinen, 2001), foram identificadas onze leis que representam os aspectos mais importantes da Teoria de Gestalt sobre a forma visual. Algumas dessas leis serão abordadas a seguir:

3.2.1 Lei da Simetria e Equilíbrio

Um elemento visual irá parecer incompleto se este não estiver equilibrado ou simétrico. Um senso psicológico de equilíbrio pode ser obtido quando o "peso" visual é distribuído de maneira igual em cada lado de uma imagem dividida por um eixo. Nas figuras 1 e 2 a seguir tem-se um exemplo de objetos posicionados de maneira simétrica e assimétrica.

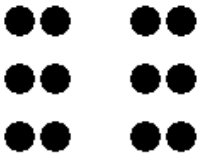

Figura 2 – Simetria

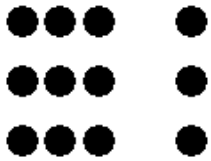

Figura 3 – Assimetria

3.2.2 Lei da Proximidade

Elementos posicionados próximos uns aos outros tendem a ser vistos como um grupo. Observadores podem organizar mentalmente objetos mais próximos em um único elemento coerente pelo fato de que objetos espaçados por uma pequena distância estão relacionados, ao contrário dos que estão espaçados por uma grande distância. Na figura 3 abaixo é possível identificar que os elementos estão agrupados em linhas ao invés de colunas.

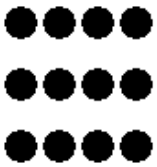

Figura 4 - Três linhas

3.2.3 Lei do Ponto Focal

Toda apresentação visual necessita de um ponto focal, denominado centro de interesse ou ponto de ênfase. Esse ponto focal captura a atenção do observador e o faz notar primeiro a mensagem visual em destaque. Nas figuras 4 e 5 nota-se claramente a diferença do foco em uma mesma imagem. Enquanto a figura 4 tem o foco dado ao vaso, a figura 5 apresenta duas faces como ponto de ênfase.

Figura 5 – Vaso

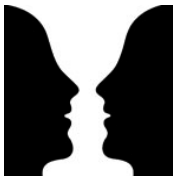

Figura 6 - Duas faces

3.2.4 Lei da Simplicidade

Quando está se tentando transmitir algum conhecimento, geralmente há um esforço para simplificar o que será apresentado, de maneira que o observador possa compreender facilmente. Essa lei pode ser aplicada visando deixar a interface de um software *clean*, termo utilizado atualmente para fazer referência a interfaces que valorizam a idéia da simplificação da interface. Por exemplo, é muito mais simples e eficiente utilizar a figura 6 para ensinar a forma da constelação do Cruzeiro do Sul do que a figura 7 que contém vários outros elementos.

Figura 7 - Cruzeiro do Sul

Figura 8 - Cruzeiro do Sul e outros elementos

3.3 TIPOS DE INTERFACE E PARÂMETROS DE ANÁLISE

Segundo (Merriam & Harrison, Evaluating the Interfaces of Three Theorem Proving Assistants, 1996) existem três tipos básicos de interface: linha de comando, menus e manipulação gráfica direta. Esses três tipos básicos de interface podem ser diretamente relacionados às três visões propostas por (Aitken, Gray, Melham, & Thomas, Interactive Theorem Proving: An Empirical Study of User Activity, 1998), citadas anteriormente na seção 3.1.

A maioria dos provadores utiliza a facilidade da linha de comando que é simples e de fácil implementação, porém apresentam a desvantagem de que o usuário deverá concentrar uma parte do seu tempo na aprendizagem da sintaxe e dos comandos do programa, o que pode se tornar um grave problema caso o provador possua um vasto dicionário de comandos. Já as interfaces baseadas em menus ou manipulação gráfica direta são consideradas mais amigáveis, já que não requerem muito esforço para aprender a manipular o programa.

Nas seções a seguir serão apresentados alguns parâmetros de análise de interfaces de provadores e assistentes para construção de provas. Tais parâmetros são propostos por (Merriam & Harrison, Evaluating the Interfaces of Three Theorem Proving Assistants, 1996). A idéia proposta é identificar as necessidades do usuário durante o processo de construção de uma prova e a partir daí verificar o que a interface de um provador ou assistente pode prover para facilitar as atividades exercidas pelo usuário. Posteriormente tais parâmetros serão utilizados para uma avaliação de algumas interfaces de provadores e assistentes atuais.

3.3.1 Planejamento

Esse critério avalia o que o programa computacional apresenta como auxílio ao planejamento da construção da prova, algum módulo que viabilize ao usuário abstrair e visualizar a seqüência de regras a serem aplicadas a fim de se obter a solução do problema. A maneira como a prova é exibida auxilia na visualização do problema.

3.3.2 Reusabilidade

O programa também deve apresentar algum módulo que viabilize a reutilização de provas já realizadas, o que facilitaria a prova de outros teoremas maiores e mais complexos já que o usuário poderia abstrair-se dos problemas já solucionados e concentrar-se nas questões específicas relativas à solução do problema atual.

A viabilização de procedimentos para criação de lemas também é uma forma de suprir essa necessidade do usuário. A idéia da criação de lemas em um provador ou assistente está geralmente associada à questão de postergar a realização de alguma pequena parte da prova e continuar focando no problema principal.

3.3.3 Reflexão

A reflexão é um critério interessante, pois é o primeiro passo para um bom planejamento da construção da prova. O entendimento do objetivo e do problema faz parte desse critério. Um possível módulo de reflexão poderia ser a viabilização de janelas de rascunho, onde o usuário poderia realizar e testar pequenas provas que

posteriormente poderiam ser inseridas no problema inicial, tendo em consideração a correta aplicação de regras e utilização de hipóteses e premissas no escopo do problema principal.

3.3.4 Articulação

A articulação consiste na associação do objetivo com as hipóteses e suposições do problema. O programa pode viabilizar ao usuário uma melhor maneira de articular as fórmulas do problema através da sugestão de regras a serem aplicadas, considerando determinadas premissas e conclusões.

4 ANÁLISE DO ESTADO DA ARTE

Foram analisados alguns softwares que apresentam uma interface gráfica razoável e que visa uma melhoria na interação usuário com o provador ou assistente, além da simples interface por linha de comando existente na maioria das soluções atuais. Dentre os softwares testados estão: o módulo de Dedução Natural do JAPE, COQ, ProverBox e SPASS.

4.1 COMPARAÇÕES DE PROVADORES E ASSISTENTES

A seguir é apresentada uma tabela com uma comparação dos programas computacionais escolhidos. Uma análise mais detalhada foi feita para os dois provadores que utilizam um método dedutivo mais próximo do que será utilizado para proposta de uma nova interface: o módulo de dedução natural do JAPE, que utiliza o método de dedução natural, e o COQ, que utiliza um método conhecido como cálculo de seqüentes.

Provadores	**JAPE**	**COQ**	**ProverBox**	**SPASS**
Sistema	Assistente	Assistente + Provador Automático	Provador Automático	Provador Automático
Interface	Menu + Manipulação Gráfica	Linha de Comando + Menu	Linha de Comando + Menu	Linha de Comando + Menu
Método Analisado	Dedução Natural	Cálculo de Seqüentes	Resolução	Resolução
Visualização da Prova	Seqüencial em forma de caixas	Seqüencial baseado em cálculo de seqüentes	Seqüencial	Seqüencial
Reutilização	Sim	Sim	Não	Não
Reflexão	Criação de lemas	-	-	-
Articulação	Divisão do conjunto de regras de inferência	Destaque do objetivo atual do problema	-	-

Tabela 2 - Comparação de Provadores e Assistentes

4.1.1 Análise do Módulo de Dedução Natural do JAPE

O JAPE (Just Another Proof Editor) possui um módulo assistente de construção de provas em dedução natural que apresenta algumas facilidades, embora ainda necessite de muitos recursos para uma boa assistência. Esse módulo também não possui a facilidade de provar um teorema automaticamente.

Figura 9 - JAPE - Just Another Proof Editor

Sua interface é baseada em menus e em manipulação gráfica, o usuário escolhe a função através de menus e faz a associação de premissas e conclusões a serem utilizadas na aplicação de uma regra através da manipulação gráfica.

A visualização da prova é feita seqüencialmente através de caixas o que melhora a visualização com relação à apresentação puramente seqüencial, porém o resultado ainda fica muito distante da representação da prova através da árvore de derivação.

Ao se abrir este módulo, várias janelas com algumas conjecturas são apresentadas, conjecturas simples e clássicas que o usuário pode provar para reutilizar posteriormente. Portanto a reutilização de provas é viável nesse assistente. É possível

verificar nas listas apresentadas nessas janelas as conjecturas e teoremas que foram provados ou os que ainda necessitam de solução. O usuário pode modificar a lista, entrando com novos teoremas a serem provados.

Esse módulo também contribui para a reflexão. Quando o usuário está realizando alguma prova, ele pode selecionar premissas e suposições e uma conclusão através da manipulação gráfica e em seguida utilizar o menu para a abertura de uma janela que permite a criação de um lema, que na verdade é uma pequena parte da prova. O usuário então pode abstrair-se do problema principal, provar o lema e em seguida retornar com a solução para o problema principal. Também está disponível para o usuário uma opção de criar regras derivadas, que na verdade é um tipo de seqüenciamento de instruções também conhecido como macros ou *tactics*.

Quanto à articulação, o JAPE divide as regras de inferência de dedução natural em dois grupos, separando as regras de introdução das regras de eliminação. Isso facilita o usuário manter foco no método de prova utilizado, seja ele o de síntese ou análise.

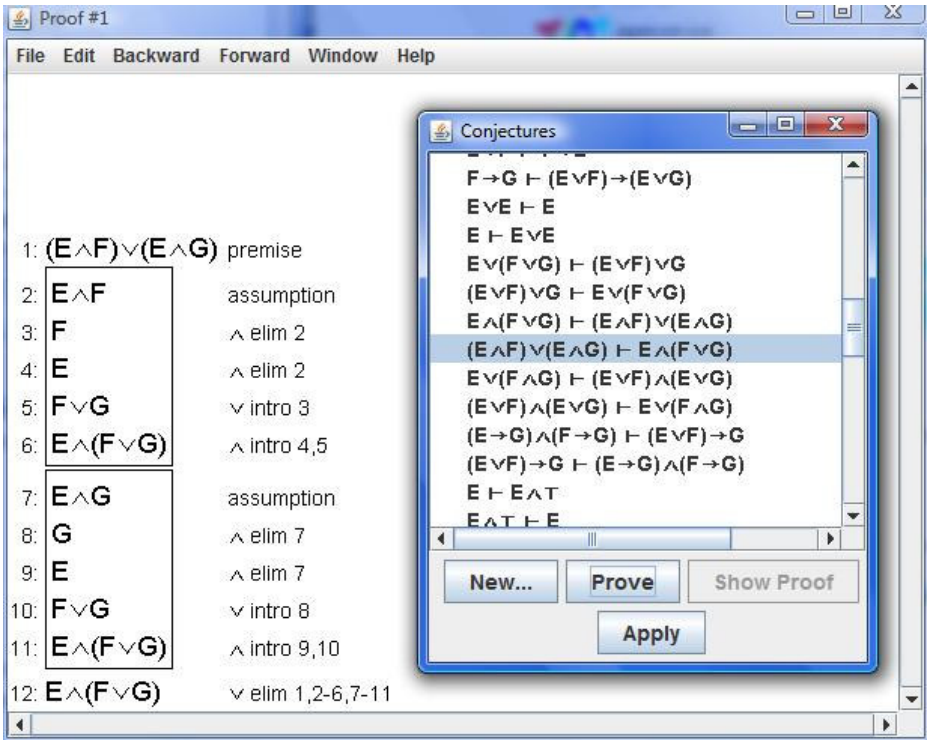

Figura 10 - Exemplo de Prova no JAPE

Na figura 10 é apresentado um exemplo de prova em Dedução Natural no JAPE. Foi cadastrada a mesma prova apresentada na figura 1 no capítulo 1. Nota-se claramente

a diferença da representação do formato de árvore de Dedução Natural para o formato linear apresentado pelo JAPE. No caso específico do JAPE, ainda há o auxílio das caixas representando os diferentes ramos da prova. Mesmo assim a visualização é complicada se comparada ao formato de árvore de derivação.

4.1.2 Análise do Provador Coq

O provador Coq foi testado utilizando-se o módulo de cálculo de seqüentes. Esse assistente consiste basicamente de uma interface de linha de comando, embora exista uma versão da interface com menus CoqIDE mas que somente se diferencia da anterior por apresentar a vasta quantidade de instruções no formato de menu.

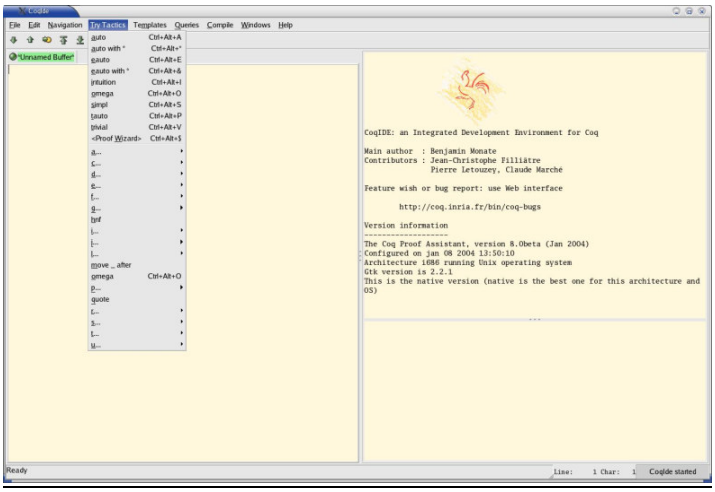

Figura 11 - CoqIDE

A visualização é baseada na organização do cálculo de seqüentes, passo a passo, o que dificulta o planejamento e o entendimento do processo de construção da prova. Em compensação o assistente apresenta facilidades como a possibilidade de destacar o objetivo atual ou a conclusão que deve ser provada em determinado momento.

A reutilização de provas também é possível, mas como o programa apresenta uma interface por linha de comando, esse módulo fica prejudicado. Um exemplo disso é a criação de lemas que deve ser feita antes da realização da prova que o utiliza.

Uma propriedade que poderia ser explorada por esse provador seria a viabilização de janelas para manipulação de pequenas provas e lemas, o que permite a construção do lema mesmo durante a construção da prova.

5 PROPOSTA DE UM ASSISTENTE PARA CONSTRUÇÃO DE PROVAS

O desenvolvimento do Assistente para Construção de Provas de Lógica Proposicional em Dedução Natural teve início em um projeto de Iniciação Científica. Mais adiante se decidiu adotar um processo desenvolvimento de software considerando que este trabalho levaria um tempo maior de desenvolvimento e podendo envolver vários alunos.

Considerando essa situação, foi realizado um Estudo de Viabilidade (Capítulo 8, Seção 8.1) a partir do qual optou-se por um processo de desenvolvimento incremental. Neste Projeto de Final de Curso foi elaborado o primeiro incremento que tem como objetivo principal apresentar a interface homem-computador proposta para o assistente.

Fica, portanto, estabelecido que o foco da implementação neste trabalho não é o assistente completo, atendendo a todos os requisitos elicitados durante o processo de engenharia de software, mas sim uma primeira interface que atenda necessidades básicas do usuário e que motive a continuação do processo iterativo de desenvolvimento do assistente.

O primeiro passo para a elaboração do assistente consiste no levantamento, análise e classificação de requisitos. Com os principais requisitos bem definidos, inicia-se a fase de análise, que consiste na descrição detalhada do software para melhor compreensão das suas funcionalidades. Em seguida a etapa de projeto tem início, quando uma solução é apresentada com base principalmente em diagramas. Após todo esse processo o assistente é implementado na etapa de codificação e passa por uma validação na etapa de transição.

Durante essas fases, atividades relacionadas ao projeto e análise de interfaces estarão sendo desenvolvidas com base em conceitos de IHC e estudos realizados especificamente sobre provadores e assistentes para construção de provas. Um maior detalhamento das atividades a serem realizadas será feito na seção 5.2 que trata da descrição do processo de desenvolvimento.

5.1 IDÉIAS INICIAIS

A idéia do projeto é desenvolver um programa computacional que ofereça uma assistência adequada durante a construção de uma prova e também permita a prova automática de uma conjectura da Lógica Proposicional, em Dedução Natural.

A prova de uma conjectura em Dedução Natural, adotando-se o formato de árvore, geralmente inicia-se com o posicionamento do objetivo principal na parte inferior do ambiente de edição de prova. Em seguida são aplicadas várias regras em um processo que é denominado síntese, de tal maneira que a fórmula que representa o objetivo é "quebrada" até se alcançar uma fórmula atômica. Visualmente, a árvore aumenta, de baixo para cima.

A partir daí, inicia-se o processo de análise, que também consiste na aplicação de regras, tentando-se articular objetivos e conclusões intermediárias com hipóteses e premissas do problema. Visualmente a árvore cresce de cima para baixo, até unir as duas partes. Informalmente é dessa maneira que funciona a construção de uma prova em Dedução Natural.

Embora essa descrição relate o processo de construção de uma prova utilizando-se o formato de árvore, é possível realizar essa prova também no formato linear (por passos). A idéia é a mesma de síntese e análise.

Sabendo disso, o desafio é permitir que o usuário consiga realizar todo esse processo de construção de uma prova tendo a liberdade que ele teria de fazer o mesmo processo em uma folha de papel e, além disso, tendo a assistência e a facilidade que um programa computacional pode oferecer.

5.2 O PROCESSO DE DESENVOLVIMENTO

O modelo de processo de desenvolvimento visado para essa proposta é uma adaptação do Rational Unified Process (RUP). Tal modelo foi escolhido como base devido aos seguintes fatos:

- A utilização de um padrão de documentação, o que favorece a facilidade para o desenvolvimento de novas versões do programa por outras equipes de desenvolvimento.

- O foco em uma arquitetura baseada em componentes, o que facilita a expansão do programa computacional.
- O foco no desenvolvimento de modelos visuais e diagramas, tornando possível uma abstração da solução proposta desde o início do projeto.
- O modelo é baseado em um processo iterativo, de maneira que em cada etapa do processo, uma versão do programa é lançada com alguma funcionalidade nova. Desse modo o cliente pode opinar e guiar o desenvolvimento da maneira desejada.

Embora esse modelo de processo seja indicado para grandes equipes de desenvolvimento e empresas, ele pode ser facilmente adaptado para projetos pequenos, como é o caso.

5.2.1 Iniciação e Planejamento

Essa etapa envolve o início das atividades do projeto. Primeiramente foi realizado um Estudo de Viabilidade, que consiste no levantamento de provadores e assistentes atualmente utilizados e no estudo do domínio do problema. O detalhamento das atividades realizadas durante esse estudo está no capítulo 8, na seção 8.1.

Com todas as informações levantadas pelo Estudo de Viabilidade, é feito uma proposta para implementação do novo Assistente para Construção de Provas. Faz-se necessário, então, um detalhamento de todo o cronograma e das atividades a serem realizadas.

Ao fim dessa etapa, a equipe do projeto terá definido o escopo preliminar, o cronograma e as estimativas de recursos, custo e tempo necessário para desenvolvimento do projeto.

5.2.2 Concepção

Nessa etapa do processo de desenvolvimento é realizada a atividade de Modelagem de Negócios, cujo detalhamento está no capítulo 8, na seção 8.2. Essa atividade consiste especificamente na fundamentação do processo de construção de uma prova para lógica proposicional em dedução natural.

Em seguida é realizado juntamente com o cliente um levantamento de requisitos. Essa atividade pode ser subdividida em duas partes: levantamento dos requisitos do assistente e levantamento dos requisitos de interface. Os requisitos do assistente são mais gerais e consistem em como o cliente deseja que o programa computacional funcione. Esses requisitos estão fortemente relacionados com as regras de negócio e o domínio do problema. Já os requisitos de interface são obtidos com base em conceitos de Interface Homem-Computador (IHC).

Foi elaborado um questionário para se identificar alguns requisitos do assistente além de confirmar os já listados pela equipe do projeto. A lista final de requisitos é utilizada para se fazer uma revisão do escopo do projeto. Deve-se estar ciente de que os requisitos de interface ainda não foram coletados. A lista de requisitos, sua classificação e o questionário utilizado estão descritos no capítulo 8, na seção 8.3.

Os requisitos que foram abordados no protótipo implementado estão listados abaixo:

[R001] – Definir objetivamente a linguagem utilizada pelo sistema (símbolos proposicionais, conectivos proposicionais, símbolos de pontuação e símbolos de verdade);

[R002] – Detectar erros de sintaxe correspondentes à lógica proposicional;

[R003] – Possibilitar a aplicação de uma regra de inferência do método de Dedução Natural, dado que premissas ou uma conclusão foram selecionadas;

[R008] – A interface com o usuário será baseada em menus e manipulação gráfica;

[R010] – Possibilitar o uso de janelas de rascunho para manusear pequenas provas e auxiliar na reflexão do problema;

[R015] – Visualizar a prova no formato de árvore de Dedução Natural;

[R018] – Manter em um arquivo as listas de conjecturas, teoremas e suas respectivas provas, com a finalidade de recuperar tais informações após o sistema ser fechado;

5.2.3 Elaboração

Nessa etapa é realizada a modelagem de casos de uso. Isso é feito utilizando-se o documento de Modelagem de Negócios e o Documento de Requisitos. O diagrama de casos de uso tem a principal finalidade de documentar as funcionalidades do programa computacional e seus relacionamentos com o ambiente externo composto pelos

usuários. A descrição de cada caso de uso e os seus respectivos fluxos de execução está no capítulo 8, na seção 8.5.

Também se deve definir nessa etapa, uma arquitetura geral para o software. A idéia é documentar como o assistente funciona em termos de componentes ou módulos. Como a proposta inicial para esse programa computacional é bastante simples, foi feito apenas uma modularização baseado no diagrama de classes. Segue o modelo de tal arquitetura. Há uma descrição detalhada desse diagrama no capítulo 8, seção 8.5.

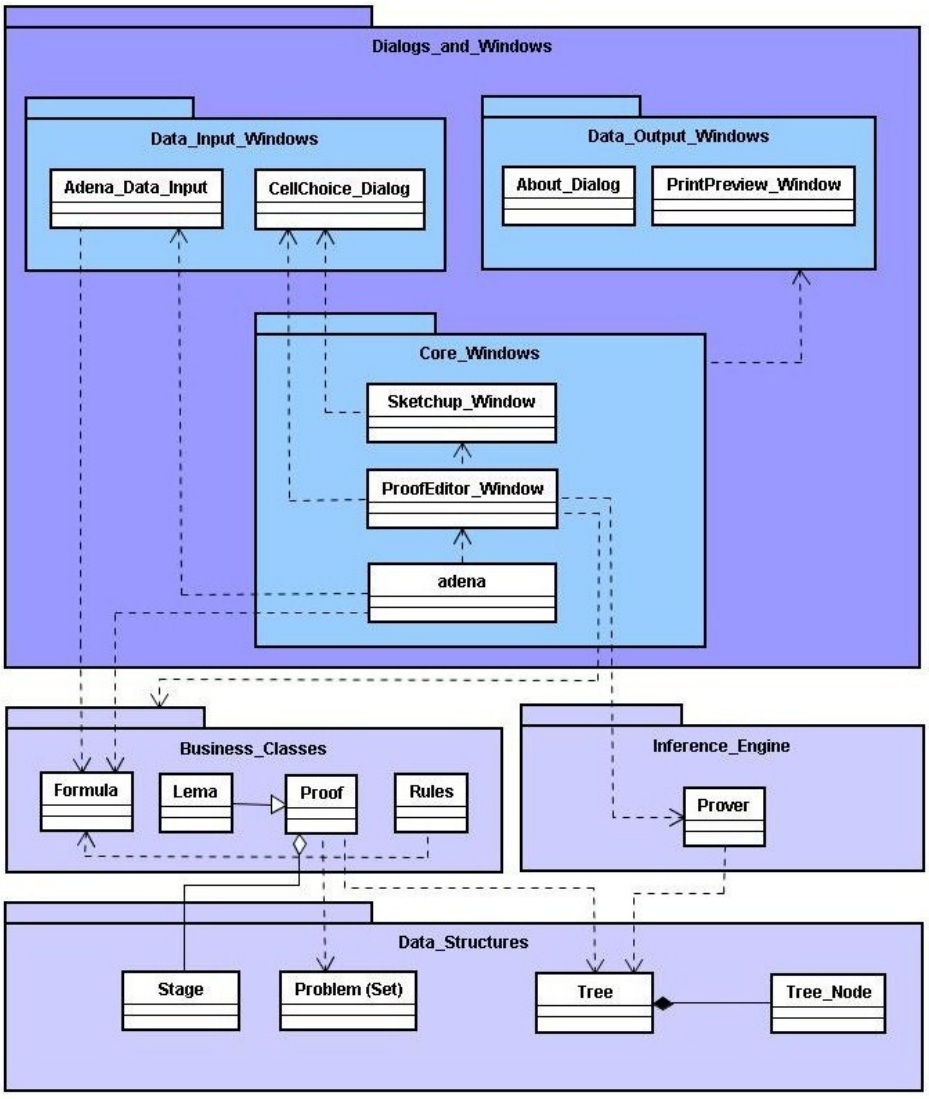

Figura 12 - Arquitetura do Assistente para Construção de Provas

Com relação a interface do assistente, é feito o *Storyboard*. Esse documento contém a especificação das telas do software e uma descrição das respectivas funcionalidades. Um detalhamento da interface é feito na seção 5.3 desse mesmo capítulo, considerando os conceitos de objetos de interação e Gestalt, além de uma descrição das janelas seguindo um modelo de *Storyboard*.

Nessa fase também é realizado o Plano de Garantia da Qualidade do projeto. Esse plano tem como principal objetivo formalizar e documentar as atividades e metodologias que visam garantir a qualidade do produto final. Com tais atividades, é possível também ter uma idéia melhor se o desenvolvimento está direcionado para o produto que o cliente deseja ou não.

5.2.4 Construção

A etapa de construção é uma das etapas que demanda maior tempo da equipe do projeto. Primeiramente é feito uma revisão dos diagramas criados na etapa de Elaboração e então uma arquitetura final é proposta como solução para o software. O maior detalhamento do diagrama de classes também permite ter uma noção mais concreta de como será implementado o assistente.

Tendo passado por todo o processo de planejamento da solução, iniciam-se as iterações de desenvolvimento do assistente. A idéia é que a cada iteração, uma nova funcionalidade ou requisito seja atendido.

Para manter o controle do projeto e garantir a qualidade do produto final, é essencial que seja alocado um tempo para atividades de Gestão de Mudanças. Qualquer revisão ou alteração solicitada pelo usuário deve ser primeiramente avaliada pela equipe do projeto para que todos os impactos sejam levados em consideração.

Além disso, a cada nova versão do programa, ou seja, a cada nova iteração, serão realizados testes unitários para verificar a integridade e a correção de cada módulo e função implementada. Os testes poderão ser feitos tanto pela equipe do projeto quanto pelos usuários finais do programa.

5.2.5 Transição

Finalmente, a versão final do assistente será entregue na etapa de Transição. Propõe-se uma apresentação da solução para os usuários finais e a disponibilização da ferramenta para *download* no site do projeto.

Com a divulgação e distribuição do produto, é possível realizar uma pesquisa sobre a receptividade dos usuários e assim propor melhorias e evoluções. Além disso, como o número de usuários tende a aumentar após a primeira divulgação, é mais provável que erros e bugs sejam descobertos com mais freqüência. Para isso deve-se manter uma equipe para realizar essas correções e fazer a manutenção do programa.

5.3 INTERFACE: GESTALT, OBJETOS DE INTERAÇÃO E STORYBOARD

Nesse documento está a especificação da interface do Assistente para Dedução Natural, ADENA. As janelas serão apresentadas e para cada uma delas haverá uma justificativa para a escolha dos objetos de interação selecionados (*combobox*, *labels*, *listviews*, *treeviews*, etc.). A disposição de tais objetos em cada janela também foi elaborada com base em algumas leis de Gestalt para especificação de interfaces de softwares. Seguem as telas do assistente:

5.3.1 A Janela Principal

Essa é a janela principal do assistente, a primeira janela que o usuário verá:

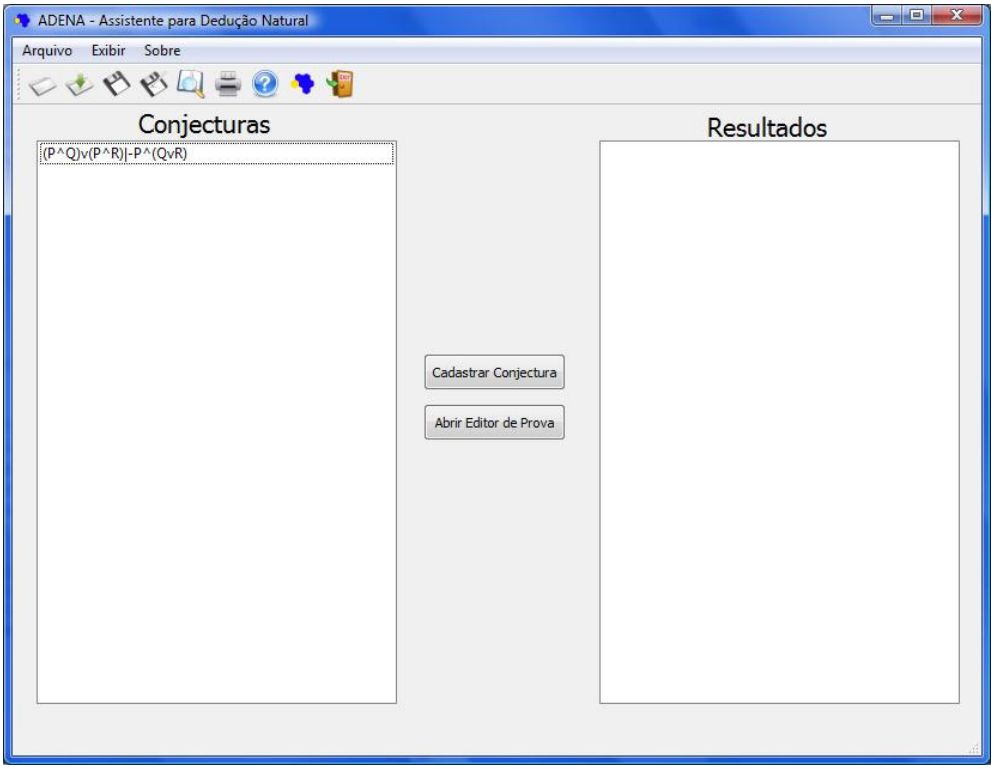

Figura 13 - Janela Principal

Ela contém dois elementos padrões dos softwares atuais: o menu no topo da janela e a barra de ferramentas com ícones de atalhos. O menu segue o padrão já consagrado pelos softwares atuais com as seguintes opções de Arquivo, Exibir e Sobre.

A barra de ferramentas com os ícones de atalhos serve apenas para agilizar o acesso a determinadas funções. No momento todas as funções apresentadas no menu estão na barra de ferramentas, mas quando novas funcionalidades forem adicionadas, isso não será possível pela *affordance* que a barra de ferramentas oferece.

Além desses elementos, a janela principal possui duas *listviews* que representam o conjunto de conjecturas e fórmulas já provadas, ou seja, elas apresentam o conteúdo da base de dados do assistente. Cada *listview* possui um *label* que indica o conjunto que ela está apresentando. A *listview* da esquerda apresenta o conjunto das conjecturas, enquanto a *listview* da direita apresenta o conjunto dos resultados, ou seja, das conjecturas já provadas. Entre as duas *listviews* estão dois botões: “Cadastrar Conjectura” e “Abrir Editor de Prova”. O primeiro é utilizado para o cadastro de novas conjecturas na *listview* da esquerda, enquanto o segundo permite o usuário provar uma conjectura em uma janela de edição de prova. Caso ele seja bem sucedido, a conjectura á automaticamente transferida para a *listview* da direita.

A disposição das *listviews* e dos botões também estão de acordo com algumas leis de Gestalt para design de interfaces. É possível notar a simetria da disposição dos objetos de interação na janela, o que transmite uma imagem de equilíbrio e harmonia. Além disso, pode-se notar que o foco da janela está no local em que estão posicionados os botões que permitem ao usuário acionar as principais funcionalidades do assistente.

5.3.2 A Janela de Cadastro de Conjecturas

Ao clicar no botão "Cadastrar Conjectura" da janela principal, o usuário verá a seguinte tela para cadastro de conjecturas:

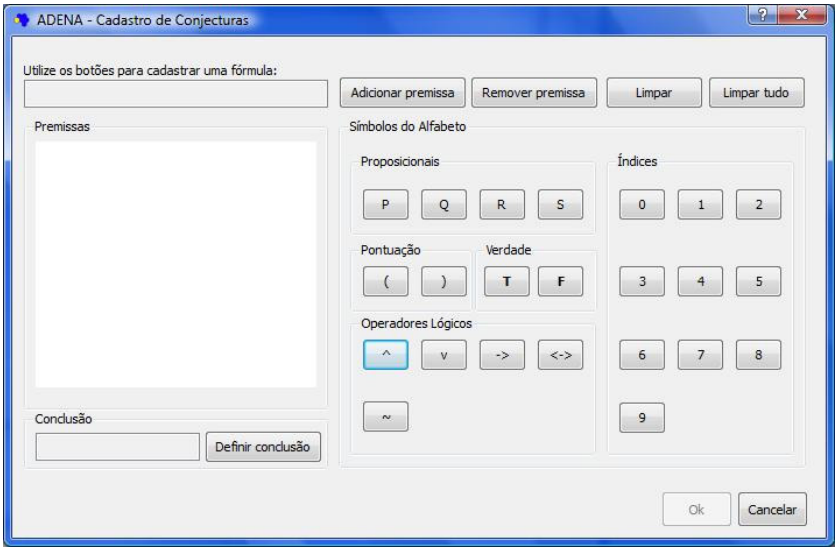

Figura 14 - Janela de Cadastro de Conjecturas

Nessa tela há um conjunto de botões no lado direito que funcionam quase como uma calculadora. Esses botões estão dispostos de acordo com alguns grupos, de acordo com a lei de Gestalt de proximidade. O usuário poderá clicar em qualquer um dos botões e o caracter que o botão representa aparecerá no *label* no canto superior esquerdo da tela (abaixo do *label* "Utilize os botões para cadastrar uma fórmula"). Considerando que esse objeto de interação será apenas para comunicar ao usuário qual a fórmula que está sendo digitada, ele foi escolhido ao invés do *line edit*.

Para cada fórmula digitada, o usuário terá a opção de registrá-la como uma premissa ou como uma conclusão. Caso o usuário queira registrá-la como premissa, basta clicar sobre o botão "Adicionar premissa" e a fórmula que estava sendo exibida no

label do canto superior direito será transferida para a *listview*. Essa *listview* tem como principal função apresentar todo o conjunto de premissas registradas para aquela conjectura. Caso o usuário deseje registrar a fórmula como uma conclusão, ele deverá clicar sobre o botão "Definir conclusão". Isso fará com que a fórmula seja transferida do *label* do canto superior direito para o *label* do canto inferior direito.

Também estão disponíveis os botões de "limpar" e "limpar tudo" para permitir a entrada de novas fórmulas ou a correção das mesmas.

Por fim, quando toda a conjectura já foi registrada pelo usuário, o cadastro pode ser concluído apertando-se o botão "Ok" no canto inferior esquerdo da tela. Caso o usuário deseje cancelar o cadastro da conjectura, ele pode a qualquer momento apertar o botão "Cancelar" ao lado do botão "Ok".

5.3.3 A Janela de Edição de Provas (Síntese)

Ao clicar no botão "Abrir Editor de Prova" da janela principal, o usuário verá a seguinte tela:

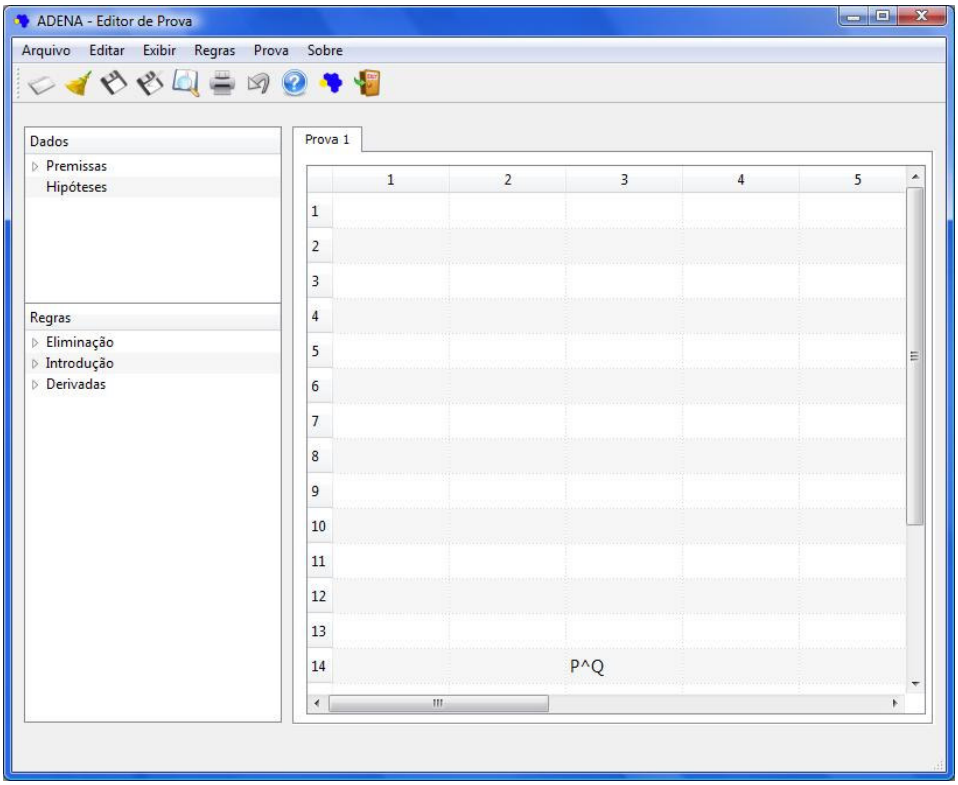

Figura 15 - Janela de Edição de Provas (Síntese)

Essa janela é o editor de provas principal. Nela o usuário pode construir uma prova aplicando as regras de Dedução Natural para Lógica Proposicional selecionando as proposições que aparecem no *grid* e na *treewidget* "Dados". A idéia é que o usuário utilize essa janela apenas para construir a prova no sentido de síntese. Embora também seja possível a aplicação de regras no sentido de análise, o ideal é que esse processo seja feito em outra janela que será apresentada posteriormente. Isso foi decidido pelo fato de que seria muito custoso controlar os dois processos (síntese e análise) no mesmo *grid*, sabendo-se que em um determinando momento as duas árvores devem se conectar.

Com relação aos objetos de interação escolhidos para essa janela, além do menu e da barra de ferramentas com os atalhos, existem três outros componentes: duas *treewidgets* e um *grid.*

A primeira *treewidget* armazena a base de dados de hipóteses e premissas da prova em construção. As premissas são constantes e não se alteram ao longo do processo de construção da prova. Já as hipóteses podem ser criadas e descartadas ao longo desse processo. Esse objeto de interação foi escolhido para transmitir a idéia de que premissas e hipóteses são conjuntos relacionados e para economizar o espaço da janela. Nesse caso a *treewidget* não requer muito espaço, já que, para esse programa, as provas não serão muito extensas e, portanto não terão muitas premissas ou hipóteses.

A segunda *treewidget* contém as regras de Dedução Natural para Lógica Proposicional. Essas regras são divididas em conjuntos para facilitar a localização de cada uma delas. Além disso, para cada regra, há uma imagem ao lado exemplificando a aplicação da mesma. Esse objeto de interação foi escolhido para viabilizar um processo de manipulação direta da prova no *grid*, via *drag & drop*.

E finalmente o *grid* foi escolhido como objeto de interação para o editor de prova pelo fato de ser mais fácil a identificação da posição de cada fórmula e o local onde ocorre a aplicação de cada regra. Com essa facilidade, é possível apresentar a prova no formato de árvore de Dedução Natural, o que até hoje ainda não foi bem explorado pelos provadores automáticos e assistentes para construção de provas.

A disposição desses objetos na tela foi determinada desse modo pelo fato de que muitas aplicações utilizam esse conceito de se ter um editor ocupando a parte central-direita da tela e um conjunto de ferramentas no canto esquerdo da tela. Na verdade toda a aplicação funciona como um painel, em que os comandos estão do lado esquerdo e o

foco está no objeto que ocupa maior espaço da tela. Além disso, pode-se dizer que com a escolha desses objetos de interação a tela ficou mais simples e limpa, obedecendo à regra de simplicidade da teoria de Gestalt.

5.3.4 A Janela de Edição de Prova (Análise -Sketchup)

Pela janela de edição de prova principal é possível acessar a janela de rascunho que é apresentada a seguir:

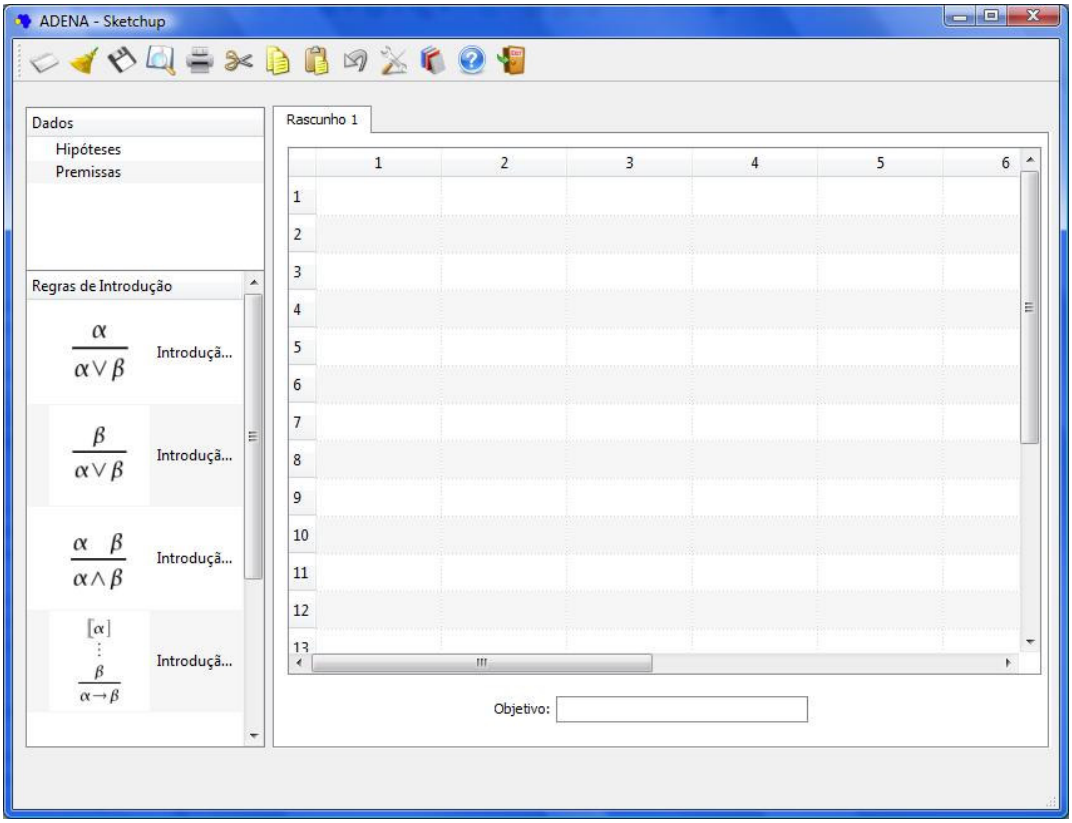

Figura 16 - Janela de Edição de Prova (Análise - Sketchup)

A idéia de se ter essa janela de rascunho é separar os processos de síntese e análise, como foi comentado anteriormente, e também permitir que o usuário faça esboços e várias tentativas sem se preocupar em "sujar" a prova principal.

Essa janela pode ser ativada pelo menu da janela de edição de prova principal especificando-se um objetivo da prova principal que se deseja provar, limitando também

o conjunto de hipóteses que se pode utilizar. Dessa maneira é possível fazer o controle do uso correto das hipóteses de acordo com o ramo da prova.

Essa janela é basicamente igual a janela de edição de prova principal com algumas pequenas diferenças. Isso foi feito dessa maneira para evitar que o usuário tenha que aprender um novo método de interação para a mesma função de construir uma prova. Portanto, as pequenas diferenças não interferem no método de interação para a construção de uma prova com a aplicação de regras via *drag & drop* ou menu. Na verdade a diferença principal entre essa janela e a janela de edição principal é o fato de o objetivo da prova estar em um *label* separado e não no *grid*. Isso significa que o usuário deve alcançar aquela fórmula do *label* construindo a prova num sentido de análise, ao contrário da janela de edição principal em que o objetivo está dentro do *grid*, permitindo o usuário construir a prova no sentido de síntese.

5.3.5 A Janela de Suporte à Aplicação de Regra (Análise)

Essa janela é utilizada para suporte à interação durante a construção de uma prova, caso o usuário deseje seguir o caminho de análise aplicando regras pelo menu ao invés de usar o método *drag & drop*. Nela o usuário deve entrar com a linha e a coluna em que se deseja posicionar o resultado da aplicação da regra selecionada no menu. Para isso foram escolhidos dois objetos do tipo *spinbox*, que nesse caso representam a posição da célula desejada no *grid*. Esse objeto é o ideal para esse caso, pois limita o usuário a entrar somente com números inteiros, ocupa pouco espaço o que é ideal para esse pequeno *dialog*. Como padrão de todo *dialog*, os botões “Ok” e “Cancelar” estão disponíveis.

A disposição dos elementos nesse *dialog* obedece a regra de simetria de Gestalt e também dão foco aos elementos *spinbox* no centro da janela. Além disso, a janela é bem pequena e com poucos elementos de interação. Essa disposição transmite uma idéia de completude e simplicidade.

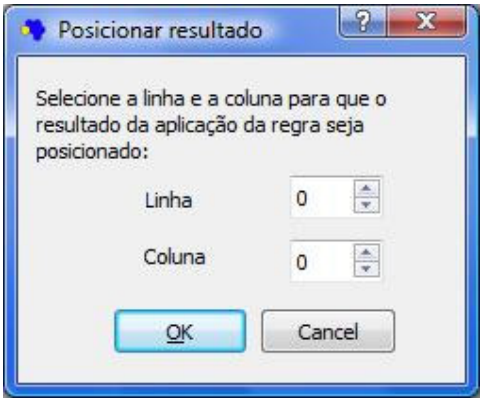

Figura 17 - Janela de Suporte a Aplicação de Regra (Análise)

5.3.6 A Janela de Informações sobre o Assistente

Essa é uma janela que serve apenas para comunicação com o usuário. Não há qualquer interação disponível. Essa tela contém somente alguns *labels* que passam algumas informações sobre o assistente e o projeto que o originou e um botão de "Ok" para se fechar a janela. Pelo fato dessa janela ser somente para comunicação foram escolhidos somente objetos do tipo *label*.

Figura 18 - Janela de Informações sobre o Assistente

5.3.7 Exemplo de Prova em Dedução Natural – ADENA

A seguir está uma figura com a mesma prova realizada no JAPE pelo modo de caixa e no modo tradicional de árvore de derivação apresentado na figura 1 do capítulo 1. Nota-se aqui que a representação visual adota um estilo de árvore de prova também. Como essa é uma primeira versão para o programa, muitas melhorias ainda podem ser realizadas de modo que a visualização da prova seja a mais fiel e próxima da realidade possível.

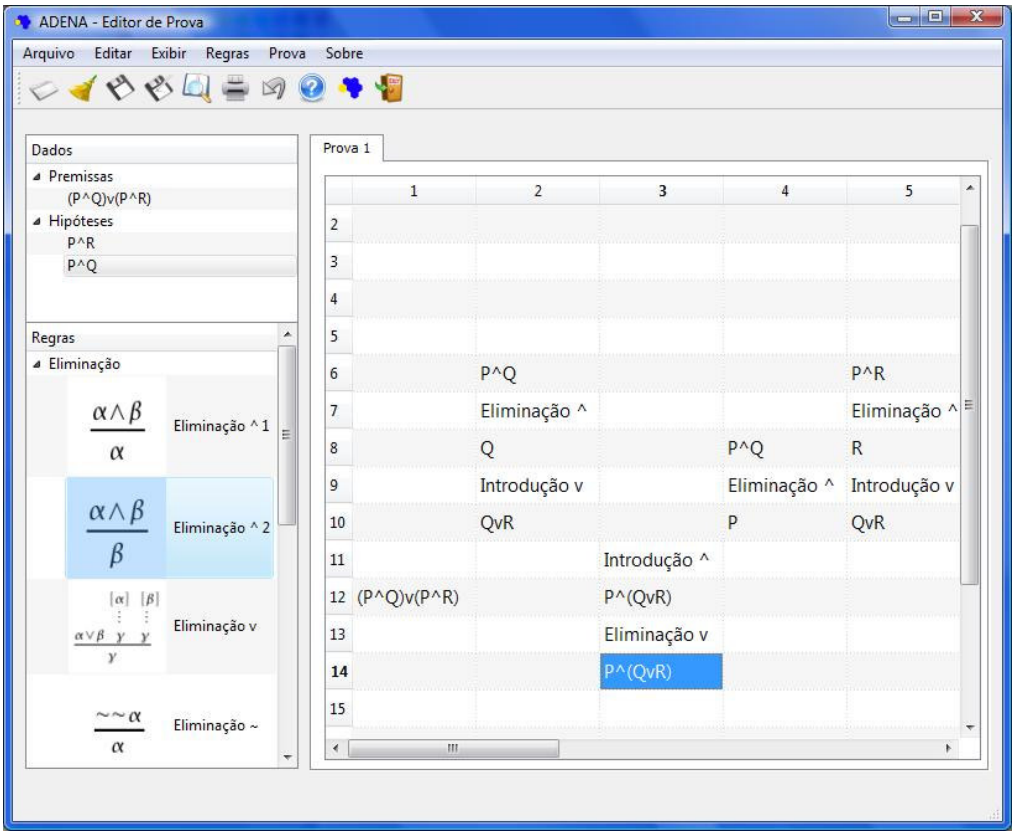

Figura 19 - Exemplo de Prova no ADENA

6. CONCLUSÃO E TRABALHOS FUTUROS

Com esse trabalho pretendeu-se abordar várias disciplinas de Ciência da Computação com a principal intenção de colocar em prática o que foi estudado ao longo do curso de graduação.

Embora o foco do projeto tenha sido a área de interfaces para um tipo específico de programa computacional, outros aspectos importantes foram a fundamentação sobre o método de Dedução Natural para Lógica Proposicional e o estudo do processo de engenharia de software que foi proposto.

Com a base fornecida pelas disciplinas e o estudo de artigos específicos da área de Interface Homem-Computador (IHC) para provadores automáticos e assistentes para construção de provas foi possível exercitar o senso crítico e propor novas idéias nessa área que adquiriu grande importância nos últimos anos.

Como resultado deste projeto, tem-se uma primeira versão do Assistente para Construção de Provas de Lógica Proposicional em Dedução Natural. Essa versão consiste apenas na interface com algumas funcionalidades básicas implementadas, tais como aplicação das regras de Dedução Natural e verificação da sintaxe de uma conjectura de acordo com o alfabeto da Lógica Proposicional.

Tendo em vista o processo de engenharia de software proposto, muito ainda deve ser feito para se ter uma versão que atenda a todos os requisitos elicitados. Mesmo assim, já é possível pensar em futuras expansões e implementações para o assistente.

Como um item interessante para trabalhos futuros, é possível expandir o domínio do problema para que seja possível a construção de provas para a Lógica de Predicados de 1ª Ordem. Para isso algumas alterações no alfabeto deverão ser realizadas, assim como a adaptação para as novas regras de inferência.

Outra expansão que pode ser realizada é a disponibilização do assistente pela *Web*. O usuário poderia ou utilizar o assistente para Dedução Natural pela internet ou fazer o *download* de uma versão para instalar localmente. O benefício dessa expansão é poder utilizar o assistente de qualquer lugar e qualquer computador.

Além disso, com certeza após a entrega da primeira versão completa do assistente e com o aumento da utilização por alunos e professores, que são os principais usuários, surgirão novas propostas de procedimentos de assistência e até mesmo funcionalidades até então não abordadas pelo documento de requisitos.

7 REFERÊNCIAS

Abreu, E. d., & Teixeira, J. C. (2007). *Apresentação de Trabalhos Monográficos de Conclusão de Curso.* Niterói: EdUFF.

Aitken, J. S. (1996). Problem Solving in Interactive Proof: A Knowledge-Modelling Approach. *European Conference on Artificial Intelligence* (p. 5). Budapest: John Wiley & Sons, Ltd.

Aitken, J. S., Gray, P., Melham, T., & Thomas, M. (1998). Interactive Theorem Proving: An Empirical Study of User Activity. *Journal of Symbolic Computation* , pp. 263-284.

Aitken, J. S., Gray, P., Melham, T., & Thomas, M. (1996). Phases, Modes and Information Flow in Theory Development. *User Interfaces for Theorem Provers*, (p. 8). York.

Alves, G. V., Dimuro, G. P., & Costa, A. C. (2001). *Um Editor de Provas para Lógica Proposicional.* Universidade Católica de Pelotas, Escola de Informática, Pelotas.

Chang, D., Laurence, D., & Tuovinen, J. E. (2001). Gestalt Theory in Visual Screen Design - A New Look at an Old Subject. *Seventh World Conference on Computers in Education* (p. 8). Copenhagen: Australian Computer Society, Inc.

Gray, P., & Aitken, S. (n.d.). *Visualizing Phases of Activity in Interactive Theorem Proving.* Retrieved Novembro 23, 2008, from Department of Computer Science - University of Glasgow: http://www.dcs.gla.ac.uk/~stuart/ITP/visual-paper.ps

Merriam, N. A., & Harrison, M. D. (1996). Evaluating the Interfaces of Three Theorem Proving Assistants. *Design, Specification and Verification of Interactive Systems*, (p. 22). Namur.

Merriam, N. A., & Harrison, M. D. (1997). What is Wrong with GUIs for Theorem Provers? *User Interfaces for Theorem Provers*, (p. 9). Sophia Antipolis.

Souza, J. N. (2002). *Lógica para Ciência da Computação: Fundamentos de Linguagem, Semântica e Sistemas de Dedução* (4a Edição ed.). Campus Ltda.

Völker, N. (2003). Thoughts on Requirements and Design Issues of User Interfaces for Proof Assistants. *User Interfaces for Theorem Provers*, (pp. 166-181). Rome.

8 APÊNDICES

8.1 ESTUDO DE VIABILIDADE

Antes de iniciar o projeto do Assistente para Dedução Natural, é interessante realizar um estudo de viabilidade para se ter certeza de qual caminho seguir na construção e especificação do sistema. Com tal estudo, tanto o cliente, quanto o desenvolvedor terão uma idéia mais clara do programa computacional a ser desenvolvido. O estudo foi dividido em vários itens, que serão apresentados a seguir:

8.1.1 Funções do Sistema Atual

O domínio do problema consiste, primeiramente, na construção de provas para Lógica Proposicional em Dedução Natural. Com algumas modificações, esse domínio pode ser expandido para abordar a Lógica de Predicados de 1ª Ordem.

Atualmente, existem alguns sistemas que tratam vários tipos de lógica, possibilitando até mesmo a construção de um novo tipo, em que o usuário define as regras. Existem programas que são provadores automáticos e outros que apenas oferecem um ambiente de construção de prova, deixando que o usuário indique as regras a serem utilizadas. Nos ambientes de edição de prova também pode ser oferecido assistência ao usuário, como, por exemplo, indicando quais regras que não podem ser aplicadas em determinado momento.

Embora já existam vários programas computacionais que se propõe a resolver a questão da construção de provas, a idéia do projeto é desenvolver um sistema que ofereça uma assistência adequada e também permita a prova automática de uma conjectura. Além disso, um dos desafios do projeto é desenvolver um método para visualização da prova no formato de árvore de derivação.

A prova de uma conjectura em Dedução Natural, adotando-se o formato de árvore, geralmente inicia-se com o posicionamento do objetivo principal na parte inferior do ambiente de edição de prova, seja ele um papel ou no computador. Logo em seguida são aplicadas várias regras em um processo que é denominado síntese, de tal maneira que a fórmula que representa o objetivo é “quebrada” até se alcançar uma fórmula atômica. Visualmente, a árvore aumenta, de baixo para cima. A partir daí,

inicia-se o processo de análise, que também consiste na aplicação de regras, tentando-se articular objetivos e conclusões intermediárias com hipóteses e premissas do problema. Visualmente a árvore cresce de cima para baixo, até unir as duas partes. Informalmente é dessa maneira que funciona a construção de uma prova em Dedução Natural.

Embora essa descrição relate o processo de construção de uma prova utilizando-se o formato de árvore, é possível realizar essa prova também no formato linear. A idéia é a mesma de síntese e análise. Um detalhamento melhor do processo de construção de uma prova está na seção 8.2 deste mesmo capítulo. Tal seção se refere ao documento de Modelagem de Negócios.

8.1.2 Principais Problemas

8.1.2.1 Apontados pelo Cliente

- O modo de exibição da prova pode variar. Algumas pessoas preferem visualizar a prova no formato de passos, enquanto outras já conseguem visualizar melhor a prova através de uma estrutura de árvore. Ainda há aqueles que preferem visualizar a prova no formato de linguagem natural, ou seja, ao invés da prova estar representada por símbolos e conectivos, ela é descrita com palavras da língua portuguesa, por exemplo.
- O planejamento da construção de uma prova é diretamente afetado pela visualização de dados relevantes, como premissas e objetivos intermediários, e do estado em que se encontra a prova.
- A articulação muitas vezes não é realizada por programas que oferecem ambientes de edição de prova. Entende-se articulação como sendo a associação que é feita entre premissas e objetivos através de regras. Em um determinado estado da prova, uma pessoa é capaz de associar premissas e objetivos com regras, mas não há um algoritmo para realizar tal articulação.
- A construção de provas no papel muitas vezes impossibilita o reuso de trechos de outras provas, fazendo com que a pessoa apenas indique através de um nome ou índice aonde se encontra determinado trecho de prova. Um

programa poderia solucionar tal problema, recortando-se tal trecho e copiando-se no local adequado da prova em construção.

8.1.2.2 Apontados pela Equipe de Desenvolvimento

- Não é trivial a implementação da exibição da prova no formato de árvore de Dedução Natural. A tradução para linguagem natural também pode ser complexa.
- A articulação também pode não ser solucionada, visto que não há um algoritmo para que o programa indique o caminho correto da prova. Mesmo assim é possível fornecer dicas de quais regras são as mais adequadas para se aplicar em um determinado estado da prova. Em último caso, pode-se apresentar todas as possíveis regras para aplicar em um determinado momento da prova.
- A instanciação e validação para reutilização de uma prova ou trecho de prova pode ser complexa, já que um algoritmo de *pattern matching* deverá realizar o processo de associação de átomos de uma prova com os de outra prova.

8.1.3 Necessidades Apontadas pelos Usuários

A fim de coletar dados para aprimorar o Estudo de Viabilidade, foram elaborados alguns questionários relacionados às funcionalidades que os usuários gostariam de ter no provador assistente. É importante ressaltar que essa lista de requisitos não foi baseada somente nas respostas obtidas com o questionário. Como a equipe de desenvolvimento e o ponto focal do cliente têm conhecimento do domínio do problema, outros requisitos foram listados para delimitar da melhor maneira possível o escopo do projeto. Um documento com a lista e a classificação dos requisitos está disponível na seção 8.3 deste capítulo.

8.1.4 Restrições

A equipe deverá buscar alternativas de programas computacionais *freeware* para implementar a solução proposta, ou pelo menos alguma ferramenta cuja licença permita a distribuição do Assistente para alunos e professores.

8.1.5 Programas como Referência

Atualmente existem programas que constroem provas automaticamente, os denominados provadores automáticos, e sistemas que viabilizam um ambiente de edição e construção de provas, os assistentes de construção de provas (*Proof Assistants*). Existem até mesmo os sistemas híbridos que fornecem essas duas funcionalidades.

Após o levantamento do estado da arte, foi realizado um estudo mais detalhado, visando entender melhor a situação atual, adquirir novas idéias e verificar a viabilidade da implementação de alguns dos requisitos especificados pelos *stakeholders*. A seguir é apresentada uma lista com alguns provadores e assistentes utilizados atualmente:

• ***Jape***	• ***NUPRL***
• ***Proofchk***	• ***Proverbox***
• ***Setheo***	• ***Spass***
• ***HOL***	• ***Otter***
• ***Coq***	• ***Gandalf***
• ***Isabelle***	

Tabela 3 - Provadores e Assistentes

Alguns desses programas computacionais foram selecionados para serem analisados com mais detalhes, até mesmo verificando que funcionalidades eles oferecem que se encaixam dentro de alguns parâmetros de análise de interface.

8.1.5.1 Módulo de Dedução Natural do JAPE

O JAPE (Just Another Proof Editor) possui um módulo assistente de construção de provas em dedução natural que apresenta algumas facilidades embora ainda necessite

de muitos recursos para uma boa assistência. Esse módulo também não possui a facilidade de provar o teorema automaticamente.

Primeiramente, sua interface é baseada em menus e em manipulação gráfica, o usuário escolhe a função através de menus e faz a associação de premissas e conclusões a serem utilizadas na aplicação de uma regra através da manipulação gráfica.

A visualização da prova é feita seqüencialmente através de caixas o que melhora a visualização com relação a apresentação puramente seqüencial, porém ainda fica muito distante da representação da prova através da árvore.

Ao se abrir este módulo, várias janelas com algumas conjecturas são apresentadas, conjecturas simples e clássicas que o usuário pode provar para utilizá-las em provas posteriores. Portanto a reutilização de provas é viável nesse sistema. É possível verificar nas listas apresentadas nessas janelas as conjecturas e teoremas que foram provados ou os que ainda necessitam de solução. O usuário pode modificar a lista, entrando com novos teoremas a serem provados.

Esse módulo também possui uma maneira de permite uma certa reflexão. Quando o usuário está realizando alguma prova, ele pode selecionar premissas e suposições e uma conclusão através da manipulação gráfica e em seguida utilizar o menu para a abertura de uma janela que permite a criação de um lema, que na verdade é uma pequena parte da prova. O usuário então pode abstrair-se do problema principal, provar o lema e em seguida retornar com a solução para o problema principal. Também está disponível para o usuário uma opção de criar regras derivadas, que na verdade é um tipo de seqüenciamento de instruções também conhecido como macros ou *tactics*.

Quanto à articulação, o JAPE divide as regras de inferência de dedução natural em dois grupos, separando as regras de introdução das regras de eliminação. Com os métodos de construção de provas de síntese e análise são destacados, evitando assim que o usuário aplique regras de maneira incorreta.

8.1.5.2 Coq

O provador Coq foi testado utilizando-se o módulo de cálculo de seqüentes. Esse assistente consiste basicamente de uma interface de linha de comando, embora exista uma versão da interface com menus CoqIDE mas que somente se diferencia anterior por apresentar as instruções listadas.

A visualização é baseada na organização do cálculo de seqüentes, passo a passo, o que dificulta o planejamento e o entendimento do processo de construção da prova. Em compensação o sistema apresenta facilidades como a possibilidade de destacar o objetivo atual ou a conclusão que deve ser provada em determinado momento.

A reutilização de provas também é possível nesse ambiente, mas como a interface é a linha de comando, esse módulo fica prejudicado. Um exemplo disso é a criação de lemas que deve ser feita antes da realização da prova que o utiliza. Uma propriedade que poderia ser explorada por esse provador seria a viabilização de janelas para manipulação de pequenas provas e lemas, o que pode viabilizar a construção do lema mesmo durante a construção da prova.

8.1.5.3 ProverBox

ProverBox é um provador que fornece a funcionalidade de prova automática e interativa e utiliza o método de resolução para lógica proposicional e lógica de predicados de primeira ordem.

A questão interessante desse sistema é a árvore de cláusulas que ele apresenta para a prova. Pode-se dizer que tal árvore é uma maneira de auxiliar no planejamento da prova e, em outras palavras, um método de assistência.

O provador utiliza uma interface baseada em linha de comando e menus, embora exista o caso da visualização da estrutura de árvore para a prova.

A articulação não é explorada por este sistema e a reflexão, semelhante ao planejamento pode estar associada à exibição da árvore de prova. A reutilização também está disponível nesse programa, já que uma prova pode reutilizar axiomas definidos ou teoremas já provados.

8.1.5.4 Spass

Spass é um provador automático de lógica de predicados de primeira ordem que utiliza o método de resolução para verificar a validade de conjecturas. Semelhante aos programas analisados anteriormente, o Spass utiliza uma interface de linha de comando, embora exista uma versão da interface com menus. Uma questão interessante desse provador é que existe uma versão dele disponível para web. Basta o usuário enviar um

formulário com as especificações do problema, utilizando a sintaxe do provador, e este retornará uma resposta em uma nova página. Caso a prova exista, ela é exibida. Além disso, também é possível habilitar ou desabilitar o uso de determinadas regras pelo provador automático.

8.1.6 Proposta

8.1.6.1 Funções do sistema e outros requisitos

A proposta principal é viabilizar um ambiente de edição de provas para Lógica Proposicional. Para construção de tais provas, serão disponibilizadas as regras de inferência do método de Dedução Natural. O programa verificará automaticamente se a prova construída pelo usuário está completa ou não.

Funções de assistência serão implementadas como: limitar quais regras podem ser aplicadas em um determinado momento da prova em uma fórmula especificada e associar fórmulas da base de premissas e hipóteses com objetivos que devem ser alcançados.

O sistema também permitirá a visualização da prova tanto no formato linear quanto no formato de árvore.

Para um melhor entendimento das funções e outros requisitos do Assistente para Dedução Natural, referir-se à seção 8.3, que contém o documento de Elicitação de Requisitos.

8.1.6.2 Recursos para operação do sistema

Inicialmente o sistema será desenvolvido para executar em qualquer sistema operacional Windows. Caso seja possível e os prazos de entrega do sistema não estejam atrasados, uma versão para Linux também poderá ser disponibilizada. Fora a questão do sistema operacional, nada mais necessita ser mencionado como recurso necessário para operação do sistema.

8.1.6.3 Recursos para desenvolvimento do sistema

A princípio o desenvolvimento do sistema será realizado com a utilização de uma ferramenta *freeware* denominada Qt. Tal ferramenta viabiliza a construção de interfaces gráficas de maneira simples e rápida. Além disso, para implementação de códigos e classes definidas para o sistema, deverá ser utilizado o IDE Eclipse-CDT/Mingw, também *freeware*. Como já pode ser visto, a linguagem a ser utilizada para implementação do sistema será a linguagem C++.

8.1.6.4 Modelo de processo adotado

O modelo de processo de desenvolvimento visado para essa proposta é uma adaptação do Rational Unified Process (RUP). Tal modelo foi escolhido como base devido aos seguintes fatos:

- A utilização de um padrão de documentação, o que favorece a facilidade para o desenvolvimento de novas versões do sistema por outras equipes de desenvolvimento.
- O foco em uma arquitetura baseada em componentes, o que facilita a expansão do sistema.
- O foco no desenvolvimento de modelos visuais e diagramas, tornando possível uma abstração da solução proposta desde o início do projeto.
- O modelo é baseado em um processo iterativo, de maneira que em cada etapa do processo, uma versão do sistema é lançada com alguma funcionalidade nova. Desse modo o cliente pode opinar e guiar o desenvolvimento do sistema desejado.

Embora esse modelo de processo seja indicado para grandes equipes de desenvolvimento e empresas, ele pode ser facilmente adaptado para projetos pequenos, como é o caso.

8.1.6.5 Cronograma

Iniciação e Planejamento:

Estudo de Viabilidade: Consiste no levantamento do estado da arte e estudo do domínio do problema. Nesse estudo algumas propostas serão entregues ao cliente como solução do programa solicitado por ele.

Definição da Proposta: Essa atividade consiste na escolha da proposta. Esse documento é equivalente a entrega do termo da abertura do projeto contendo a proposta escolhida do Estudo de Viabilidade.

Validação: Corresponde a validação da proposta de projeto. Para isso todas as partes comprometidas devem assinar o documento da proposta de projeto.

Concepção:

Elaboração do Plano do Projeto: Consiste no detalhamento da proposta escolhida do Estudo de Viabilidade.

Modelagem de Negócios: Modelagem do Problema. Listagem das regras de Dedução Natural para Lógica Proposicional e regras para construção de provas nesse tipo de Lógica.

Análise de Requisitos: Levantamento de Requisitos. Detalhamento dos requisitos, classificando-os quanto a funcionais e não-funcionais. Delimitar o escopo do projeto utilizando essa classificação.

IHC: Levantamento dos requisitos de interface. Elaboração de um questionário e definição do perfil do usuário principal.

Revisão da Proposta: Revisão dos documentos de Plano do Projeto, Modelagem de Negócios e Elicitação de Requisitos.

Elaboração:

Análise e Projeto: Elaboração dos Diagramas de Análise e Projeto: Diagrama de Casos de Uso; Diagrama de Classes (Análise); Diagrama de Classes (Projeto); Diagrama de Estados (Provador); Diagrama de Componentes;

IHC: É definido o *Storyboard*. Esse documento contém a especificação das telas do programa e uma descrição das respectivas funcionalidades.

Definição da Arquitetura: A idéia é documentar como o assistente funciona em termos de componentes.

Construção:

Definição da Arquitetura Solução: Primeiramente é feito uma revisão dos diagramas criados na etapa de Elaboração e então uma arquitetura final é proposta como solução para o software.

Implementação: Tendo passado por todo o processo de planejamento do software, iniciam-se as iterações de desenvolvimento do sistema. A idéia é que a cada iteração, uma nova funcionalidade ou requisito seja atendido.

Gestão de Mudanças: Controle do projeto e garantia da qualidade do produto final. Qualquer revisão ou alteração solicitada pelo usuário deve ser primeiro avaliada pela equipe do projeto para que todos os impactos sejam levados em consideração.

Testes unitários: A cada nova versão do assistente, ou seja, a cada nova iteração, serão realizados testes unitários para verificar a integridade e a corretude de cada módulo e função implementada.

Transição:

Entrega da versão final: Propõe-se uma apresentação da solução para os usuários finais e a disponibilização da ferramenta para download no site do projeto.

Demandas evolutivas: Com a divulgação e distribuição do programa, é possível realizar uma pesquisa sobre a receptividade dos usuários e assim propor melhorias e evoluções no software.

Manutenção: É provável que erros e bugs sejam descobertos com mais freqüência na etapa de Transição. Para isso deve-se manter uma equipe para realizar essas correções e fazer a manutenção do assistente.

8.1.6.6 Riscos

É a primeira vez que a equipe de desenvolvimento adota o modelo de processo citado. Como tal modelo foca bastante no desenvolvimento de diagramas, um tempo maior do que o normalmente necessário pode ser gasto. Uma maneira de mitigar esse risco é fazer com que a equipe de desenvolvimento familiarize-se com o modelo de processo antes do início do projeto.

O ambiente de desenvolvimento utilizado para implementação da interface também não foi utilizado anteriormente pela equipe. Entretanto esse risco não é tão alto já que existe suporte para a ferramenta e, além disso, basta a equipe estudá-la antes do início do projeto. Outro ponto interessante é que a comunidade de desenvolvedores que utilizam essa ferramenta está crescendo consideravelmente. Isso facilita a troca de informações e resoluções de problemas.

8.1.6.7 Benefícios

No final do projeto haverá uma documentação padronizada que poderá ser utilizada por outras pessoas a fim de aprimorar ou expandir o assistente. Também se pode notar que essa proposta acrescenta funcionalidades, até então não muito exploradas pelos programas atuais, como:

- A escolha da forma de exibição da prova,
- A maneira de exibir dados relevantes durante a construção de uma prova,
- A viabilização de janelas de rascunho para reflexão durante a construção de uma prova,
- A maneira de se construir uma prova, baseado na edição de estruturas, e
- Assistência provida pelo sistema quando o usuário estiver construindo uma prova como, por exemplo, sugestões de quais regras devem ser aplicadas;

Tais funcionalidades visam solucionar os problemas que foram listados anteriormente durante o processo de construção de uma prova em Dedução Natural para Lógica Proposicional.

A seguir está uma tabela que associa as soluções propostas pelo projeto para os problemas listados pelo representante dos clientes e o desenvolvedor que também está familiarizado com o domínio do problema:

Problema	Solução
Algumas pessoas preferem visualizar a prova no formato de passos, enquanto outras já conseguem visualizar melhor a prova através de uma estrutura de árvore.	Permitir a escolha da forma de exibição da prova.
O planejamento da construção de uma prova é diretamente afetado pela visualização de dados relevantes, como premissas e objetivos intermediários, e do estado em que se encontra a prova.	Exibir dados relevantes durante a construção de uma prova, através de grupos de dados: conclusões intermediárias, suposições e premissas.
A construção de provas no papel muitas vezes impossibilita o reuso de trechos de outras provas, fazendo com que a pessoa apenas indique através de um nome ou índice aonde se encontra determinado trecho de prova.	Viabilizar de janelas de rascunho para reflexão durante a construção de uma prova e reutilização de trechos de provas.
A articulação muitas vezes não é realizada por programas que oferecem ambientes de edição de prova. Como já foi estudado anteriormente, entende-se articulação como sendo a associação que é feita entre premissas e objetivos através de regras. Em um determinado estado da prova, uma pessoa é capaz de associar premissas e objetivos com regras, mas não há um algoritmo para realizar tal articulação.	Prover assistência enquanto o usuário estiver construindo uma prova como, por exemplo, sugestões de quais regras devem ser aplicadas.
Facilidade e rapidez na construção de uma prova muitas vezes é prejudicada quando as fórmulas em questão são muito extensas.	Permitir a construção de uma prova, baseado na edição de estruturas ou menus.

Tabela 4 - Benefícios: Problema x Solução

Sugestões dos clientes que responderam ao questionário realizado durante o estudo de viabilidade também serão levadas em consideração. A seguir estão algumas respostas e comentários interessantes que geraram requisitos não - funcionais desejáveis para o sistema. Com essas respostas é possível confirmar a utilidade dos benefícios listados anteriormente para um sistema como esse.

Questão 1: Como você se organiza quando realiza a construção de uma prova utilizando o método de Dedução Natural? Faz um rascunho? Segue das hipóteses para a conclusão ou busca retroagir da conclusão para as hipóteses, ou tenta ambos os sentidos?

A maior parte dos alunos respondeu que faz um rascunho durante a construção de uma prova. Além disso, a maioria prefere utilizar o método de *backward chaining*, retroagindo da conclusão para as hipóteses.

Questão 2: Como você prefere visualizar uma prova? No formato linear através de passos ou por uma estrutura de árvore?

Aluno 1: *"- Através de passos escritos na linguagem natural, no nosso caso o português, é bem mais visível."*

Questão 3: Você acha interessante dividir os elementos que compõem uma prova e destacá-los em grupos durante a construção de uma? (Por exemplo: Objetivo Principal, Hipóteses, Premissas, Objetivos Intermediários, Conclusões Intermediárias).

Aluno 2: *"-Sim, pois isso torna o procedimento mais organizado, evitando erros e,no caso de não se obter a solução logo na primeira tentativa, pode-se encontrar o erro mais rapidamente."*

Questão 4: Utilizando um provador, o que você considera melhor: construir uma prova editando estruturas gráficas (arrastando regras e fórmulas pelo ambiente) ou através de menus? Seria interessante viabilizar os dois modos?

Também foi unânime a resposta de que o modo de edição de estruturas gráficas é mais amigável do que o modo de menus, embora a viabilização dos dois modos seja interessante.

Questão 5: De que modo um provador poderia auxilia-lo na construção de uma prova? Por exemplo, quando você sente dificuldade em um determinado momento durante a construção, o que poderia ser feito?

Aluno 3: *"- Acho que podería haver uma espécie de "central de ajuda"..."*

Aluno 4: *"-Poderia haver um botão de ajuda contendo algumas estratégias de prova, para que quando sentíssemos dificuldade em algum momento, pudéssemos clicar neste botão e ter acesso a estas estratégias."*

8.2 MODELAGEM DE NEGÓCIOS

8.2.1 Introdução

O processo a ser analisado consiste na construção da prova formal de uma conjectura da Lógica Proposicional utilizando o método conhecido como Dedução Natural. Esse método é lecionado nas universidades nas disciplinas de Lógica como uma introdução a esse ramo da Matemática.

Atualmente, além do método tradicional de ensino, algumas universidades utilizam sistemas conhecidos como Provadores Automáticos (*Provers*) ou Assistentes para Construção de Provas (*Proof Assistants*), para lecionar as diversas metodologias de prova. Entretanto, nem sempre esses sistemas atendem as expectativas ou as necessidades dos usuários que podem ser professores, alunos ou pesquisadores. Considerando isso, um dos objetivos do projeto é analisar tais necessidades e implementar um sistema que possa atendê-las satisfatoriamente.

A primeira atividade a ser realizada é a modelagem de negócios, que consiste na observação e análise do processo de construção de provas e na metodologia de ensino comumente utilizada. Desse modo é possível construir um sistema que viabilize tanto a prova automática quanto funções de assistência que sejam interessantes para o ensino do método de prova em questão. Alguns conceitos importantes da Lógica Proposicional serão abordados primeiramente e logo em seguida os conceitos relevantes do sistema de Dedução Natural serão levantados para este projeto.

8.2.2 A Lógica Proposicional: Sintaxe e Semântica

Toda a parte das regras de negócio foi abordada no capítulo 2 da monografia. Há uma seção específica para a Lógica Proposicional.

8.2.3 O Método de Dedução Natural

Toda a parte das regras de negócio foi abordada no capítulo 2 da monografia. Há uma seção específica para o método de Dedução Natural.

8.3 ELICITAÇÃO DE REQUISITOS

8.3.1 Introdução

A fim de coletar dados para elicitação e análise de requisitos, foram elaboradas algumas questões relacionadas às funcionalidades que os usuários gostariam de ter no Assistente para Construção de Provas. O questionário pode ser visualizado no final desse documento, seção 2.5. É importante ressaltar que essa lista de requisitos não foi baseada somente nas respostas obtidas com o questionário. Como a equipe de desenvolvimento e o ponto focal do cliente têm conhecimento do domínio do problema, outros requisitos foram listados para delimitar da melhor maneira possível o escopo do projeto.

Também é importante ressaltar que esse documento envolve apenas os requisitos do programa de maneira geral. Não foi abordado aqui a elicitação dos requisitos de interface e a definição de um perfil de usuário.

8.3.2 Requisitos Funcionais

Como a funcionalidade principal do sistema é viabilizar um ambiente para construção de provas interativamente, foram considerados como requisitos funcionais os seguintes tópicos:

[R001] – Definir objetivamente a linguagem utilizada pelo sistema (símbolos proposicionais, conectivos proposicionais, símbolos de pontuação e símbolos de verdade);
[R002] – Detectar erros de sintaxe correspondentes à lógica proposicional;
[R003] – Possibilitar a aplicação de uma regra de inferência do método de Dedução Natural, dado que premissas ou uma conclusão foram selecionadas;
[R004] – Embora todas as regras sejam apresentadas, viabilizar a utilização somente das que podem ser aplicadas a uma fórmula em um determinado momento da construção da prova.
[R005] – Viabilizar sistema de ajuda (tutoriais e manuais) com estratégias de prova mais utilizadas;

[R006] – Verificar se a prova de uma conjectura foi concluída com êxito;
[R007] – Visualizar a prova no formato linear ou de script;
[R008] – A interface com o usuário será baseada em menus e manipulação gráfica;

8.3.3 Requisitos Não-Funcionais

A maioria dos requisitos não-funcionais apresentados a seguir é interessante que seja implementada, já que também são funcionalidades para auxílio na construção de uma prova.

[R009] – Apresentar dicas e/ou sugestões de decisões a serem tomadas durante a construção de uma prova;
[R010] – Possibilitar o uso de janelas de rascunho para manusear pequenas provas e auxiliar na reflexão do problema;
[R011] – Agrupar vários comandos (regras de inferência da lógica proposicional) em um só comando (macros);
[R012] – Salvar uma prova, completa ou não;
[R013] – Exportar a prova para o formato texto;
[R014] – Reutilizar provas;
[R015] – Visualizar uma prova no formato de árvore;
[R016] – Visualizar uma prova no formato de linguagem natural;
[R017] – Realizar a prova de uma conjectura automaticamente ou avisar caso não exista uma prova;
[R018] – Manter em um arquivo as listas de conjecturas, teoremas e suas respectivas provas, com a finalidade de recuperar tais informações após o sistema ser fechado;
[R019] – Apresentar através de menus pop-up, previsões de aplicações de regras sobre uma determinada conclusão ou premissa.

8.3.4 Requisitos Inversos

[R020] – A primeira versão do sistema não dará suporte à lógica de predicados;
[R021] – Não serão implementados outros métodos de construção de provas, além do método de Dedução Natural.

8.3.5Questionário das Necessidades do Cliente

1) Como você se organiza quando realiza a construção de uma prova utilizando o método de Dedução Natural?

2) Como você prefere visualizar uma prova? No formato linear através de passos ou por uma estrutura de árvore?

3) Você acha interessante dividir os elementos que compõem uma prova e destacá-los em grupos durante a construção de uma? (Por exemplo: Objetivo Principal, Hipóteses, Premissas, Objetivos Intermediários, Conclusões Intermediárias).

4) Utilizando um provador, o que você considera melhor: construir uma prova editando estruturas gráficas (arrastando regras e fórmulas pelo ambiente) ou através de menus? Seria interessante viabilizar os dois modos?

5) De que modo um provador poderia auxiliá-lo na construção de uma prova? Por exemplo, quando você sente dificuldade em um determinado momento durante a construção, o que poderia ser feito?

8.4 PLANO DE GARANTIA DA QUALIDADE

8.4.1 Introdução

Visando controlar a qualidade do sistema proposto, foi elaborado este documento que contém as principais medidas a serem adotadas durante o processo de desenvolvimento.

Todas as etapas do desenvolvimento serão cobertas pelo plano de garantia de qualidade, em maior ou menor grau, de acordo com a relevância de cada fase para o projeto. Por exemplo, segundo o processo escolhido no plano do projeto, a etapa de transição é uma etapa muito importante, uma vez que será nessa etapa que o cliente e alguns usuários poderão expor suas opiniões sobre o resultado final do projeto e propor alterações para o software.

A equipe do projeto terá a responsabilidade de garantir que o processo de software foi seguido de maneira adequada, assim como os padrões de engenharia de software foram corretamente adotados. Também deverá ser realizado um acompanhamento através de baselines sobre o cronograma do projeto. A cada transição de fase um baseline será salvo para posterior comparação.

8.4.2 Atividades Propostas

A seguir são listadas as atividades propostas a fim de garantir a qualidade do sistema:

- Adoção de uma estratégia de gerência de configuração (Seção 4 – Estratégia de Gerência de Configuração),
- A cada etapa finalizada, a revisão de todos os documentos gerados,
- Verificação periódica da adequação do projeto ao processo adotado,
- Verificação periódica da adequação dos documentos aos padrões de engenharia de software adotados (Seção 3),
- Análise das métricas e parâmetros relevantes para o projeto (Seção 5 – Métricas e Parâmetros), e
- Testes de unidade e integração.

8.4.3 Documentação Gerada

São listados a seguir os produtos a serem obtidos durante todo o processo de desenvolvimento do sistema. Tais produtos devem ser revisados pela equipe a fim de melhorar a qualidade não só do sistema final, mas também de toda documentação do projeto.

- Diagramas UML:
  - Diagramas de Casos de Uso
  - Diagramas de Classes (Análise e Projeto)
  - Diagramas de Estados (Para as entidades em que tal diagrama é relevante)
- Documentação:
  - Estudo de Viabilidade
  - Modelagem de Negócios
  - Elicitação de Requisitos
  - Plano de Garantia da Qualidade
  - Storyboard e documentação da interface
- Código Fonte

8.4.4 Estratégia de Gerência de Configuração

A fim de permitir o múltiplo acesso a um repertório comum de componentes e a manipulação de diversas versões diferentes de um mesmo produto do projeto, será utilizada uma ferramenta online provida pelo Google, disponível em http://groups.google.com. O controle de versões será feito incrementado de uma unidade o número da última versão do documento, partindo do número 1. Exemplo: Apêndice III – Elicitação de Requisitos V1.

A página destinada ao projeto final está no seguinte endereço: http://groups.google.com.br/group/adena-proofassistant.

8.4.5 Padrões, Práticas e Convenções

A norma de documentação a ser adotada será um padrão para todos os documentos gerados durante o processo de desenvolvimento de software. Tal norma é a indicada para o desenvolvimento de monografias para a Universidade Federal Fluminense. Como o projeto a ser realizado é voltado para a universidade, a norma de documentação a ser adotada será o padrão divulgado pela instituição.

O projeto será desenvolvido segundo a arquitetura de orientação a objetos (OO). Os padrões de diagramas são todos estabelecidos de acordo com a linguagem UML. O código gerado será na linguagem C++ com padronização de todos os termos em inglês.

8.4.6 Métricas e Parâmetros

Uma métrica a ser utilizada no projeto será uma avaliação da receptividade do cliente. Como ao longo do processo serão desenvolvidos vários protótipos, será possível obter uma medida de como o sistema está atendendo as necessidades dos clientes. Para simplificar o estudo serão especificados apenas quatro níveis: nulo, baixo, parcial e completo. Com tal métrica será possível verificar se o desenvolvimento do sistema está seguindo um caminho adequado ou não. Um gráfico também será elaborado contendo todos os resultados anteriores para fins de acompanhamento da evolução da receptividade do cliente.

Além disso, outros parâmetros avaliados serão: Correção, Facilidade de Uso, Modularidade e Autodocumentação:

A correção avalia o quanto o documento de requisitos atendeu as necessidades do cliente.

A facilidade de uso poderá ser analisada durante o desenvolvimento a cada entrega de um protótipo para validação. Nesse caso, o cliente poderá classificar o sistema quanto ao nível de facilidade de uso de maneira bem simples (fácil, razoável, difícil).

A modularidade também deve ser analisada uma vez que implica na facilidade de manutenção e entendimento do código. Dados que ajudaram na análise da modularidade do sistema serão coletados durante a fase de construção e transição, quando o código fonte do sistema for gerado.

A autodocumentação também é um parâmetro importante, pois permitirá a manutenção e expansão do sistema posteriormente. Será coletada na fase de transição e poderá ser analisada por integrantes da equipe de desenvolvimento e pessoas que não estejam envolvidas no projeto, com o objetivo de saber se o nível de documentação do código está ideal.

8.5 DIAGRAMAS DE ANÁLISE E PROJETO

8.5.1 Introdução

Primeiramente será realizada uma análise do sistema, que consiste basicamente no entendimento e descrição do sistema e seus relacionamentos com o meio externo. Isso será feito através da modelagem dos diagramas de casos de uso e diagramas de classe de análise.

Posteriormente, a etapa de projeto é iniciada. O diagrama de classes é fundamental nessa etapa, pois será ele que indicará quais serão as classes a serem implementadas na etapa de codificação. O diagrama elaborado durante a etapa de análise será refinado a fim de se ter um detalhamento de como implementar uma solução para o sistema.

8.5.2 Diagrama de Casos de Uso

Este é o diagrama de casos de uso elaborado para análise do sistema.

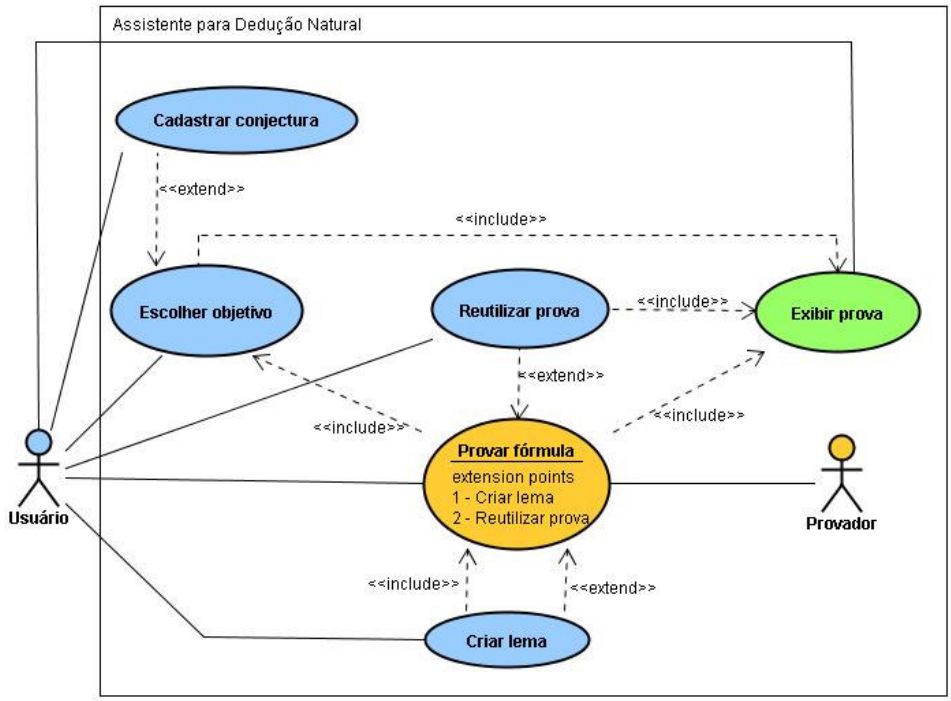

Figura 20 - Diagrama de Casos de Uso

Com ele é possível visualizar as principais funções do provador assistente, além de validar os requisitos apresentados no Apêndice III – Elicitação de Requisitos. Nesse diagrama os casos de uso são ativados por agentes externos, os atores, que nesse caso são o usuário e o módulo responsável pela construção automática da prova de uma conjectura, o provador.

O provador é um módulo do sistema responsável pela construção e verificação automática de prova de conjecturas. Posteriormente será descrito o funcionamento deste elemento através de um diagrama de estados que caracteriza a metodologia de construção de prova utilizada.

O usuário tem a capacidade de decidir quais ferramentas utilizar e quais decisões tomar durante o processo de construção de prova. Diferentemente do Provador, esse ator pode escolher um objetivo, criar um lema ou reutilizar uma prova, atividades envolvidas no processo de provar uma fórmula.

8.5.3 Descrição dos Casos de Uso

A seguir os casos de uso e suas principais características são definidas. Esta seqüência de tabelas será utilizada para validação dos requisitos e posteriormente para modelagem dos diagramas de classe para análise e projeto.

Caso de uso:	**Escolher objetivo**
Resumo:	O usuário seleciona uma fórmula já provada ou uma conjectura, ou cadastra uma nova conjectura.
Pré-condições:	É apresentada ao usuário uma lista de teoremas e uma lista de conjecturas. Uma opção de cadastramento de uma nova conjectura também está disponível.
Resultado:	A fórmula escolhida como objetivo da prova é apresentada em uma janela de edição.
Fluxo Principal:	01 - Usuário seleciona uma conjectura na lista de conjecturas. (*extend* – Cadastrar conjectura) 02 - Sistema abre uma janela de edição de prova com a fórmula escolhida como o objetivo.

Fluxo alternativo:	01 - Usuário seleciona um teorema na lista de teoremas. 02 - Usuário pode optar por visualizar as provas já realizadas. (*include* – Exibir prova) 03 - Sistema abre uma janela de edição de prova com o teorema a ser provado novamente.

Tabela 5 - Caso de Uso: Escolher objetivo

Caso de uso:	**Cadastrar conjectura**
Resumo:	Usuário cadastra uma fórmula da Lógica Proposicional e o Sistema a insere na lista de conjecturas.
Pré-condições:	----------
Resultado:	Uma conjectura é adicionada a lista de conjecturas do sistema.
Fluxo Principal:	01 – Usuário entra com uma fórmula da Lógica Proposicional. 02 – Sistema valida a fórmula. 03 – Sistema insere a fórmula na lista de conjecturas do sistema.
Fluxo de Exceções:	A fórmula não corresponde a uma fórmula bem formulada (fbf) da Lógica Proposicional. O sistema indica onde está o erro de sintaxe para o usuário corrigi-lo.

Tabela 6 - Caso de Uso: Cadastrar conjectura

Caso de uso:	**Provar fórmula**
Resumo:	O usuário constrói a prova de uma conjectura/teorema ou solicita ao provador para apresentá-la.
Pré-condições:	Uma conjectura ou um teorema a ser provado foi selecionado. (Caso de uso: Escolher objetivo).
Resultado:	O sistema cadastra a prova do teorema, retirando-o da lista de conjecturas, inserindo-o na lista de teoremas.

Fluxo Principal: (Agente: usuário)	01 – Usuário seleciona um objetivo. (include - Escolher objetivo). 02 – Sistema cria a janela de edição de prova e exibe a prova. (include – Exibir prova). 03 - Usuário verifica a possibilidade de reutilizar uma prova. (*extend* - Reutilizar uma prova) 04 - Usuário verifica a possibilidade de utilizar um lema. (*extend* - Criar lema) 05 - Usuário seleciona argumentos, premissas ou conclusão. 06 - Usuário seleciona regra para aplicar. 07 - Sistema valida aplicação da regra. 08 - Sistema efetua alteração na prova. 09 - Sistema testa se a prova foi finalizada. 10 - Se a prova não terminou, voltar ao passo 2. 11 - Sistema salva a prova.
Fluxo Alternativo: (Agente: provador)	Nesse fluxo, ao invés do usuário construir uma prova, o próprio sistema o faz. O fluxo é o mesmo que o apresentado no fluxo principal. A única diferença é que o não haverá a possibilidade de criar um lema ou reutilizar uma prova. Além disso, logo após o sistema criar a janela de edição de prova, o usuário deve indicar que o sistema deverá construir a prova automaticamente.
Fluxo de exceções:	Caso a prova não seja realizada com sucesso, esgotando as possibilidades de aplicações de regras, o sistema deve parar e transferir tal teorema para uma lista de teoremas com provas sem sucesso. Caso o usuário não consiga construir uma prova, ele pode utilizar a funcionalidade de prova automática para verificar a existência da prova.

Tabela 7 - Caso de Uso: Provar fórmula

Caso de uso:	**Reutilizar prova**
Resumo:	O usuário realiza uma busca na lista de teoremas para verificar se é possível reutilizar alguma prova. Caso seja possível, o sistema valida a reutilização da prova e insere na prova em processo de construção.
Pré-condições:	A construção de uma prova esteja sendo realizada.
Resultado:	A prova a ser reutilizada é inserida na prova que está em processo de construção.
Fluxo Principal:	01 - Usuário está construindo uma prova X. 02 - Usuário seleciona uma prova a ser reutilizada Y. (*include* – Exibir prova) 03 - Sistema valida prova a ser reutilizada Y. 04 - Sistema insere a prova Y na prova atual X.
Fluxo de exceções:	Quando não existem provas a serem reutilizadas, não há o que fazer. Portanto essa opção deve estar desabilitada.

Tabela 8 - Caso de Uso: Reutilizar prova

Caso de uso:	**Criar lema**
Resumo:	O usuário cria um lema.
Pré-condições:	Uma prova está em construção.
Resultado:	A interface apresenta o lema a ser provado em uma janela de edição. O usuário dá um nome para o lema e isso servirá de referência na prova principal. A princípio o usuário pode deixar para provar o lema posteriormente, mas nesse caso a prova principal só será dada como completa quando o tal lema for provado.

Fluxo Principal:	01 - Usuário solicita criação do lema para uma determinada fórmula. 02 - Sistema viabiliza uma janela com um editor de prova. 03 - Usuário define um nome para o lema. 04 - Sistema insere o nome do lema na prova em processo de construção. 05 - Usuário constrói a prova do lema. (*include* – Provar fórmula). 06 - Sistema salva o lema.

Tabela 9 - Caso de Uso: Criar lema

Caso de uso:	**Exibir prova**
Resumo:	O sistema apresenta ao usuário a(s) prova(s) de teorema(s) ou conjectura(s).
Pré-condições:	----------
Resultado:	A prova, completa ou não, de uma fórmula é apresentada.
Fluxo Principal:	01 - Uma fórmula foi selecionada. (*include* – Escolher objetivo). 02 - A prova é exibida de forma linear na janela de edição de provas. 03 - Sistema viabiliza opções de visualização de prova. (árvore e script).

Tabela 10 - Caso de Uso: Exibir prova

8.5.4 Diagrama de Classes – Análise

Abaixo é apresentado o diagrama de classes de análise. A partir deste momento já é possível visualizar uma modularização do sistema em classes que poderão ser implementadas depois de um refinamento do diagrama na etapa de projeto. Os métodos e os atributos das classes ainda não foram definidos porque eles representam uma possível solução para o sistema, o que não faz parte dessa etapa.

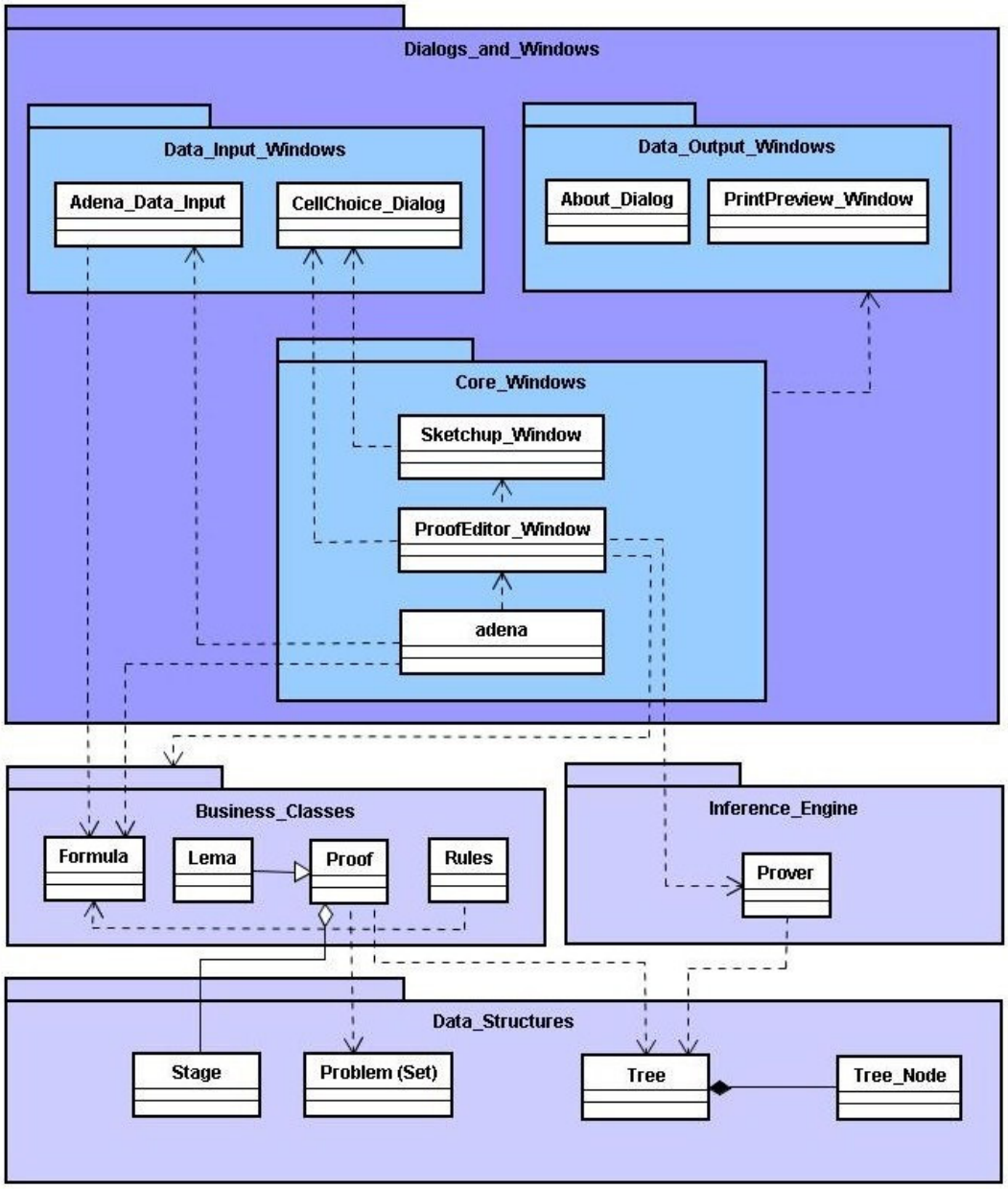

Figura 21 - Diagrama de Classes (Análise)

A visão inicial proposta por esse diagrama de classes mostra que o software é conceitualmente dividido em três camadas: uma camada de estrutura de dados, uma camada que envolve as classes que implementam regras de negócio e uma camada que trata da interface com o usuário.

A camada de estrutura de dados irá prover algumas abstrações e adaptações de estruturas de dados conhecidas, como um conjunto ou uma árvore, a fim de facilitar a implementação da camada seguinte referente as classes de regras de negócio. Nessa camada estão as seguintes classes: *Stage*, *Problem*, *Tree* e *Tree_Node*.

Para facilitar o controle do processo de construção de uma prova, foi proposta a classe *Stage*. Essa classe será responsável por manter armazenado o estado da estrutura de uma prova em um determinado momento. As classes *Tree* e *Tree_Node* são responsáveis por armazenar a estrutura de árvore de derivação.

Por fim, a classe *Problem* é uma abstração dos conjuntos de hipóteses, premissas e objetivos durante o processo de construção de uma prova. Os conjuntos de hipóteses e objetivos serão constantemente alterados durante a construção da prova e por isso será necessário essa estrutura para facilitar operações sobre tais conjuntos.

A camada de regras de negócio envolve um conjunto de classes que foi denominado *Business_Classes* e uma outra classe que será responsável pelo *Inference_Engine* (Motor de Inferência), que na verdade é o provador automático. No conjunto das *Business_Classes* estão as seguintes classes: *Formula*, *Lema*, *Proof*, *Rules.*

A classe *Formula*, como o próprio nome diz, implementa uma fórmula da Lógica Proposicional. Nessa classe estão definidos alguns métodos que tratam da verificação se uma fórmula é bem-formulada ou não.

A classe *Proof* é a entidade que armazena as informações referentes ao processo de construção de prova, assim como uma prova completa de um teorema. Foi definida como classe derivada de *Proof*, a classe *Lema* que consiste basicamente numa pequena prova com um nome. E por fim, a classe *Rules* é a entidade que vai tratar do processamento das fórmulas com base nas regras do método de Dedução Natural.

No diagrama também é possível notar que um objeto da classe *Proof* é uma agregação de objetos da classe *Stage*. Em outras palavras, uma prova contém zero ou mais estágios, enquanto um estágio pertence à somente uma prova. Além disso, a classe *Proof* utiliza-se da classe *Problem* para poder controlar os conjuntos de hipóteses, premissas e objetivos e assim dar início ao processo de verificação da finalização da prova.

Finalmente será descrita a camada de interface, que é o foco do software desenvolvido. Essa camada foi dividida em três partes: *Data_Input_Window*, *Data_Output_Windows* e *Core_Windows*.

Em *Data_Input_Windows* estão as classes *Adena_Data_Input* e *CellChoice_Dialog*. A classe *Adena_Data_Input* é uma classe que provê a janela de cadastro de conjecturas no assistente para construção de provas. Por isso é considerada a principal classe de entrada de dados para o software.

Já a classe *CellChoice_Dialog* é uma janela responsável pela indicação de qual célula deve ser utilizada para o posicionamento do resultado de aplicação de uma regra de Dedução Natural. Logo também deve ser classificada como uma janela de entrada de dados, já que ela apenas pega dados informados pelo usuário e os transmite ao Core da interface.

Em *Core_Windows* estão as classes mais importantes do software e que efetivamente estruturam todo o ambiente de edição de uma prova. São elas: *adena*, *ProofEditor_Window* e *Sketchup_Window*. A classe *adena* trata da janela principal do software, ou seja, é a primeira janela que aparece e, portanto, contém as listas de conjecturas e resultados obtidos. A partir dela é possível abrir a janela de cadastro de conjecturas ou o editor de provas.

As outras duas classes implementam o ambiente de edição de prova. A classe *ProofEditor_Window* trata do ambiente de edição principal, aonde é realizado o processo de construção de uma prova no sentido de síntese. Já a classe *Sketchup_Window* provê a mesma interface para construção de uma prova, embora essa janela só possa ser acessada por meio da janela de edição de prova principal, a *ProofEditor_Window*. A diferença da Sketchup_Window para a ProofEditor_Window é o propósito de cada uma. Enquanto a ProofEditor_Window permite a construção de uma prova no sentido de síntese, a Sketchup_Window foi feita para permitir a construção de uma prova no sentido de análise. Daí o nome Sketchup (Rascunho) uma vez que não é necessário que o conteúdo obtido nessa janela seja transferido para a ProofEditor_Window, que no caso apresenta a prova da conjectura.

Em *Data_Output_Windows* estão as classes que apenas transmitem informações para o usuário. São as classes: *About_Dialog* e *PrintPreview_Window*.

A classe About_Dialog apenas apresenta uma janela com algumas informações do sistema, enquanto a janela implementada pela classe PrintPreview_Window apresenta uma janela de visualização da impressão.

Considera-se assim que os diagramas essenciais da fase de análise estão elaborados: O diagrama de casos de uso descreve bem os relacionamentos entre o mundo externo e o sistema, destacando as principais funcionalidades de tal sistema. O diagrama de classes já apresenta uma nova idéia, introduzindo o conceito de classes, descrevendo o sistema por meio de entidades e camadas.

8.5.5 Diagrama de Estados

Um diagrama necessário ao desenvolvimento do Assistente para Construção de Provas, pelo fato de permitir um melhor entendimento do processo e da dinâmica do problema, é o diagrama de estados. Para esse software será elaborado um diagrama de estados para o provador. Esse diagrama será utilizado somente para implementar a funcionalidade de prova automática. A partir dele é possível entender as principais funções do Provador e como elas estão relacionadas.

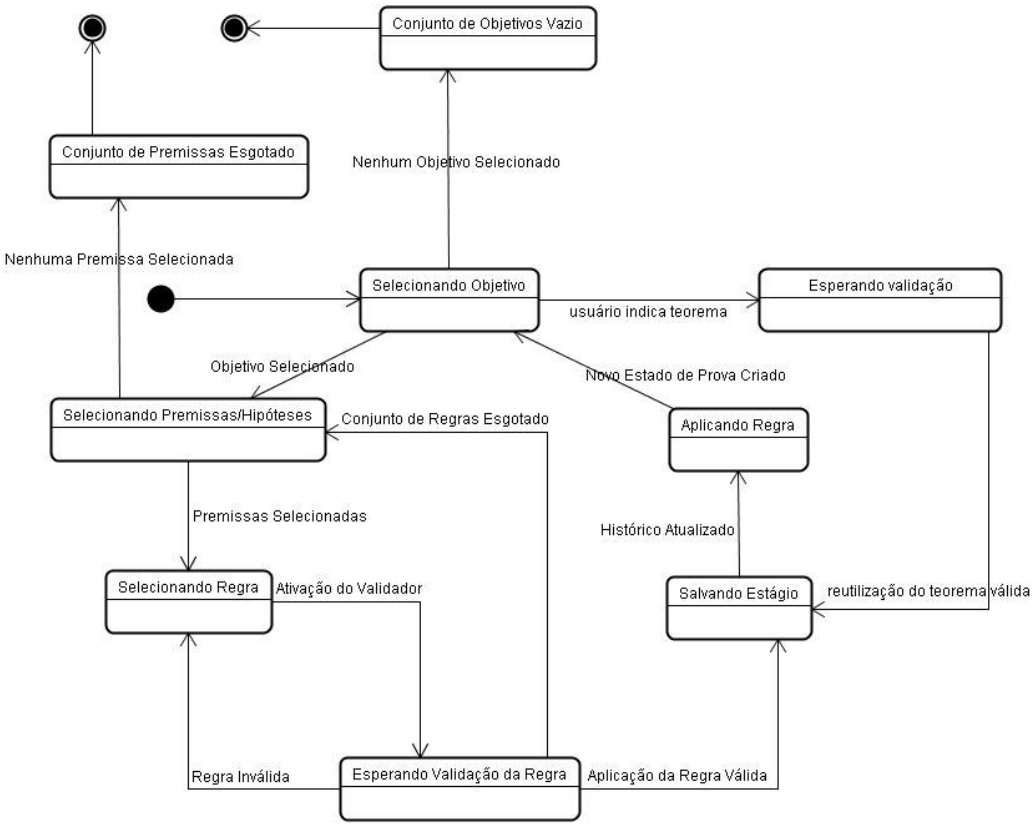

Figura 22 - Diagrama de Estados - Provador

Nesse diagrama, cada caixa representa um estado do objeto Provador e as setas indicam os eventos que viabilizam a transição de estados. Esse modelo não contém qualquer heurística de construção de prova, o Provador apenas seleciona um objetivo e tenta prová-lo a partir das hipóteses, premissas e regras disponíveis. O círculo totalmente preenchido aponta para o estado inicial do provador enquanto o círculo parcialmente preenchido indica os estados finais de todo processo, quando o objeto provador é destruído.

8.5.6 Diagrama de Classes - Projeto

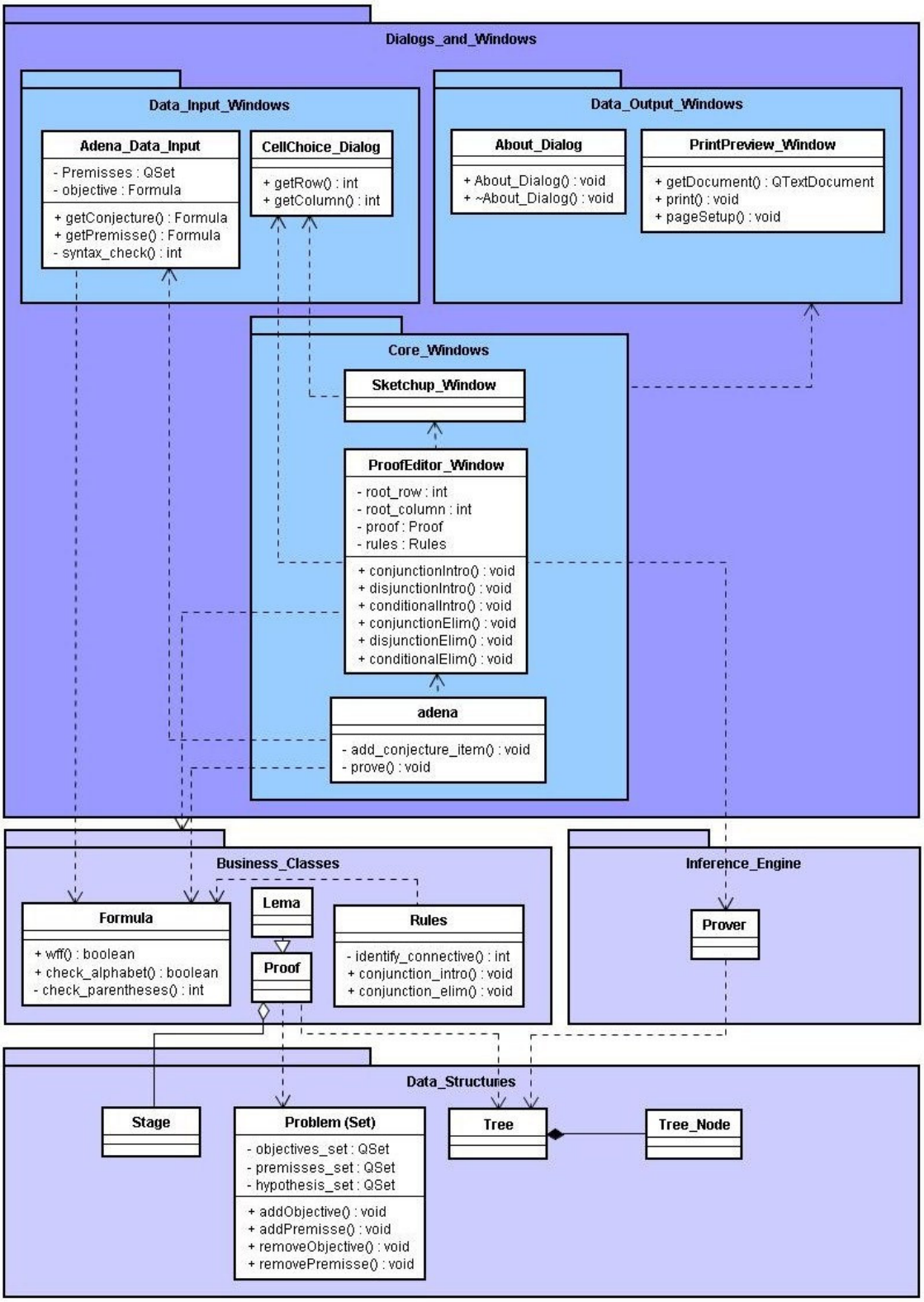

Figura 23 - Diagrama de Classes (Projeto)

O diagrama de classes de projeto é apresentado a partir de um refinamento do diagrama elaborado para análise. Os métodos das classes já foram definidos e atributos foram adicionados. O foco deste modelo de diagrama de classes consiste nos relacionamentos entre as diversas classes e as funções por elas disponibilizadas. Os detalhes de implementação como estruturas de dados, definição de tipos e controle de passagem de parâmetros fazem parte da fase de codificação. Além disso, deve-se deixar claro que os métodos e atributos de cada classe não são limitados aos que foram apresentados nesse diagrama. As classes que estão sem métodos ou atributos definidos no diagrama ainda não foram implementadas e portanto devem ser estudadas melhor durante o projeto para que elas possam suportar as funções das outras classes.

Para a camada de estrutura de dados, a classe *Problem* funcionará como um DAO (*Data Access Object*) e portanto irá prover métodos como adicionar, remover e alterar para cada um dos conjuntos definidos durante o momento de construção de uma prova. Tais conjuntos, como já foi explicado anteriormente, controlam as fórmulas geradas durante a construção de uma prova. São eles: conjunto de hipóteses, conjunto de premissas e conjunto de objetivos.

Os métodos das outras classes (*Stage*, *Tree* e *Tree_Node*) ainda precisam ser definidos e estudados com maior cautela para garantir que todas as funções necessárias para a camada de regras de negócio e de interface sejam suportadas pela camada de estrutura de dados.

Na camada de regras de negócio, uma das classes fundamentais é a classe que define o conjunto de regras de derivação do método de Dedução Natural. Os métodos apresentados na classe *Rules* consistem na aplicação de cada regra de dedução natural sobre um conjunto de premissas, hipóteses e/ou objetivos. O identificador de cada método para aplicação de uma regra tem os sufixos *Intro* ou *Elim* indicando que a respectiva regra é de introdução ou de eliminação. Além disso as regras de redução ao absurdo (RAA) e silogismo disjuntivo (SD) também são apresentadas. O método *identify_connective()* é utilizado para identificar o conectivo associado a aplicação da regra de Dedução Natural.

Os métodos na classe *Formula* estão relacionados a validação da sintaxe. Nesse caso, há um método geral (*wff() : boolean*) que verifica se uma fórmula é bem-formulada ou não. Os outros dois métodos listados no diagrama são utilizados por essa função *wff()*.

As classes *Proof* e *Lema* ainda precisam ter seus métodos e atributos definidos. Esses vão depender dos métodos fornecidos pela camada de estruturas de dados.

Finalmente, na camada interface estão os principais métodos que tratarão da interação do usuário com o assistente de construção de provas. No diagrama está apresentado uma pequena parte dos métodos e atributos definidos para cada classe, apenas como exemplo do que foi implementado.

8.6 APRESENTAÇÕES PARA O CLIENTE

Na apresentação da primeira versão do sistema para os usuários, foram observados alguns pontos que devem ser revistos. A reunião foi realizada no dia 24/04 para alunos e professores do Grupo de Lógica da Universidade Federal Fluminense. Estavam presentes:

Nome	Função
Renata de Freitas	Professora do GAN-IM
Petrucio	Professor do GAN-IM
Hugo Nobrega	Aluno de IC do GAN-IM
Cecília Englander	Aluna de Mestrado da PUC-Rio
Marcelo Corrêa	Professor do GAN-IM e Orientador

Após uma breve apresentação do trabalho e do que foi realizado até o momento com relação a implementação do sistema, foram discutidas melhorias para este. A seguir estão alguns pontos levantados pelos presentes na reunião:

1. Tendo em vista que o sistema tem o propósito de auxiliar no ensino de Lógica, foi proposto que a maneira como as conjecturas são listadas na tela inicial obedeça as regras formais da Lógica Proposicional, diferenciando-se Tautologia de Conseqüência Lógica.
2. Como sugestão para implementação da árvore de prova, foi proposto utilizar cores diferentes para destacar as fórmulas e os nomes das regras que forem utilizadas.
3. A janela de rascunho gerou uma grande discussão que apenas levou a um *brainstorming* de idéias que podem ser consideradas na construção desse componente. Dentre as sugestões estavam:
    a. Reorganização dos elementos dentro da janela,
    b. Permitir utilizar regras derivadas (somente na janela de rascunho ou não?), e
    c. Deixar livre o posicionamento das fórmulas e a aplicação de regras, deixando que o usuário possa escrever em qualquer célula (isso iria permitir o usuário corromper a estrutura da árvore de provas...).

4. Reorganização dos elementos que compõem a janela de rascunho, bem como repensar os nomes utilizados para esses componentes (Objetivos Intermediários, Conclusões Intermediárias...) e para a própria janela de rascunho.
5. Sugeriu-se prosseguir na implementação da estrutura de prova para o ambiente principal. Ao se ter uma visualização melhor de como funcionará o ambiente, a discussão sobre a janela de rascunho será retomada.