ISABEL GOMES BARBOSA

OBTENÇÃO DE RESPOSTAS BASEADAS EM CASOS A PARTIR DE ÁRVORES DE PROVA

Belo Horizonte
18 de julho de 2008

ISABEL GOMES BARBOSA
Orientador: Newton José Vieira

OBTENÇÃO DE RESPOSTAS BASEADAS EM CASOS A PARTIR DE ÁRVORES DE PROVA

Proposta de dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Belo Horizonte
18 de julho de 2008

UNIVERSIDADE FEDERAL DE MINAS GERAIS

FOLHA DE APROVAÇÃO

Obtenção de Respostas Baseadas em Casos a Partir de Árvores de Prova

ISABEL GOMES BARBOSA

Proposta de dissertação defendida e aprovada pela banca examinadora constituída por:

D. Sc. Newton José Vieira – Orientador
Universidade Federal de Minas Gerais

D. Sc . José Lopes de Siqueira Neto
Universidade Federal de Minas Gerais

D. Sc. Marcos Alexandre Castilho
Universidade Federal do Paraná

Belo Horizonte, 18 de julho de 2008

Agradecimentos

Inicialmente, agradeço ao professor Newton, pelos ensinamentos e pela dedicação a este trabalho. Agradeço aos meus pais, pelo amor, carinho e constante apoio a minha trajetória. Agradeço aos amigos e demais familiares, pelo apoio e pela amizade durante o período de mestrado. Agradeço aos professores do departamento de Ciência da Computação da UFMG por, através de seus conhecimentos, contribuírem para a minha formação de mestre. E, por fim, agradeço ao CNPQ, pelo suporte financeiro.

Resumo

Este trabalho investiga o conjunto de respostas para consultas existencialmente quantificadas no contexto de um sistema de prova que representa as suas deduções na forma de árvores de prova. Em particular, nas situações em que o sistema gera uma resposta disjuntiva para a consulta do usuário, este trabalho propõe que uma resposta baseada em casos deveria ser fornecida ao invés. Inicialmente, é apresentada uma definição formal de respostas nesse contexto, a qual considera que mesmo árvores de prova geradas nos passos intermediários de uma dedução produzem respostas em potencial para a consulta. Particiona-se, ainda, o conjunto de respostas existentes segundo a definição dada em três classes de respostas: extensionais, intensionais e hipotéticas, e mostra-se que qualquer um desses tipos de respostas pode corresponder a uma resposta disjuntiva. Para as situações em que o sistema gera uma resposta disjuntiva da forma $P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$, é proposto um algoritmo que, a partir da dedução da resposta na forma de árvore de prova, determina uma resposta baseada em casos para a mesma consulta. Uma *resposta baseada em casos* consiste em uma resposta definida em função de um número finito e exaustivo de casos no domínio do problema, sendo que cada caso implica em uma resposta não disjuntiva $P(A_i)$ para a consulta.

Abstract

This dissertation investigates the set of answers to existentially quantified questions in the context of a system that represents its deductions by proof trees. In particular, in those situations in which a disjunctive answer is obtained, this dissertation proposes that a case-based answer should be provided instead. First, it is presented a formal definition of answers in the context of a proof tree based system, which considers that even proof trees generated at the intermediate steps of a deduction may produce possible answers to the question. The set of answers in accordance with the given definition is then partitioned into three answer classes, termed extensional, intensional and hypothetical answers, and it is showed that any of these three answer types may correspond to a disjunctive answer. To those situations where the system produces a disjunctive answer that has the general form $P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$, it is proposed an algorithm that from the answer deduction in the form of a proof tree, is capable to determine a case-based answer to the exact same question. A case-based answer is an answer defined in terms of a finite and exhaustive number of cases in the problem domain, where each case implies in a non disjunctive answer $P(A_i)$ to the question.

Sumário

Capítulo 1

Introdução

Obter informações precisas que nos auxiliem em nossas atividades é essencial nos dias de hoje. Na busca por uma informação, consultamos fontes que acreditamos ser capazes de nos ajudar a obtê-la. A utilidade de uma consulta está relacionada ao quanto a mesma atende à necessidade de informação daquele que a realizou.

O advento da tecnologia e a explosão no número de informações levou a criação de sistemas computacionais capazes de armazenar, manipular e recuperar a informação disponível. *Sistemas de consulta-resposta*[1] são aqueles que fornecem respostas a consultas feitas pelos seus usuários a partir da pesquisa a fontes de informações (documentos ou base de dados). Tais sistemas tem sido estudados sob o foco de diferentes subáreas em Inteligência Artificial (IA), sendo encontradas pesquisas nas áreas de Processamento de Linguagem Natural, Representação do Conhecimento e Sistemas Inteligentes, e resultando nas mais variadas abordagens para a construção de um sistema de consulta-resposta.

O uso da prova de teoremas como um modelo para o desenvolvimento de sistemas de consulta-resposta tem sido há longo tempo uma abordagem explorada em IA, com o intuito de se utilizar a capacidade de inferência presente nos provadores de teoremas para deduzir respostas para as consultas. Exemplos de trabalhos iniciais a utilizarem essa abordagem são (Green, 1969; Luckham e Nilsson, 1971; Reiter, 1977), os quais baseavam-se em um provador de teoremas por resolução.

Este trabalho tem dois objetivos principais, ambos aplicados ao contexto de um provador de teoremas que representa as suas deduções no formato de uma árvore de prova: (i) apresentar uma definição completa e formal de respostas para as consultas existencialmente quantificadas feitas pelos usuários; (ii) propor uma solução para os ca-

[1] Em inglês, *question-answering systems*.

sos em que o sistema gera uma resposta disjuntiva da forma $P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$. Essa solução envolve a realização de transformações na árvore de prova gerada como refutação pelo sistema, a fim de que seja obtida uma *resposta baseada em casos* para a consulta. Esses objetivos serão melhor especificados ao longo deste capítulo.

1.1 A Prova de Teoremas aplicada a Sistemas de Consulta-Resposta

A prova de teoremas foi utilizada como um primeiro modelo para o desenvolvimento de sistemas de consulta-resposta em Inteligência Artificial (Green, 1969; Luckham e Nilsson, 1971; Reiter, 1977). Nesse modelo, o conhecimento do sistema acerca do mundo é representado como um conjunto de axiomas ou hipóteses e a consulta como uma conjectura a ser verificada. O processo de se verificar a validade da conjectura corresponde ao processo de se deduzir a resposta para a consulta do usuário. Ou seja, se for encontrada uma prova de que a conjectura segue do conjunto de axiomas e hipóteses, então a resposta para a consulta é "sim". Caso contrário, a resposta é "não" (hipótese do mundo fechado)[2] ou ainda "não sei".

Esse modelo é, no entanto, apropriado quando a conjectura associada à consulta contém apenas *termos fechados*[3]. Como exemplo, para a consulta "Edson cursa a disciplina Compiladores?", uma resposta do tipo "sim/não" é esperada e provavelmente suficiente. Por outro lado, um modelo que se limite a fornecer respostas desse tipo não é capaz de lidar adequadamente com consultas como "Quais disciplinas Edson cursa?". De fato, tais consultas requerem como resposta uma especificação daqueles objetos ou indivíduos no domínio que satisfazem as propriedades indicadas, no caso, as propriedades "é uma disciplina" e "é cursado por Edson". Consultas como essa podem usualmente ser expressas da forma: *determinar* X *tal que* $P(X)$, em que X é uma n-upla de variáveis $x_1, x_2, \ldots, x_n$ e P são as propriedades que devem ser satisfeitas por X.

Para que pudessem ser aplicados para a recuperação de informação e/ou resolução de problemas era necessário modificar os provadores de teoremas, adicionando algum mecanismo para a extração de informação a partir das provas geradas pelos mesmos.

[2] A *hipótese do mundo fechado* é feita quando se presume que apenas é verdadeiro aquilo que está explicitamente declarado na base de conhecimentos, ou que é explicitamente derivável da mesma.

[3] Um *termo fechado* é um termo que não contém variáveis.

O método de prova por resolução, ao mesmo tempo que encontra uma refutação para $\exists X P(X)$, determina valores para X tais que $P(X)$. Para extrair tais valores, a solução proposta por Green (1969) foi adicionar um literal especial, denominado *literal de resposta*, à conjectura a ser provada. O papel do literal de resposta consiste em coletar as subtituições feitas pelo procedimento de prova por resolução durante a busca por uma refutação da negação da conjectura. Quando uma refutação é encontrada, os argumentos presentes nos literais de resposta especificam os valores que as variáveis existencialmente quantificadas da conjectura devem assumir para se provar a sua validade.

Dessa forma, o trabalho de Green permitiu que um provador de teoremas por resolução fosse capaz de fornecer como resposta objetos que satisfazem propriedades. Respostas dessa forma são denominadas *respostas extensionais* e consistem no paradigma dominante de respostas fornecidas pelos sistemas de consulta-resposta atuais. Como exemplo, dada a consulta "Quais disciplinas Edson cursa?", se {Compiladores, Redes} são os objetos que satisfazem as propriedades "é disciplina" e "é cursado por Edson", então são respostas extensionais para a consulta: "Compiladores" e "Redes". Alguns autores, porém, consideram que a resposta extensional deve incluir a conjectura associada à consulta original (instanciada pelos objetos). Utilizando essa abordagem, as respostas extensionais para a consulta seriam: "Edson cursa a disciplina Compiladores" e "Edson cursa a disciplina Redes".

A geração de uma resposta extensional está acoplada à obtenção de uma prova. Assim, se apenas respostas extensionais são fornecidas, então na ausência de uma prova nenhuma resposta é retornada ao usuário. Na busca por uma prova, podem ser obtidas/geradas informações relevantes para a consulta do usuário. Essas informações, se recuperadas, poderiam ser utilizadas para se responder às consultas feitas pelos usuários, mesmos nos casos de ausência de uma prova para a conjectura associada à consulta.

A informação intensional existente em uma base de conhecimentos pode incluir caracterizações adicionais a respeito de uma resposta extensional, tornando-se explícito porque um conjunto de objetos responde a uma consulta. Essas caracterizações ou descrições dos objetos que constituem uma resposta para a consulta são conhecidas na literatura como *respostas intensionais*. Como exemplo, se toda disciplina do 2^o período de Ciência da Computação é cursada por Edson, então uma resposta intensional para a consulta exemplo seria: "Edson cursa todas as disciplinas do 2^o período de Ciência da Computação". Em muitos casos, respostas intensionais estão mais próximas

da maneira como os seres humanos raciocinam, uma vez que as mesmas descrevem grupos ou categorias, ao invés de listar/identificar cada objeto. Dessa forma, trabalhos desenvolvidos na literatura procuraram adicionar aos sistemas de consulta-resposta, e em particular aos provadores de teoremas, a habilidade de se produzir respostas intensionais (Cholvy, 1990; Motro, 1994; Cholvy e Demolombe, 1986).

Cholvy e Demolombe (1986) apresentam uma definição de respostas intensionais em bases de dados lógicas. Segundo os autores, uma resposta intensional para uma consulta $P(X)$ é toda fórmula $ans(X)$ tal que $(\forall X)\ (ans(X) \rightarrow P(X))$ segue da base de conhecimentos.

Em adição às respostas extensionais e intensionais, alguns autores ainda consideram a classe de *respostas condicionais* (Demolombe, 1992). Tais respostas possuem associada uma condição que deve ser satisfeita para a resposta ser considerada correta. Como exemplo, "se Marcelo cursa a disciplina Compiladores, então Edson cursa a disciplina Compiladores" é uma resposta condicional. Respostas condicionais são usualmente fornecidas quando não é possível deduzir uma resposta extensional ou intensional para a consulta.

O trabalho de Burhans (2002) apresenta uma definição formal e completa de respostas no contexto de um provador de teoremas por resolução. O procedimento de prova por resolução consiste na resolução repetida das cláusulas, com o objetivo de se gerar novas cláusulas, denominadas resolventes, até que seja derivada a cláusula vazia, indicando a existência de uma prova, ou o procedimento finalize por algum outro motivo, ou não finalize. Aplicando-se uma estratégia de busca apropriada[4], todos os resolventes descendem de cláusulas derivadas da consulta original. Burhans propõe um modelo no qual todos esses resolventes são considerados relevantes para a consulta e são respostas em potencial para a consulta. Burhans particiona, então, o conjunto de resolventes em três classes que correspondem a três tipos de respostas: específicas, genéricas e hipotéticas, as quais correspondem às classes de respostas extensionais, intensionais e condicionais, respectivamente. Essa divisão permite caracterizar formalmente e semanticamente cada tipo de resposta.

[4]Utilizando o conjunto de cláusulas correspondentes à negação da consulta como "conjunto de suporte".

1.2 Motivação e Objetivos

Os sistemas de consulta-resposta foram tradicionalmente projetados para fornecer respostas diretas às suas consultas. No entanto, ao longo dos anos, diversos trabalhos na literatura preocuparam-se em expandir os tipos de respostas fornecidas por tais sistemas. E trabalhos com esse propósito continuam a ser desenvolvidos, com o intuito de que os sistemas de consulta-resposta atendam cada vez mais à necessidade de informação dos seus usuários, fornecendo respostas precisas e que sejam capazes de auxiliá-los em suas tarefas.

A investigação desse trabalho ocorre no contexto de um provador de teoremas que representa as suas deduções na forma de uma árvore de prova. A representação das deduções em forma de árvores de prova possibilita uma "leitura" intuitiva dos passos realizados pelo sistema, facilitando a manipulação de tais estruturas para o fornecimento de explicações a usuários leigos. O conhecimento utilizado em uma dedução, no contexto de explicações, é visto não como um conjunto de cláusulas, mas como uma cadeia de implicações levando à conclusão procurada (Vieira, 1987).

O primeiro objetivo deste trabalho é adaptar a definição e classificação de respostas feita por Burhans (2002), originalmente para sistemas que se fundamentam no método de resolução, para sistemas de prova baseados em árvores de prova. Desse modo, torna-se possível definir em que se constitui uma resposta no contexto de um provador de teoremas que representa as suas deduções em forma de árvores de prova, além de comparar os sistemas construídos em tal contexto em relação às respostas que os mesmos reconhecem. No entanto, ao contrário de Burhans, que não impõe limitações na forma da consulta, a classificação de respostas aqui apresentada é feita considerando-se apenas consultas existencialmente quantificadas. Dessa forma, não são considerados os demais tipos de consultas, como consultas universalmente quantificadas ou consultas que possuem ambos os quantificadores.

O segundo e mais importante objetivo deste trabalho refere-se às situações em que o sistema gera uma resposta disjuntiva. Apesar da ampla pesquisa existente sobre o uso de sistemas de consulta-resposta com ênfase na prova de teoremas, poucos trabalhos na literatura têm-se preocupado em propor soluções quando o sistema gera uma resposta disjuntiva para a consulta do usuário. Uma resposta disjuntiva pode ser definida como: um conjunto de respostas, sendo que pelo menos uma resposta desse conjunto é verdadeira. Em certas aplicações, uma resposta disjuntiva não é aceitável por ser considerada imprecisa para auxiliar o usuário na tomada de decisão. A impre-

cisão existente em uma resposta disjuntiva é decorrente do fato de não sabermos quais respostas, dentre as que compõem a resposta disjuntiva, são corretas.

Considere uma consulta da forma

determinar X tal que $P(X)$,

onde X é uma n-upla de variáveis $x_1, x_2, \ldots, x_n$, que resulte em uma resposta disjuntiva da forma

$$P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k),$$

$k \geqslant 2$, em que cada A_i é uma n-upla termos fechados. Supondo a dedução da resposta disjuntiva acima representada na forma de uma árvore de prova, o segundo objetivo deste trabalho é a proposta de um algoritmo que, a partir dessa árvore de prova, determine um conjunto $v_1, v_2, \ldots, v_m$ finito de casos, de modo a gerar uma *resposta baseada em casos* para a mesma consulta da forma

$$\begin{aligned} &v_1 \rightarrow P(A_{i_1}) \\ &v_2 \rightarrow P(A_{i_2}) \\ &\ldots \\ &v_m \rightarrow P(A_{i_m}) \end{aligned}$$

onde $1 \leqslant i_1, i_2, \ldots, i_m \leqslant k$, v_j é um caso que implica na veracidade de $P(A_{i_j})$ e $v_1 \vee v_2 \vee \ldots \vee v_m$ é uma conseqüência lógica da base de conhecimentos.

Quando deparado com uma resposta dessa forma, o conhecimento de que um caso v_j é verdadeiro no contexto do problema considerado, permite ao usuário derivar uma resposta não disjuntiva $P(A_{i_j})$ para a sua consulta.

1.3 Organização do Trabalho

Os demais capítulos deste trabalho estão organizados da seguinte maneira. Nos capítulos 2 e 3 são descritos os fundamentos importantes para a leitura do trabalho: no capítulo 2 são apresentados conceitos básicos da lógica e é feita uma revisão da lógica de predicados; no capítulo 3 é descrito o sistema formal baseado em árvores de provas proposto em (Vieira, 1996) e é mostrado como uma refutação pode ser realizada nesse sistema formal. No capítulo 4 é apresentada uma definição formal de respostas no contexto de um sistema de prova baseado em árvores de prova e as respostas geradas segundo essa definição são particionadas em três classes de respostas: extensionais,

intensionais e hipotéticas. No capítulo 5, o mais importante do trabalho, é descrito o algoritmo proposto para a geração de respostas baseadas em casos a partir de árvores de prova que correspondem a respostas disjuntivas. E, por fim, no capítulo 6 são apresentadas as conclusões e trabalhos futuros.

Capítulo 2

Lógica

Eis duas definições de lógica:

> O estudo da lógica é o estudo dos métodos e princípios usados para distinguir o raciocínio correto do incorreto (Copi, 1978).

> Lógica é o estudo dos métodos para determinar se uma conclusão pode ou não ser corretamente inferida a partir de um conjunto de suposições (Bessie e Glennan, 2000).

O estudo da lógica em ciência da computação tem como objetivo especificar uma linguagem que permita representar o conhecimento e raciocinar em cima desse conhecimento de maneira formal (Huth e Ryan, 2000).

Um sistema lógico consiste basicamente em um conjunto de sentenças (ou fórmulas) representando fatos do mundo real e um conjunto de procedimentos que podem ser aplicados sobre essas sentenças, visando produzir novos conhecimentos. As sentenças são especificadas por meio de uma linguagem constituída de uma sintaxe e de uma semântica bem definidas. A sintaxe diz respeito principalmente à formulação da linguagem, ou seja, inclui um conjunto de símbolos e determina como esses símbolos podem ser combinados para construir fórmulas bem-formadas. A semântica de uma linguagem lógica, por outro lado, preocupa-se com a interpretação que deve ser dada a uma fórmula, ou seja, como a fórmula se relaciona com o conhecimento em questão.

Uma interpretação que torna uma fórmula α verdadeira é dita ser um *modelo* para α. Um modelo para um conjunto de fórmulas $\Gamma = \{\beta_1, \beta_2, \ldots, \beta_n\}$ é uma interpretação na qual todas as fórmulas do conjunto são verdadeiras. Uma fórmula α é *conseqüência lógica* de um conjunto de fórmulas Γ, denotado por $\Gamma \models \alpha$, se, e somente se, todo modelo

de Γ é um modelo de α. Diz-se que duas fórmulas α e β são *equivalentes*, denotado por $\alpha \equiv \beta$, se, e somente se, $\{\alpha\} \models \beta$ e $\{\beta\} \models \alpha$.

A noção correspondente a conseqüência lógica no nível formal é a de *dedução*. Um *sistema dedutivo* para uma lógica é um conjunto de axiomas lógicos e um conjunto de regras de inferências que determinam que seqüências de fórmulas constituem deduções válidas do sistema. Dado um sistema dedutivo, diz-se que uma fórmula α é dedutível a partir de um conjunto de fórmulas Γ, denotado por $\Gamma \vdash \alpha$ se, e somente se, existe um conjunto finito de fórmulas $\beta_1, \beta_2, \ldots, \beta_n$ tal que β_n é α e cada β_i é: (i) um axioma lógico, ou (ii) uma fórmula de Γ, ou (iii) o resultado da aplicação de uma regra de inferência a fórmulas anteriores a β_i.

É desejável que um sistema dedutivo possua duas características: correção e completude. Um sistema dedutivo é dito ser *correto* se permite apenas deduções corretas, isto é:

$$\text{se } \Gamma \vdash \alpha \text{ então } \Gamma \models \alpha.$$

Por outro lado, um sistema dedutivo é dito ser *completo* se tudo que é conseqüência lógica de um conjunto de hipóteses pode ser deduzido pelo sistema, ou seja:

$$\text{se } \Gamma \models \alpha \text{ então } \Gamma \vdash \alpha.$$

2.1 Lógica de 1^a Ordem

Considere o seguinte argumento:

Todo homem é mortal. Sócrates é um homem. Logo, Sócrates é mortal.

Para verificar a validade desse argumento é preciso uma linguagem lógica que seja capaz de representar os detalhes da estrutura interna das proposições, ou seja, na qual seja possível dizer que um objeto (Sócrates) de um certo contexto ou universo de discurso possui uma certa propriedade (ser mortal).

A lógica de 1^a ordem, também conhecida como lógica de predicados de 1^a ordem, é uma extensão da lógica proposicional que permite modelar uma realidade em termos dos objetos que a constituem e de suas propriedades ou predicados. Dessa forma, através do uso da lógica de predicados, pode-se fazer referências a objetos (Maria, Jose, casa) de um universo de discurso, atribuir propriedades (novo, bonito, alto) a esses objetos, e representar funções (raiz quadrada de, pai de) ou relações (são irmãos,

são colegas) sobre tais objetos. Para representar asserções que exprimam a idéia de que todos os objetos de um universo de discurso possuem uma certa propriedade (todo homem é mortal) ou que um objeto específico do universo de discurso possui uma certa propriedade (existe um número divisível por 5) são introduzidos nessa lógica os quantificadores $\forall$ (universal) e $\exists$ (existencial).

O poder expressivo da lógica de 1^a ordem nos permite deduzir casos particulares de uma propriedade geral dos objetos de um universo de discurso, assim como concluir generalizações a partir de fatos que valem para um objeto específico do universo de discurso. A seguir, definimos a sintaxe e a semântica para a linguagem de 1^a ordem.

2.1.1 A Linguagem de 1^a Ordem

O alfabeto da lógica de predicados é constituído dos seguintes símbolos:

- *Símbolos não lógicos*: (i) um conjunto de *constantes* $a, b, c, \ldots$, com ou sem índices, para denotar objetos do universo; (ii) um conjunto de *símbolos funcionais* $f, g, h, \ldots$, com ou sem índices, para denotar funções; (iii) um conjunto de *símbolos predicativos* $p, q, r, s, \ldots$, com ou sem índices, para denotar relações, além dos símbolos $\bot$ (falso) e $\top$ (verdadeiro). A cada símbolo funcional ou predicativo vem associado o número de argumentos do mesmo. Símbolos funcionais de aridade zero são denominados constantes, e símbolos predicativos de aridade zero são denominados variáveis proposicionais.

- *Símbolos lógicos:* (i) os *conectivos lógicos*: $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$, $\forall$, $\exists$; (ii) um conjunto de *variáveis* $x, y, z, \ldots$, com ou sem índices.

- *Símbolos auxiliares*: "(" e ")".

A linguagem de 1^a ordem possui dois tipos de expressões lógicas distintas: os *termos* e *as fórmulas propriamente ditas*. O conjunto dos *termos* da linguagem de 1^a ordem é definido recursivamente pelas seguintes regras:

a) uma variável é um termo;

b) uma constante é um termo;

c) se f é um símbolo funcional e $t_1, t_2, \ldots, t_n$ são termos, então $f(t_1, t_2, \ldots, t_n)$ é um termo;

d) só são termos as palavras que podem ser obtidas como mostrado em a, b e c.

Exemplo 1. São exemplos de termos: a (função de aridade 0, constante), x (variável), $f(x, a)$, $f(g(x), a)$.

O conjunto de *fórmulas bem formadas* da linguagem de 1^a ordem é definido recursivamente da seguinte forma:

a) se $t_1, t_2, ..., t_n$ são termos e p é um símbolo predicativo, então $p(t_1, t_2, ..., t_n)$ é uma fórmula, denominada fórmula atômica;

b) se α é uma fórmula, então $(\neg\alpha)$ é uma fórmula;

c) se α e β são fórmulas, então $(\alpha \wedge \beta)$, $(\alpha \vee \beta)$, $(\alpha \rightarrow \beta)$ e $(\alpha \leftrightarrow \beta)$ são fórmulas;

d) se α é uma fórmula e x é uma variável, então $(\exists x\alpha)$ e $(\forall x\alpha)$ são fórmulas.

e) só são fórmulas as palavras que podem ser obtidas como mostrado em a, b, c e d.

Exemplo 2. Constituem exemplos de fórmulas atômicas: $p(a, x)$, $p(x)$, $r(f(g(x), a), x)$.

Embora a simbologia definida para constantes, símbolos funcionais e símbolos predicativos seja útil quando deseja-se fazer um tratamento formal da lógica de predicados, em certos casos é mais interessante utilizar uma simbologia que nos forneça uma idéia clara do que está sendo representado, como exemplificado a seguir.

Exemplo 3. Uma maneira usual de representar o argumento

Todo homem é mortal. Sócrates é um homem. Logo, Sócrates é mortal.

na linguagem de predicados é a seguinte:

$$\forall x(homem(x) \rightarrow mortal(x))$$
$$homem(Socrates)$$
$$mortal(Socrates),$$

onde *Socrates* é uma constante; *homem* e *mortal* são predicados.

2.1.2 Semântica da Linguagem

Para verificar a validade de uma fórmula é preciso inicialmente atribuir significado aos símbolos não lógicos, ou seja, é preciso mapear os símbolos não lógicos para objetos, funções e relações em um domínio, denominado universo de discurso ($\mathcal{U}$). Uma *interpretação* de uma linguagem de 1^a ordem em um universo não vazio $\mathcal{U}$ é uma função I, definida da seguinte forma:

- para cada constante a, a^I é um elemento de $\mathcal{U}$;
- para cada símbolo funcional n-ário f, $n > 0$, $f^I : \mathcal{U}^n \to \mathcal{U}$ é uma função n-ária; e
- para cada símbolo predicativo n-ário p, $p^I \subseteq \mathcal{U}^n$ é uma relação n-ária.

A interpretação de um termo ou fórmula pode ainda variar conforme a interpretação de suas variáveis livres. Assim, é necessário definir uma função de valoração ρ que atribui a cada variável um elemento do universo de discurso, ou seja, $\rho : \mathcal{V} \to \mathcal{U}$, onde $\mathcal{V}$ é o conjunto de variáveis.

Com isso, pode-se definir a interpretação de um termo, $\mathcal{I}_\rho$, com respeito a uma valoração ρ, recursivamente:

a) $\mathcal{I}_\rho(x) = \rho(x)$ para cada variável x;

b) $\mathcal{I}_\rho(a) = a^I$ para cada constante a;

c) $\mathcal{I}_\rho(f(t_1, t_2, ..., t_n)) = f^I(\mathcal{I}_\rho(t_1), \mathcal{I}_\rho(t_2), ..., \mathcal{I}_\rho(t_n))$ para cada símbolo funcional f e número natural n.

Definimos ainda a função $\rho[x \to a]$ da seguinte maneira:

$$\rho[x \to a](y) = \begin{cases} a & \text{se } x = y; \\ \rho(y) & \text{caso contrário.} \end{cases}$$

Dada uma interpretação I para os símbolos não lógicos (constantes, funções, predicados) em um universo de discurso $\mathcal{U}$ e uma função de valoração ρ, denotados por $(I^{\mathcal{U}}, \rho)$, podemos definir a interpretação de uma fórmula na lógica de 1^a ordem. Se uma fórmula φ é verdadeira em $\mathcal{I} = (I^{\mathcal{U}}, \rho)$, então diz-se que $\mathcal{I}$ é um modelo para φ, denotado por $\mathcal{I} \models \varphi$.

A relação $\mathcal{I} \models \varphi$ é definida por uma recursão na estrutura da fórmula:

$\mathcal{I} \models p(t_1, t_2, \ldots, t_n)$	sse	$(\mathcal{I}_\rho(t_1), \mathcal{I}_\rho(t_2), \ldots, \mathcal{I}_\rho(t_n)) \in p^I$
$\mathcal{I} \models \neg\varphi$	sse	$\mathcal{I} \models \varphi$ não se verifica
$\mathcal{I} \models \varphi \wedge \psi$	sse	$\mathcal{I} \models \varphi$ e $\mathcal{I} \models \psi$
$\mathcal{I} \models \varphi \vee \psi$	sse	$\mathcal{I} \models \varphi$ ou $\mathcal{I} \models \psi$
$\mathcal{I} \models \varphi \rightarrow \psi$	sse	$\mathcal{I} \models \psi$ sempre que $\mathcal{I} \models \varphi$
$\mathcal{I} \models \varphi \leftrightarrow \psi$	sse	$\mathcal{I} \models \psi$ sempre que $\mathcal{I} \models \varphi$
		e $\mathcal{I} \models \varphi$ sempre que $\mathcal{I} \models \psi$
$\mathcal{I} \models \forall x\varphi$	sse	para todo $a \in \mathcal{U}$, $\mathcal{I}[x \rightarrow a] \models \varphi$ se verifica
$\mathcal{I} \models \exists x\varphi$	sse	para algum $a \in \mathcal{U}$, $\mathcal{I}[x \rightarrow a] \models \varphi$ se verifica

Se $\mathcal{I} = (I^{\mathcal{U}}, \rho)$, então $\mathcal{I}[x \rightarrow a]$ é definido como $(I^{\mathcal{U}}, \rho[x \rightarrow a])$.

Exemplo 4. Seja a seguinte linguagem de 1^a ordem:

Predicados: $q(x, y)$ e $r(x, y)$
Constantes: c

e seja a seguinte interpretação I para a linguagem acima:

1. O universo $\mathcal{U}$ é o conjunto de números naturais;
2. $r^I(x, y)$ é a relação "$x = y$" sobre $\mathcal{U}$ e $q^I(x, y)$ é a relação "$x < y$" sobre $\mathcal{U}$;
3. c^I é 0.

A fórmula $\forall y(q(c, y) \vee r(c, y))$, que pode ser traduzida em linguagem natural para "o número zero é menor ou igual a qualquer número natural", é satisfeita sob $\mathcal{I} = (I^{\mathcal{U}}, \rho)$, uma vez que para todo número natural $n \in \mathcal{U}$, tem-se que

$$(I^{\mathcal{U}}, \rho[y \rightarrow n]) \models q(c, y) \vee r(c, y).$$

Capítulo 3

Um Sistema Formal Baseado em Árvores de Prova

Neste capítulo é apresentado um sistema formal que fornece suporte a procedimentos de prova baseados em árvores de prova (APs), o qual consiste em um sistema completo para a lógica de 1^a ordem. Esse sistema formal foi proposto por Vieira (1987, 1996).

O sistema formal apresentado realiza as suas provas por refutação. Em uma prova por refutação, se Γ é o conjunto de fórmulas correspondentes aos axiomas do sistema e α é a fórmula que deseja-se provar verdadeira, prova-se que $\Gamma \models \alpha$, mostrando que não existe uma interpretação que satisfaça simultaneamente todas as fórmulas de $\Gamma \cup \{\neg\alpha\}$, ou seja, esse conjunto é insatisfatível. Como o sistema formal assume o conhecimento expresso na forma de cláusulas, o problema consiste em demonstrar a insatisfabilidade do conjunto de cláusulas correspondentes a $\Gamma \cup \{\neg\alpha\}$.

Como se sabe, para cada fórmula bem formada, é possível determinar um conjunto de cláusulas correspondentes. Uma cláusula é uma disjunção de literais; um literal é uma fórmula atômica ou a negação de uma. Será utilizada a notação $|L|$ para denotar o átomo do literal L. Dois literais $|L_1|$ e $|L_2|$ são ditos complementares se, e somente se, têm sinais trocados e $|L_1| = |L_2|$. Designaremos o literal complementar a um literal L por $\overline{L}$.

O problema de demonstrar que um conjunto de cláusulas é insatisfatível equivale a mostrar que não existe uma interpretação que satisfaça todas as fórmulas do conjunto. Em lógica de predicados, teria-se que considerar todas as possíveis interpretações em todos os universos de discurso, o que seria, em muitos casos, impossível. Herbrand (1967) mostrou que o problema de provar a insatisfabilidade de um conjunto de cláusulas pode ser reduzido ao problema de encontrar um conjunto de instâncias das

cláusulas que seja insatisfatível. O teorema de Herbrand é enunciado assim:

> "*Um conjunto de cláusulas* Γ *é insatisfatível se, e somente se, existe um conjunto finito de instâncias de cláusulas de* Γ *que é insatisfatível.*"

O mecanismo de prova subjacente ao sistema formal aqui apresentado baseia-se no teorema de Herbrand e trabalha mostrando a insatisfabilidade de um conjunto de cláusulas.

3.1 Árvores de Prova

Nesta seção será apresentado sucintamente o conceito de árvores de prova, utilizando como auxílio a árvore ilustrada na Figura 3.1.

Uma árvore de prova consiste em uma forma de representação das deduções realizadas pelo sistema de inferência. Em uma árvore de prova, os nós da árvore são rotulados por literais, os quais podem ser divididos em dois grupos: literais eleitos e literais candidatos. Os literais candidatos correspondem àqueles literais da árvore que não foram utilizados em nenhuma regra de inferência. Já os literais eleitos podem ainda ser divididos em dois grupos: literais expandidos e literais reduzidos, de acordo com a regra de inferência aplicada. Na Figura 3.1, os literais expandidos são mostrados dentro de retângulos e os literais reduzidos dentro de hexágonos. Os literais restantes (apenas um) são os literais candidatos.

As árvores de prova comumente utilizadas em sistemas de inferência subjacentes a linguagem de programação Prolog apresentam apenas literais expandidos (Vieira, 1987). No entanto, na busca pela completude, é necessário incluir a regra de redução, cuja aplicação produz os literais reduzidos. Para cada literal reduzido L, existe um literal complementar $\overline{L}$ ancestral de L, denominado *literal redutor*.

Em uma árvore de prova, se L_0 é um literal expandido em um nó interno da árvore e $L_1, L_2, \ldots, L_n$ são seus filhos, então segue-se que $\overline{L_1} \vee \overline{L_2} \vee \ldots \vee \overline{L_n} \vee L_0$ é instância de uma cláusula do conjunto de partida a partir do qual se obteve a árvore de prova. Na árvore de prova ilustrada na Figura 3.1, cada número dentro de um círculo indica a instância da cláusula utilizada na expansão do literal. Assim, por exemplo,

$$\neg C \vee \neg D \vee A$$

é instância da cláusula 2 do conjunto de partida. Essa mesma instância pode ser vista da seguinte forma: $C \wedge D \rightarrow A$. Dessa forma, uma árvore de prova pode ser vista como

uma codificação de uma estrututa de implicação.

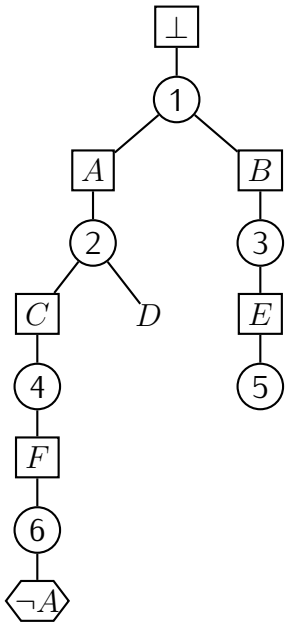


Figura 3.1: Árvore de Prova

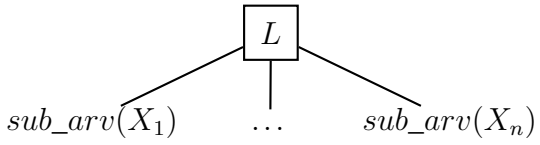
3.2 O Sistema Formal

O sistema formal tem 3 regras de inferência: as regras de codificação, expansão e redução. E dois tipos de linguagens: a linguagem de cláusulas, para expressar axiomas *extra-lógicos*, que são fórmulas que descrevem o que é assumido como verdade dentro da aplicação, e a linguagem de *cláusulas estruturadas*, para expressar fórmulas deduzidas. Estas últimas consistem simplesmente de árvores de prova *linearizadas*.

Uma *cláusula estruturada* (CE) é uma expressão da forma $L\alpha$, em que L é um literal e α é um conjunto (que pode ser vazio) de literais e/ou cláusulas estruturadas. Os literais M de todas as subcláusulas estruturadas $M\beta$ que ocorrem em uma cláusula estruturada $L\alpha$ são denominados *literais expandidos*; os literais restantes de $L\alpha$ são divididos em duas classes: os denominados *literais candidatos* e os *literais reduzidos*.

As classes dos literais expandidos e dos literais reduzidos compreendem aqueles literais que foram utilizados em alguma regra de inferência: na *regra de expansão* ou na *regra de redução*, ambas as regras descritas na próxima seção. Já a classe dos literais candidatos compreende aqueles literais que não foram utilizados ainda em alguma regra de inferência, ou seja, os literais são candidatos a uma futura aplicação de alguma regra de inferência.

A árvore de prova representada pela cláusula estruturada $X = L\{X_1, ..., X_n\}$, $arv(X)$, é:

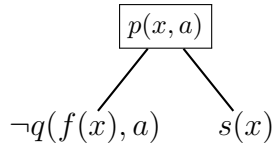


onde $sub\_arv(X_i) = \begin{cases} arv(X_i) & \text{caso } X_i \text{ seja uma cláusula estruturada;} \\ \langle X_i \rangle & \text{caso } X_i \text{ seja um literal reduzido;} \\ X_i & \text{caso } X_i \text{ seja um literal candidato.} \end{cases}$

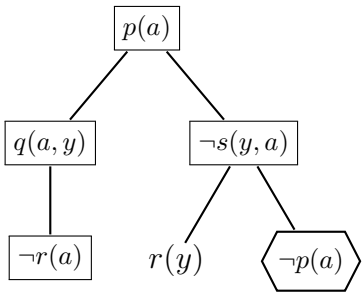
Seguem exemplos de cláusulas estruturadas e árvores de prova correspondentes:

1.
   - CE: $p(x)\{\}$
   - Árvore de prova:

2.
   - CE: $p(x, a)\{\neg q(f(x), a), s(x)\}$
   - Árvore de prova:



3.
   - CE: $p(a)\{q(a, y)\{\neg r(a)\{\}\}, \neg s(y, a)\{r(y), \langle \neg p(a) \rangle\}\}$
   - Árvore de prova:



Os literais dentro de retângulos são os literais expandidos, os literais dentro de hexágonos são os literais reduzidos e os literais restantes são os literais candidatos.

3.2.1 Regras de Inferência

As três regras de inferência que compõem o sistema formal são as regras de codificação, expansão e redução. Para apresentar as três regras, é necessário primeiramente introduzir a noção de substituição.

Uma *substituição* é um conjunto $\sigma = \{v_1/t_1, v_2/t_2, \ldots, v_n/t_n\}$, em que, para $1 \leqslant i \leqslant n$, v_i é variável, t_i é um termo e $v_i \neq v_j$ para $i \neq j$. Se F é uma fórmula, então a *instanciação* de F por σ, denotada por $F\sigma$, é a fórmula obtida substituindo-se simultaneamente cada v_i por t_i na fórmula F. Duas fórmulas F_1 e F_2 são ditas *unificáveis* se existe uma substituição σ tal que $F_1\sigma = F_2\sigma$. Assim, uma substituição unificadora para $\{p(a, y, z), p(x, y, c)\}$ é $\{x/a, y/b, z/c\}$, resultando na fórmula $p(a, b, c)$. A substituição σ é denominada o *unificador mais geral* (UMG) de duas fórmulas F_1 e F_2 se $F_1\sigma = F_2\sigma$ e, para qualquer outro unificador θ de F_1 e F_2, $F_1\theta = F_2\theta$ é uma instância de $F_1\sigma = F_2\sigma$.

3.2.1.1 Regras de Inferência

A *operação de codificação* de uma cláusula da base de conhecimentos consiste no primeiro passo de uma dedução. Usualmente, a cláusula codificada consiste em uma das cláusulas da negação da consulta.

Definição 1. Regra de Codificação. Seja C uma cláusula qualquer e sejam $L_1, L_2, \ldots, L_n$ seus literais ($n > 0$). Então, o resultado da aplicação da regra de codificação, usando C como premissa, é a cláusula estruturada $\bot\{\overline{L_1}, \overline{L_2}, \ldots, \overline{L_n}\}$. Isto é justificável pelo fato de que a cláusula C é logicamente equivalente a $(\overline{L_1} \wedge \overline{L_2} \wedge \ldots \wedge \overline{L_n}) \rightarrow \bot$. Com isto, toda árvore de prova terá como raiz o literal $\bot$.

A *operação de expansão*, a ser apresentada a seguir, tem como objetivo expandir um literal candidato L_1, de uma árvore de prova, via uma cláusula da base de conhecimentos que contenha um literal L_2 que unifica com L_1.

Definição 2. Regra de Expansão. Seja uma cláusula estruturada $C_1 = \bot\{\ldots L_1 \ldots\}$ e uma cláusula $C_2 = M_1 \vee \ldots \vee M_n \vee L_2$, sendo L_1 um literal candidato tal que exista um UMG σ entre L_1 e L_2. Uma expansão de C_1 via C_2 resulta na cláusula estruturada $\bot\{\ldots L_1\{\overline{M_1}, \overline{M_2}, \ldots, \overline{M_n}\} \ldots\}\sigma$.

Como exemplo, dada a cláusula estruturada

$$X = \bot\{\neg p(x, y)\{\neg s(x), r(x, y)\}, s(y)\}$$

e uma cláusula

$$C = r(a, y) \vee \neg p(a, b) \vee q(y)$$

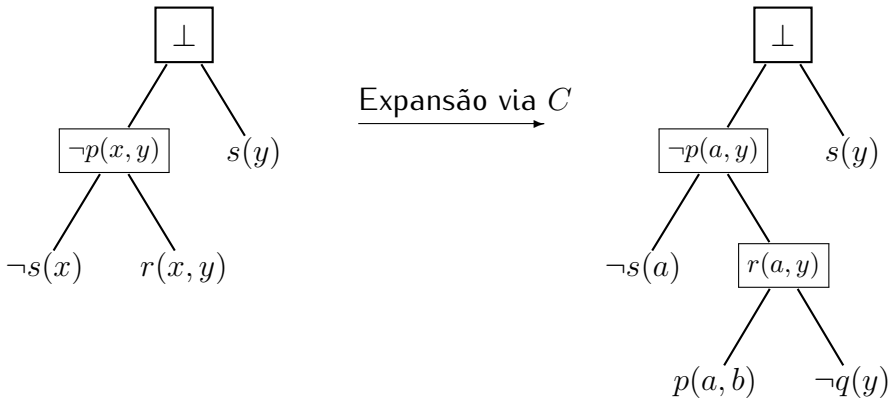
que é equivalente a

$$p(a, b) \wedge \neg q(y) \rightarrow r(a, y),$$

uma expansão de X pela cláusula C é

$$Y = \bot\{\neg p(a, y)\{\neg s(a), r(a, y)\{p(a, b), \neg q(y)\}\}, s(y)\}.$$

Graficamente tem-se:



A *operação de redução*, a ser apresentada a seguir, permite que um literal candidato da árvore de prova torne-se um literal reduzido (isto é, não será necessário expandí-lo) se este se unifica com o complemento de um literal expandido que é seu ancestral na árvore de prova. Esta operação implementa uma forma de raciocínio por contradição, do tipo: se $(\neg P \wedge Q) \rightarrow P$, então pode-se concluir que $Q \rightarrow P$ considerando-se dois casos:

- P é verdadeiro e, portanto, $Q \rightarrow P$;
- $\neg P$ é verdadeiro e, como $(\neg P \wedge Q) \rightarrow P$, conclui-se que $Q \rightarrow P$.

Uma outra maneira de se mostrar que as duas fórmulas são equivalentes é a seguinte:

$$\begin{aligned}(\neg P \wedge Q) \rightarrow P &\equiv \neg(\neg P \wedge Q) \vee P \\ &\equiv (P \vee \neg Q) \vee P \\ &\equiv P \vee \neg Q\end{aligned}$$

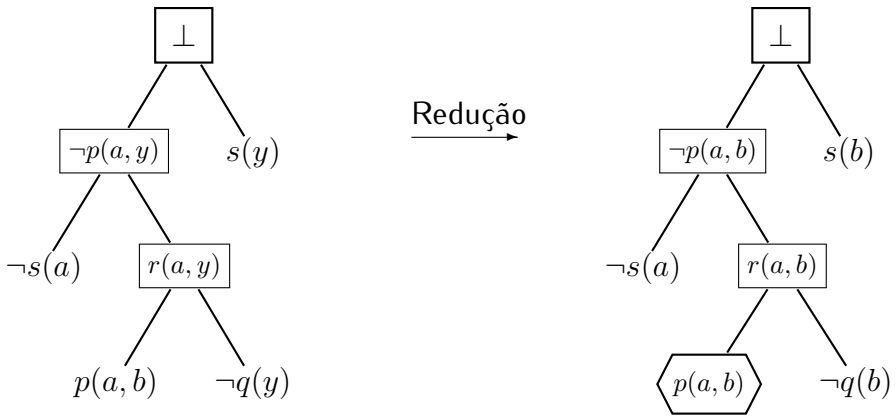
$$\equiv Q \rightarrow P.$$

Definição 3. Regra de Redução.
Uma redução de uma cláusula estruturada $\perp\{\ldots L_1\{\ldots L_2 \ldots\}\ldots\}$ onde L_1 é um literal expandido ancestral de um literal candidato de sinal contrário L_2, tal que $|L_1|$ e $|L_2|$ têm um UMG σ, é a cláusula estruturada $\perp\{\ldots L_1\{\ldots \langle L_2\rangle \ldots\}\ldots\}\sigma$.

O resultado da aplicação da regra de redução à cláusula estruturada Y apresentada anteriormente é:

$$\perp\{\neg p(a,b)\{\neg s(a), r(a,b)\{\langle p(a,b)\rangle, \neg q(b)\}\}, s(b)\}$$

Graficamente:



3.2.2 Dedução

Uma *dedução* de uma cláusula estruturada D, a partir de um conjunto de cláusulas Γ, é uma sequência finita de cláusulas estruturadas $D_1, D_2, \ldots, D_n$ tal que:

1. $D_n = D$;
2. a) D_1 é o resultado da aplicação da regra de codificação a uma das cláusulas de Γ;
   b) D_i, para todo $i > 1$, é o resultado da aplicação da regra de expansão, via uma das cláusulas do conjunto Γ, ou de redução, à cláusula estruturada D_{i-1}.

3.2.3 Refutação

Uma *refutação* nesse sistema formal consiste em uma dedução de uma cláusula estruturada livre de literais candidatos. Como exemplo, seja o conjunto de cláusulas:

1. $p(x) \vee \neg s(x) \vee r(x)$
2. $\neg r(x) \vee q(x)$
3. $s(a) \vee r(a)$
4. $\neg p(a)$

e a negação da consulta $\exists y\ q(y)$:

5. $\neg q(y)$

Uma possível refutação é apresentada abaixo:

6. $\bot\{q(y)\}$ (cod 5)
7. $\bot\{q(y)\{r(y)\}\}$ (exp 2)
8. $\bot\{q(y)\{r(y)\{\neg p(y), s(y)\}\}\}$ (exp 1)
9. $\bot\{q(a)\{r(a)\{\neg p(a)\{\}, s(a)\}\}\}$ (exp 4)
10. $\bot\{q(a)\{r(a)\{\neg p(a)\{\}, s(a)\{\neg r(a)\}\}\}\}$ (exp 3)
11. $\bot\{q(a)\{r(a)\{\neg p(a)\{\}, s(a)\{\langle\neg r(a)\rangle\}\}\}\}$ (red)

A árvore de prova correspondente à CE 11 está ilustrada na Figura 3.2. Essa árvore de prova não possui literais candidatos, ou seja, é uma árvore de prova fechada.

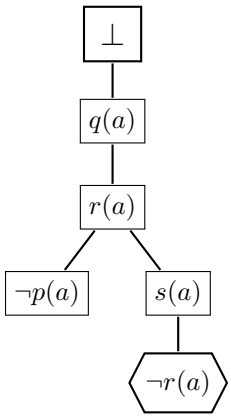


Figura 3.2: Árvore de prova sem literais candidatos.

3.3 Extração de Respostas

Estando a dedução representada na forma de uma árvore de prova, então podemos extrair a partir da mesma a resposta para a consulta se soubermos quais instâncias de cláusulas da negação da consulta figuram na árvore de prova.

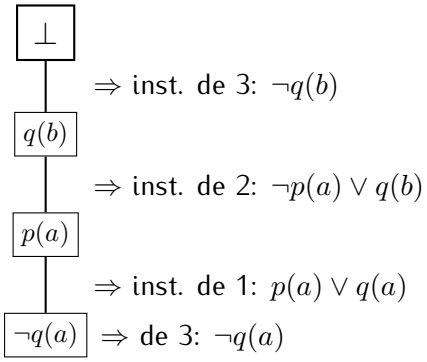
Como exemplo, seja o conjunto de cláusulas:

1. $p(x) \vee q(x)$
2. $\neg p(a) \vee q(b)$

Seja o problema de determinar y tal que $q(y)$:

3. $\neg q(y)$

Mostra-se abaixo uma árvore de prova, com uma indicação das instâncias utilizadas:



A resposta é obtida das instâncias de 3: $q(a) \vee q(b)$.

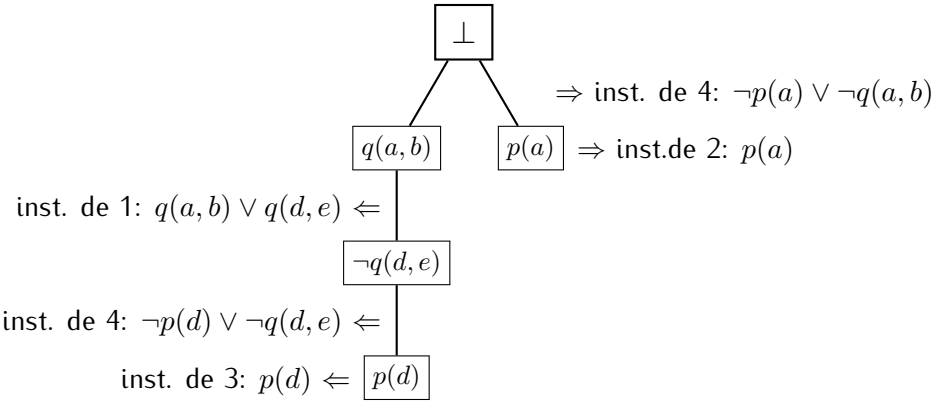
Em um outro exemplo, considera-se o seguinte conjunto de cláusulas:

1. $q(a, b) \vee q(d, e)$
2. $p(a)$
3. $p(d)$

Seja o problema de determinar x, y tal que $p(x) \wedge q(x, y)$:

4. $\neg p(x) \vee \neg q(x, y)$

Mostra-se abaixo uma árvore de prova, com uma indicação das instâncias utilizadas:



A resposta é obtida das instâncias de 4: $(p(a) \wedge q(a,b)) \vee (p(d) \wedge q(d,e))$.

Foram apresentados exemplos de como pode ser feita a extração de uma resposta para a consulta a partir de uma árvore de prova fechada. No capítulo seguinte, vamos estender a noção de respostas, permitindo-se que respostas para a consulta sejam também obtidas a partir de árvores de prova não fechadas. A definição de respostas apresentada no capítulo seguinte e o algoritmo proposto no capítulo 5 exigem, no entanto, que a consulta seja representada por um único literal. Com este intuito, transformamos sempre o problema em outro equivalente no qual a consulta é constituída por um único literal, da forma descrita no parágrafo abaixo.

Se a consulta é da forma $\exists X P(X)$, sendo P um conjunto de predicados e X uma n-upla de variáveis existencialmente quantificadas da fórmula $P(X)$, então substitui-se a fórmula da consulta pela fórmula $\forall X(P(X) \rightarrow resp(X))$, em que $resp$ é um símbolo predicativo novo. Esta fórmula é convertida para o formato clausal e adicionada ao conjunto de cláusulas da base de conhecimentos. A consulta passa então a ser representada por $\exists X\ resp(X)$. O literal que possui o predicado $resp$ será denominado um *literal de resposta* (Green, 1969).

No exemplo anterior, a fórmula da consulta é: $(\exists x, y)(p(x) \wedge q(x,y))$. Esta fórmula é substituída por $(\forall x, y)((p(x) \wedge q(x,y)) \rightarrow resp(x,y))$, cuja cláusula correspondente é: $\neg p(x) \vee \neg q(x,y) \vee resp(x,y)$. Assim, o conjunto de cláusulas da base de conhecimentos passa a ser:

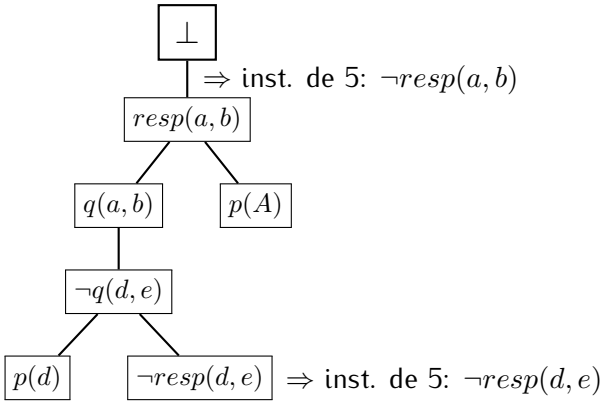
1. $q(a,b) \vee q(d,e)$
2. $p(a)$
3. $p(d)$

4. $\neg p(x) \vee \neg q(x,y) \vee resp(x,y)$

e a cláusula da consulta é:

5. $\neg resp(x,y)$

Mostra-se abaixo uma árvore de prova, com uma indicação das instâncias da cláusula $\neg resp(x,y)$:



A resposta é obtida das instâncias da cláusula $\neg resp(x,y)$: $resp(a,b) \vee resp(d,e)$, o que dá como realização dos quantificadores existenciais, $\{x/a\}$ e $\{y/b\}$ ou $\{x/d\}$ e $\{y/e\}$, e como resposta: $(p(a) \wedge q(a,b)) \vee (p(d) \wedge q(d,e))$.

Para facilitar algumas definições realizadas nos capítulos seguintes, definimos a função $\mathcal{R}(resp(a_1,\ldots,a_n))$, a qual dá como resultado

$$P(x_1,\ldots,x_n)\{x_1/a_1,\ldots,x_n/a_n\},$$

onde $P(x_1,\ldots,x_n)$ é a fórmula da consulta, $x_1,\ldots,x_n$ é uma n-upla de variáveis existencialmente quantificadas da sentença $P(x_1,\ldots,x_n)$ e $a_1,\ldots,a_n$ é uma n-upla de variáveis e/ou termos. Assim, para o exemplo anterior, $\mathcal{R}(resp(a,b)) = p(a) \wedge q(a,b)$.

Capítulo 4

Definição de Respostas para Sistemas Baseados em APs

4.1 Introdução

Tradicionalmente, os provadores de teoremas eram utilizados para responder a consultas do tipo "é uma proposição consistente com uma dada base de conhecimentos?". Consultas dessa forma consistem em requisições ao sistema para verificar se uma fórmula é ou não uma conseqüência lógica da base de conhecimentos e requerem respostas simples, como "sim", "não", "não sei". O uso de um provador de teoremas para fornecer respostas a essas consultas consiste em seu uso habitual para provar teoremas.

Existe uma variedade de consultas feitas em nossos dias que requerem respostas mais sofisticadas do que uma resposta "sim" ou "não". Essas consultas exigem do sistema a capacidade de extrair informações da base de conhecimentos, a fim de que essas possam ser retornadas aos usuários. Uma consulta que pergunta por itens específicos de informação pode usualmente ser expressa na forma, "Determinar X tal que $P(X)$ é verdadeiro", onde, no caso geral, X é um conjunto de variáveis e P são predicados sobre X. Na prova de teoremas, consultas dessa forma envolvem verificar se a fórmula $\exists X P(X)$ é um teorema, mas, além disso, o sistema deve ainda determinar instâncias específicas para as variáveis existencialmente quantificadas da fórmula.

Green (1969) mostra em seu trabalho que se existe uma prova de que a fórmula $\exists X P(X)$ segue de um conjunto de axiomas, então, a partir do método de prova por resolução, pode-se obter ou construir um X tal que $P(X)$. Para isto, Green adiciona um literal de resposta à conjectura a ser verificada. O papel do literal de resposta consiste em coletar as substituições feitas pelo procedimento de resolução em $P(X)$. Quando

uma refutação é encontrada, os argumentos presentes nos literais de resposta correspondem a objetos no domínio que satisfazem a propriedade P. Respostas dessa forma, que mencionam objetos que satisfazem uma propriedade, são denominadas *respostas extensionais*.

A definição de respostas de Gallaire et al. (1984) em bases de dados dedutivas (BDD) é feita considerando-se apenas respostas extensionais. Por bases de dados dedutivas diz-se uma base de dados na qual existem fatos explícitos, constituindo a base de dados extensional (BDE), e regras gerais que permitem a dedução de fatos não-explícitos, constituindo a base de dados intensional (BDI). Os autores definem que uma resposta para uma consulta $P(X)$, em que X é uma n-upla de variáveis, é o conjunto

$$\{A : A \text{ é uma } n\text{-tupla de termos e BDE} \cup \text{BDI} \vdash P(A)\}.$$

No curso de obtenção de uma prova para a conjectura associada à consulta, muitas informações relevantes geradas para a consulta são descartadas, muitas das quais poderiam ser utilizadas para se responder a consulta do usuário. A partir do conhecimento intensional existente pode-se extrair condições ou características que permitam descrever de uma maneira abstrata um conjunto de objetos que constituem uma resposta extensional. Respostas especificando as condições para um elemento ser uma resposta para a consulta são conhecidas na literatura como *respostas intensionais*.

Cholvy e Demolombe (1986) definem uma resposta intensional para uma consulta $P(X)$ como o conjunto de fórmulas $ans(X)$ tal que $\forall X(ans(X) \rightarrow P(X))$. A fórmula $\forall X(ans(X) \rightarrow P(X))$ define as condições $ans(X)$ que objetos ou tuplas devem satisfazer para ser uma resposta para a consulta $P(X)$.

Alguns autores ainda consideram a classe de *respostas hipotéticas ou condicionais*. Estas respostas possuem associadas uma condição que deve ser satisfeita para a resposta ser considerada correta. Demolombe (1992) apresenta uma definição de respostas condicionais em bases de dados lógicas. Se a consulta é uma fórmula geral $F(x)$, então cria-se um novo símbolo predicativo $q(x)$ e adiciona-se a fórmula $Q = \forall x(F(x) \rightarrow q(x))$ à BDD. A consulta passa então a ser representada pelo literal positivo $q(x)$. Sendo a BDE um conjunto de fórmulas fechadas e a BDI um conjunto de fórmulas, o autor define a teoria $\mathrm{T} = Q \cup \text{BDE} \cup \text{BDI}$, onde todas as fórmulas estão representadas no formato clausal. A resposta para a consulta $q(x)$ é o conjunto de fórmulas:

$$\{q\sigma \vee c \mid \mathrm{T} \vdash q\sigma \vee c \text{ e não existe nenhuma cláusula que subjuga } q\sigma \vee c\},$$

sendo que uma cláusula C_1 *subjuga* uma cláusula C_2 se, e somente se, existe uma substituição θ tal que $C_1\theta \subseteq C_2$. Na presença de C_1, C_2 é redundante, e pode ser desconsiderada.

De acordo com essa definição de respostas, qualquer resposta na forma de regra é uma resposta condicional. Em um trabalho posterior, Demolombe (1997) restringe a sua definição de respostas condicionais, proibindo a presença de variáveis na resposta. Por não poder constar variáveis na resposta, essa definição exclui de seu escopo respostas intensionais. Segundo essa nova definição, uma resposta condicional para a consulta $q(x)$ é um conjunto de fórmulas fechadas $cond(a, b) \rightarrow q(a)$, em que $cond(a, b)$ representa a informação adicional necessária para se derivar a resposta $q(a)$. Uma definição geral de respostas condicionais por esse novo trabalho de Demolombe é:

$$\{cond(a, b) \rightarrow q(a) \mid \mathsf{BD} \vdash cond(a, b) \rightarrow q(a)\}.$$

Este capítulo apresenta uma definição formal e completa de respostas para consultas existencialmente quantificadas no contexto de um sistema de prova baseado em árvores de prova. Além disso, as respostas obtidas segundo a definição dada são particionadas em três classes de respostas: extensionais, intensionais e hipotéticas. A definição de respostas aqui apresentada segue do trabalho de Burhans (2002), desenvolvido no contexto de um provador de teoremas por resolução, e difere das definições de respostas usuais, as quais vinculam a obtenção de uma resposta à existência de uma prova. Considera-se que mesmo árvores de prova não-fechadas, geradas em passos intermediários da dedução, produzem respostas em potencial para a consulta.

4.2 Uma Definição de Respostas para Sistemas Baseados em Árvores de Prova

Considere uma base de conhecimentos BC na qual o conhecimento é expresso mediante um conjunto de fórmulas da lógica de predicados e uma consulta da forma $\exists X P(X)$, em que X é é uma n-upla de variáveis existencialmente quantificadas da sentença $P(X)$. Tendo a consulta a forma geral $P(X)$, inicialmente adiciona-se a fórmula $Q = \forall X(P(X) \rightarrow resp(X))$ à base de conhecimentos BC, onde $resp$ é um símbolo predicativo novo, e a consulta passa a ser representada por $\exists X\ resp(X)$.

Seja CQ o conjunto de cláusulas correspondentes à fórmula Q e seja o conjunto

$BC' = BC \cup CQ$, no qual todas as fórmulas estão representadas no formato clausal. O mecanismo de prova baseado em árvores de prova tenta refutar a consulta $\exists x\ resp(X)$, gerando uma árvore de prova sem literais candidatos a partir de $BC' \cup \{\neg resp(X)\}$. Seja, então, uma árvore de prova $\mathcal{A}_i$ qualquer gerada em um passo i da refutação. Se $l_1, \ldots, l_n$ são os literais candidatos em $\mathcal{A}_i$ e $\neg r_1, \ldots, \neg r_m$ são as instâncias da cláusula $\neg resp(X)$ que figuram em $\mathcal{A}_i$, então:

$$BC' \cup \{l_1 \wedge \ldots \wedge l_n\} \models r_1 \vee \ldots \vee r_m,$$

logo

$$BC' \models l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m.$$

Definimos que a implicação

$$l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$$

é uma *resposta* para a consulta $resp(X)$ se, e somente se, em seu conseqüente figura pelo menos um literal de resposta (ou seja, $m \geqslant 1$) e $\overline{l_1} \vee \ldots \vee \overline{l_n} \vee r_1 \vee \ldots \vee r_m$ não é uma cláusula de CQ (cláusula da consulta) ou uma tautologia.

Observe que no conseqüente da implicação $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$ figura uma disjunção formada apenas por literais de resposta. Queremos, no entanto, que literais de resposta figurem apenas no conseqüente da implicação. Em um único momento da refutação tem-se uma árvore de prova que possui como literal candidato um literal de resposta na forma positiva: na árvore de prova gerada a partir da codificação da cláusula $\neg resp(X)$. No entanto, essa árvore de prova não está associada a uma resposta para a consulta. Como os literais de resposta são anexados às cláusulas da consulta na sua forma positiva, em qualquer outra árvore de prova, se existe um literal de resposta que é literal candidato, então o mesmo é da forma $\neg resp(A)$, em que A é uma n-upla de termos. E, uma vez que a cláusula $\neg resp(X)$ faz parte de todo conjunto de partida, supõe-se todo literal candidato $\neg resp(A)$ como um literal expandido pela instância da cláusula $\neg resp(X)$ obtida pela substituição $\{X/A\}$.

4.3 Respostas Extensionais

Respostas extensionais mencionam objetos que satisfazem a consulta. Na prova de teoremas, uma resposta extensional é uma "testemunha" que prova a veracidade de uma

fórmula existencial: é a substituição das variáveis existencialmente quantificadas da conjectura associada à consulta por termos fechados que provam a sua veracidade.

Assim, a obtenção de uma resposta extensional está associada à existência de uma prova para a conjectura associada à consulta. Em sistemas de prova baseados em árvores de prova, a existência de uma prova é denotada pela dedução de uma árvore de prova sem literais candidatos (fechada) a partir do conjunto formado pelas cláusulas da base de conhecimentos e da negação da consulta. Como a árvore de prova não possui literais candidatos, $n = 0$, logo a resposta extensional é uma cláusula formada apenas por literais de resposta.

Definição 4. Resposta Extensional. A implicação $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$ corresponde a uma *resposta extensional* se, e somente se, $n = 0$ e todos os literais na cláusula $r_1 \vee \ldots \vee r_m$ são fechados. A resposta extensional para a consulta é a disjunção $\mathcal{R}(r_1) \vee \ldots \vee \mathcal{R}(r_m)$, onde a operação $\mathcal{R}$ foi definida no capítulo 3.

A Tabela 4.1 mostra em suas seis primeiras linhas as cláusulas de uma base de conhecimentos BC adaptada de (Rich e Knight, 1990). Em relação a essa base, é feita a seguinte questão ou consulta: "Do que Tom se alimenta?". A cláusula correspondente a essa consulta é apresentada na sétima linha da tabela. Esse mesmo exemplo é utilizado em (Burhans, 2002) e foi escolhido aqui exatamente com o intuito de possibilitar ao leitor a comparação entre uma mesma abordagem aplicada a dois métodos de prova distintos: resolução e árvores de prova.

	Descrição	Cláusula
1	gatos se alimentam de peixes	$\neg gato(x) \vee \neg peixe(y) \vee alimenta(x, y)$
2	siameses são gatos	$\neg siames(x) \vee gato(x)$
3	sardinhas são peixes	$\neg sardinha(x) \vee peixe(x)$
4	Nina é uma sardinha	$sardinha(Nina)$
5	Tina é uma sardinha	$sardinha(Tina)$
6	Tom é um siamês	$siames(Tom)$
7	Do que Tom se alimenta?	$\neg alimenta(Tom, x) \vee resp(x)$

Tabela 4.1: Base de Conhecimentos

Dadas as sete cláusulas apresentadas na Tabela 4.1 e a cláusula da negação da consulta, $\neg resp(x)$, mostramos a seguir duas refutações ($\mathcal{A}_0$-$\mathcal{A}_{6.1}$ e $\mathcal{A}_0$-$\mathcal{A}_{6.2}$) no sistema formal considerado. Para toda árvore de prova $\mathcal{A}_i$ deduzida, mostramos a implicação $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$ que a mesma gera.

$\mathcal{A}_0 : \bot\{resp(x)\}$
$resp(x) \rightarrow resp(x) \equiv \neg resp(x) \vee resp(x)$ é uma tautologia

$\mathcal{A}_1 : \bot\{resp(x)\{alimenta(Tom, x)\}\}$
$alimenta(Tom, x) \rightarrow resp(x) \equiv \neg alimenta(Tom, x) \vee resp(x) \in CQ$

$\mathcal{A}_2 : \bot\{resp(x)\{alimenta(Tom, x)\{gato(Tom), peixe(x)\}\}\}$
$gato(Tom) \wedge peixe(x) \rightarrow resp(x)$ (resposta)

$\mathcal{A}_3 : \bot\{resp(x)\{alimenta(Tom, x)\{gato(Tom)\{siames(Tom)\}, peixe(x)\}\}\}$
$siames(Tom) \wedge peixe(x) \rightarrow resp(x)$ (resposta)

$\mathcal{A}_4 : \bot\{resp(x)\{alimenta(Tom, x)\{gato(Tom)\{siames(Tom)\{\}\}, peixe(x)\}\}\}$
$peixe(x) \rightarrow resp(x)$ (resposta)

$\mathcal{A}_5 : \bot\{resp(x)\{alimenta(Tom, x)\{gato(Tom)\{siames(Tom)\{\}\}, peixe(x)\{sardinha(x)\}\}\}\}$
$sardinha(x) \rightarrow resp(x)$ (resposta)

$\mathcal{A}_{6.1} : \bot\{resp(Nina)\{alimenta(Tom, Nina)\{gato(Tom)\{siames(Tom)\{\}\},$
$peixe(Nina)\{sardinha(Nina)\{\}\}\}\}\}$
$resp(Nina)$ (resposta)

$\mathcal{A}_{6.2} : \bot\{resp(Tina)\{alimenta(Tom, Tina)\{gato(Tom)\{siames(Tom)\{\}\},$
$peixe(Tina)\{sardinha(Tina)\{\}\}\}\}\}$
$resp(Tina)$ (resposta)

Nesse exemplo, as duas respostas extensionais geradas são:

$$alimenta(Tom, Nina)$$
"Tom se alimenta de Nina"

$$alimenta(Tom, Tina)$$
"Tom se alimenta de Tina"

4.4 Respostas Intensionais

Respostas intensionais consistem em descrições dos objetos que satisfazem a consulta. Essas descrições definem as condições necessárias para que um objeto seja uma resposta para a consulta e são usualmente capturadas na forma de regras. A definição de respostas intensionais apresentada neste trabalho segue da definição de Burhans (2002),

a qual baseia-se no conceito de variáveis compartilhadas. Inicialmente, definimos o conjunto de variáveis compartilhadas de um literal.

Definição 5. Conjunto de Variáveis Compartilhadas de um Literal. O conjunto de variáveis compartilhadas de um literal l em relação a um conjunto de literais C, é o conjunto de literais $\mathcal{VS}(l)$ formados pelos literais $C - \{l\}$ que compartilham pelo menos uma variável com l.

Como exemplo, dado $C = \{R(x,z), P(x), Q(z)\}$, tem-se $\mathcal{VS}(P(x)) = \{R(x,z)\}$ e $\mathcal{VS}(R(x,z)) = \{P(x), Q(z)\}$.

Definição 6. Conjunto de Variáveis Compartilhadas de um Conjunto de Literais. O conjunto de variáveis compartilhadas de um conjunto finito de literais $S = \{l_1, l_2, \ldots, l_n\}$ é a união dos conjuntos de variáveis compartilhadas para cada um dos literais:
$\mathcal{VS}(S) = \bigcup \mathcal{VS}(l_i)$.

Como exemplo, dado $C = \{R(x,z), P(x), Q(z)\}$,
$\mathcal{VS}(\{P(x), R(x,z)\}) = \{R(x,z)\} \cup \{P(x), Q(z)\} = \{R(x,z), P(x), Q(z)\}$.

Definição 7. Fecho das Variáveis Compartilhadas de um Conjunto de Literais. O fecho das variáveis compartilhadas de um conjunto de literais $\mathcal{L}$, denotado por $\mathcal{CVS}(\mathcal{L})$, é o ponto fixo do operador $\mathcal{VS}$: $\mathcal{VS}(\mathcal{CVS}(\mathcal{L})) = \mathcal{CVS}(\mathcal{L})$. É definido recursivamente como segue:

$$\begin{aligned}
&\mathcal{CVS}_0 = \mathcal{VS}(\mathcal{L}) \\
&\mathcal{CVS}_1 = \mathcal{VS}(\mathcal{CVS}_0) \\
&\vdots \\
&\mathcal{CVS}_n = \mathcal{VS}(\mathcal{CVS}_{n-1}) \\
&\mathcal{CVS}_{n+1} = \mathcal{CVS}_n
\end{aligned}$$

Como exemplo, dado $C = \{R(x,z), P(x), Q(z)\}$, para $\mathcal{L} = \{R(x,z)\}$:

$$\begin{aligned}
\mathcal{CVS}_0 &= \mathcal{VS}(\{R(x,z)\}) \\
&= \{P(x), Q(z)\}
\end{aligned}$$

$$\begin{aligned}
\mathcal{CVS}_1 &= \mathcal{VS}(\mathcal{CVS}_0) \\
&= \mathcal{VS}(\{P(x), Q(z)\}) = \{R(x,z), P(x), Q(z)\}
\end{aligned}$$

$$\mathcal{CVS}_2 = \mathcal{VS}(\mathcal{CVS}_1)$$

$= \mathcal{VS}(\{R(x,z), P(x), Q(z)\}) = \{R(x,z), P(x), Q(z)\}$
$= \mathcal{CVS}_1$.

Após introduzidas essas definições, estamos em condições de definir quais respostas consistem em respostas intensionais.

Definição 8. Resposta Intensional. A implicação $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$ corresponde a uma *resposta intensional* se, e somente se, as seguintes propriedades são satisfeitas:

1. existe pelo menos um literal de resposta r_i que contém uma variável;
2. todo literal candidato l_i deve pertencer ao fecho das variáveis compartilhadas dos literais de resposta $\{r_1, \ldots, r_m\}$ em relação ao conjunto $\{l_1, \ldots, l_n, r_1, \ldots, r_m\}$.

Caso essas duas propriedades sejam satisfeitas, então a resposta intensional é a regra $l_1 \wedge \ldots \wedge l_n \rightarrow \mathcal{R}(r_1) \vee \ldots \vee \mathcal{R}(r_m)$, com as variáveis locais ao conseqüente universalmente quantificadas com escopo apenas sobre o conseqüente e as demais variáveis universalmente quantificadas com escopo sobre a fórmula toda.

Para ilustrar a obtenção de respostas intensionais, considere novamente o exemplo de Rich e Knight. As implicações $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$ que não estão associadas a respostas extensionais (e correspondem a respostas) são:

- $gato(Tom) \wedge peixe(x) \rightarrow resp(x)$
- $siames(Tom) \wedge peixe(x) \rightarrow resp(x)$
- $peixe(x) \rightarrow resp(x)$
- $sardinha(x) \rightarrow resp(x)$

As duas primeiras implicações não correspondem a respostas intensionais. Observe que isto é capturado pela definição dada de respostas intensionais, uma vez que $gato(Tom)$,$siames(Tom) \notin \mathcal{CVS}(resp(x))$. Assim, as respostas intensionais geradas nesse exemplo são:

$$\forall x(peixe(x) \rightarrow alimenta(Tom, x))$$

"Tom se alimenta de peixes"

$$\forall x(sardinha(x) \rightarrow alimenta(Tom, x)$$

"Tom se alimenta de sardinhas"

4.5 Respostas Hipotéticas

Uma resposta hipotética é uma regra que compreende duas partes: uma parte é uma resposta extensional ou intensional e a outra é uma condição que deve ser satisfeita para a resposta ser considerada correta. A condição é denominada o *componente hipotético da resposta* e a resposta extensional ou intensional, o *componente de resposta*. O que nos permite distinguir uma resposta hipotética é exatamente a forma como o componente hipotético da resposta é identificado: literais candidatos não pertencentes ao fecho das variáveis compartilhadas dos literais de resposta.

Definição 9. Resposta Hipotética. A implicação $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \ldots \vee r_m$ corresponde a uma *resposta hipotética* se, e somente se, as seguintes propriedades são satisfeitas:

1. existe pelo menos um literal de resposta r_i;
2. existe pelo menos um literal candidato l_i que não pertence ao fecho das variáveis compartilhadas dos literais de resposta $\{r_1, \ldots, r_m\}$ em relação ao conjunto $\{l_1, \ldots, l_n, r_1, \ldots, r_m\}$.

Suponha que $l_1, \ldots, l_k$, $k \leqslant n$, são os literais candidatos que não pertencem ao fecho das variáveis compartilhadas dos literais de resposta $\{r_1, \ldots, r_m\}$. Então, a resposta hipotética é a regra

$$\underbrace{l_1 \wedge \ldots \wedge l_k}_{\text{componente hipotético}} \rightarrow \underbrace{(l_{k+1} \wedge \ldots \wedge l_n \rightarrow \mathcal{R}(r_1) \ldots \vee \mathcal{R}(r_m))}_{\text{componente de resposta}},$$

com as variáveis no componente hipotético existencialmente quantificadas com escopo apenas sobre o componente hipotético. O componente de resposta corresponde a uma resposta extensional ou intensional.

Considerando mais uma vez o exemplo de Rich e Knight, as respostas hipotéticas consistem naquelas regras que não correspondem a uma resposta intensional:

$$gato(Tom) \rightarrow (\forall x[peixe(x) \rightarrow alimenta(Tom, x)])$$

"se Tom é um gato, então Tom se alimenta de peixes"

$$siames(Tom) \rightarrow (\forall x[peixe(x) \rightarrow alimenta(Tom, x)])$$

"se Tom é um siamês, então Tom se alimenta de peixes"

4.6 Respostas Disjuntivas

A definição de respostas apresentada neste capítulo abrange respostas da forma $l_1 \wedge \ldots \wedge l_n \rightarrow r_1 \vee \ldots \vee r_m$, em que todo literal r_i é um literal de resposta. Nas seções anteriores, particionou-se o conjunto de respostas obtidas em três classes de respostas: extensionais, intensionais e hipotéticas. Qualquer um desses tipos de resposta pode conter mais de um literal de resposta. Nesse caso, os literais de resposta constituem uma disjunção na resposta e a resposta é denominada uma *resposta disjuntiva*. Assim, respostas disjuntivas estão associadas a respostas contendo múltiplos literais de resposta. Ou, em uma resposta disjuntiva existe mais de um literal que possui a mesma estrutura da consulta.

Para exemplificar a obtenção de respostas disjuntivas, a base de conhecimentos do exemplo de Rich and Knight foi alterada, passando o exemplo a ser formado pelas seguintes cláusulas:

1. $\neg gato(x) \vee \neg peixe(y) \vee alimenta(x, y)$
2. $\neg raposa(x) \vee \neg esquilo(z) \vee alimenta(x, z)$
3. $\neg sardinha(x) \vee peixe(x)$
4. $sardinha(Tina)$
5. $esquilo(Addie)$
6. $gato(Tom) \vee raposa(Tom)$

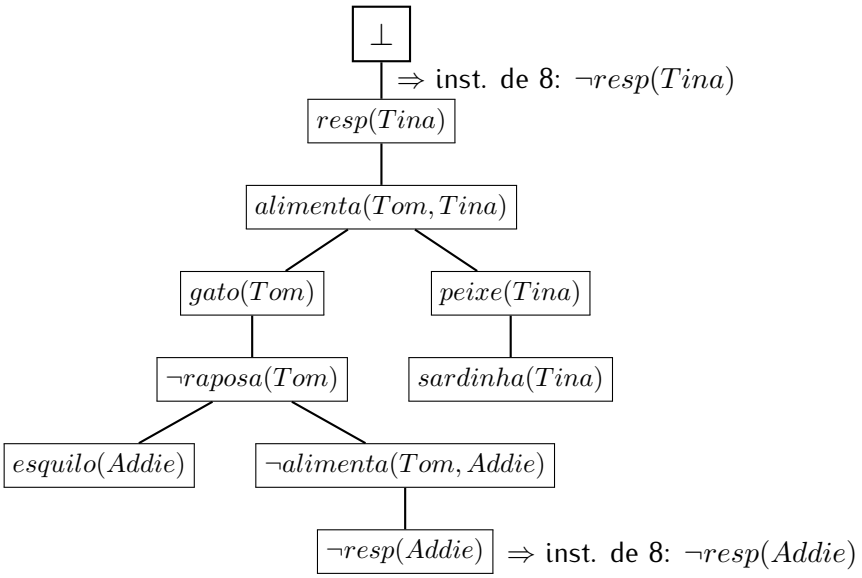
A consulta é a mesma do exemplo anterior: "Do que Tom se alimenta?":

7. $\neg alimenta(Tom, x) \vee resp(x)$

E, por fim, é adicionada a cláusula $\neg resp(x)$:

8. $\neg resp(x)$

Mostra-se abaixo uma árvore de prova fechada para essa consulta:

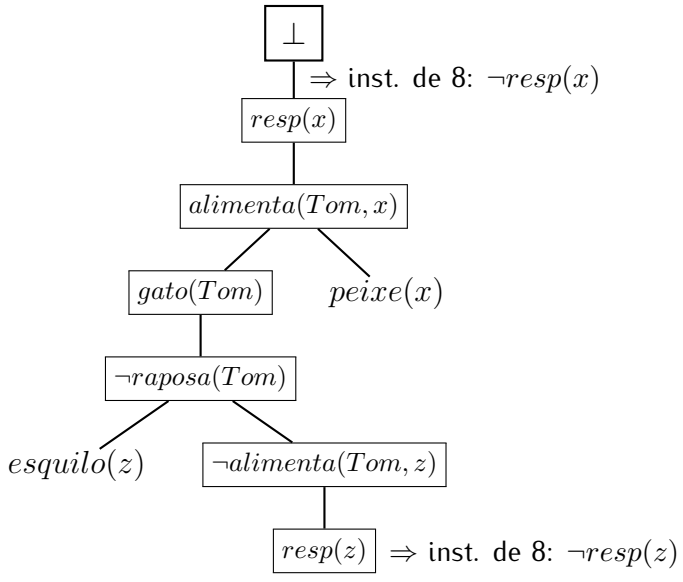


Essa árvore de prova gera um exemplo de resposta disjuntiva extensional:

$$alimenta(Tom, Tina) \vee alimenta(Tom, Addie)$$

"Tom se alimenta de Tina ou Tom se alimenta de Addie"

Os passos intermediários da refutação correspondente a árvore de prova acima podem incluir a seguinte árvore de prova, a qual gera um exemplo de resposta disjuntiva intensional.



A resposta disjuntiva intensional gerada a partir dessa árvore de prova é:

$$(\forall x, z)(peixe(x) \wedge esquilo(z) \rightarrow alimenta(Tom, x) \vee alimenta(Tom, z))$$

"Tom se alimenta de peixes ou Tom se alimenta de esquilos"

O que particularmente provocou a geração de respostas disjuntivas no exemplo anterior foi a presença da cláusula $gato(Tom) \vee raposa(Tom)$ na base de conhecimentos. Isto mostra que respostas disjuntivas podem surgir na presença de informações disjuntivas na base de conhecimentos. Uma outra forma de se obter uma resposta disjuntiva é quando a própria consulta feita é uma consulta disjuntiva. Por exemplo, se fosse feita a consulta $\exists x(peixe(x) \vee passaro(x))$, uma vez que a base de conhecimentos contém os fatos de que "Tina é uma sardinha" e "sardinhas são peixes", então a resposta para a consulta seria: "Tina é um peixe ou Tina é um pássaro". Obviamente que, quando se é feita uma consulta disjuntiva, respostas disjuntivas são esperadas.

A investigação deste trabalho diz respeito à obtenção de respostas disjuntivas extensionais, da forma $P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$, em que cada $P(A_i)$ é uma n-upla de termos fechados. Considere a resposta disjuntiva obtida no exemplo anterior: "Tom se alimenta de Tina ou Tom se alimenta de Addie". Considere, então, a seguinte resposta alternativa à essa resposta disjuntiva: "se Tom é um gato, então Tom se alimenta de Tina; se Tom é uma raposa, então Tom se alimenta de Addie". Uma resposta dessa forma, em que se explicita os casos ou condições para que uma determinada resposta seja a resposta da consulta, é obviamente mais informativa do que a resposta disjuntiva original. É possível determinar tais casos ou condições a partir de uma análise da dedução da resposta disjuntiva.

Chang e Lee (1973) abordam este problema no contexto de um provador de teoremas por resolução. Os autores particionam as consultas em quatro classes - A, B, C, D - de acordo com a forma da resposta obtida para a consulta. A classe D abrange aquelas consultas cujas respostas envolvem o teste ou a satisfação de condições. Como exemplo, consultas que resultam em respostas da forma "se há passagens de ônibus, então viaje para o Rio de ônibus, caso contrário, viaje de trem". Observe que originalmente esta consiste em uma resposta disjuntiva, "viaje para o Rio de ônibus ou viaje para o Rio de trem", e pressuposto pelos autores está que tais condições devem ser determinadas. Os autores mostram que, a partir da análise da árvore de resolução correspondente à dedução da resposta disjuntiva, é possível extrair informações para a consulta. Com esse intuito, os autores propõem um algoritmo que transforma a árvore de resolução em uma árvore que pode ser convenientemente vista como uma árvore de decisão, na

qual os literais no caminho da raiz até um nó folha determinam as condições que, se satisfeitas, implicam na resposta associada a esse nó folha para a consulta.

No capítulo seguinte é proposto um algoritmo com um objetivo semelhante ao algoritmo proposto por Chang e Lee: determinar os casos em que cada elemento $P(A_i)$ de uma resposta disjuntiva $P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$ se verifica, no entanto, para as situações em que a dedução da resposta está representada na forma de uma árvore de prova. O algoritmo proposto realiza transformações na árvore de prova, gerando uma espécie de árvore de decisão, na qual os nós internos contêm associadas as condições (literais ou conjunção de literais) que devem ser satisfeitas para se seguir naquele caminho da árvore e cada nó folha corresponde a uma resposta $P(A_i)$ para a consulta, no caso das condições no caminho da raiz até esse nó serem satisfeitas. As condições no caminho da árvore que vai da raiz até um nó folha com rótulo $P(A_i)$ determinam um caso no qual a resposta $P(A_i)$ se verifica. Uma vez obtida a árvore de decisão, a mesma pode ser facilmente convertida para um conjunto equivalente de regras, gerando uma resposta baseada em casos para a consulta do usuário.

Capítulo 5

Obtenção de Respostas Baseadas em Casos a Partir de APs

5.1 Introdução

Dada uma base de conhecimentos na qual o conhecimento é expresso mediante um conjunto de sentenças da lógica de predicados, uma consulta da forma

determinar X tal que $P(X)$,

em que X é uma n-upla constituída de variáveis livres da sentença $P(X)$, pode, nos sistemas de inferências mais usuais, ter uma resposta disjuntiva da forma

$$P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k),$$

em que $k \geqslant 2$ e A_i é uma n-upla de termos fechados.

A imprecisão existente em uma resposta disjuntiva dessa forma está associada ao fato de não sabermos exatamente qual elemento $P(A_i)$ é verdadeiro, ou quais os são. Uma maneira de reduzir essa imprecisão consiste em explicar a resposta disjuntiva segundo um conjunto de casos possíveis no domínio do problema. Isto é, podemos dividir o problema em um número finito e exaustivo de casos, sendo que cada um dos casos implica em uma resposta $P(A_i)$ para a consulta do usuário.

Como exemplo, suponha que a consulta, determinar x tal que $receitar(\text{Manoel}, x)$, retorne como resposta

$$receitar(\text{Manoel}, R_1) \vee receitar(\text{Manoel}, R_2),$$

onde $receitar(\text{Manoel}, R_z)$ denota que o remédio R_z deve ser receitado a Manoel.

A partir dessa resposta não é possível determinar qual dos remédios, R_1 ou R_2, deve ser receitado a Manoel. No entanto, se, ao invés, fosse retornada a resposta

$$adulto(\text{Manoel}) \rightarrow receitar(\text{Manoel}, R_1)$$
$$\neg adulto(\text{Manoel}) \rightarrow receitar(\text{Manoel}, R_2)$$

e o usuário tivesse conhecimento a respeito da idade de Manoel, então o mesmo determinaria o remédio a ser receitado a Manoel. Observe que na resposta acima são considerados apenas dois casos: o caso em que Manoel é um adulto e o caso em que Manoel não o é.

A dedução de uma resposta carrega consigo informações a respeito da consulta, muitas das quais não são disponibilizadas aos usuários. Árvores de prova são estruturas que armazenam a história de uma dedução. A representação em forma de árvores de prova induz a leitura das cláusulas da base de conhecimentos como implicações, possibilitando a explanação independente dos passos efetivamente realizados pelo sistema de inferência.

No presente capítulo é apresentado um algoritmo que, a partir de uma árvore de prova que corresponde a uma resposta disjuntiva da forma

$$P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k),$$

obtém uma resposta baseada em casos, da forma

$$\upsilon_1 \rightarrow P(A_{i_1})$$
$$\upsilon_2 \rightarrow P(A_{i_2})$$
$$\ldots$$
$$\upsilon_m \rightarrow P(A_{i_m})$$

onde $1 \leqslant i_1, i_2, \ldots, i_m \leqslant k$, $\upsilon_1 \vee \upsilon_2 \vee \ldots \vee \upsilon_m$ é uma conseqüência lógica da base de conhecimentos e υ_j é um caso que implica na veracidade de $P(A_{i_j})$.

O processo de *explanação abdutiva* consiste em, dados uma base de conhecimentos BC e uma fórmula β a ser explicada, obter uma fórmula α para explicar β tal que $BC \cup \{\alpha\} \models \beta$, ou de forma equivalente, $BC \models (\alpha \rightarrow \beta)$ e $BC \cup \{\alpha\}$ é consistente. Dessa forma, pode-se dizer que os casos $\upsilon_1, \upsilon_2, \ldots, \upsilon_m$ constituem explanações para $P(A_{i_1}), P(A_{i_2}), \ldots, P(A_{i_m})$ mais fortes do que explanações abdutivas visto que $\upsilon_1 \vee \upsilon_2 \vee \ldots \vee \upsilon_m$ é conseqüência lógica da base de conhecimentos.

É importante, porém, ressaltar que a valia do algoritmo proposto depende do conhecimento armazenado na árvore de prova. Se o conhecimento é intrinsicamente disjuntivo em relação ao que foi perguntado, então pode não ser possível obter uma resposta da forma ilustrada acima, em que cada caso υ_j implica em uma resposta não disjuntiva $P(A_{i_j})$ para a consulta. Em situações como essa, na resposta gerada pelo algoritmo, pode-se ter um caso υ_j associado a uma resposta disjuntiva r_j, desde que $r_j \subseteq P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$.

Para que a definição de respostas baseadas em casos possa abranger tais situações, a forma geral de uma resposta baseada em casos passa a ser:

$$\upsilon_1 \rightarrow r_1$$
$$\upsilon_2 \rightarrow r_2$$
$$\ldots$$
$$\upsilon_m \rightarrow r_m$$

onde $\upsilon_1 \vee \upsilon_2 \vee \ldots \vee \upsilon_m$ é uma conseqüência lógica da base de conhecimentos e υ_j é um caso que implica na veracidade de r_j. Tem-se ainda que cada caso υ_j é formado por uma conjunção de literais e cada resposta r_j é tal qual $r_j \subseteq P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$. No sentido de precisão de que uma resposta $r_j \subset r'_j$ é mais precisa do que a resposta disjuntiva r'_j, o algoritmo gera a resposta mais precisa possível, dadas as informações disponíveis na árvore de prova.

Inicialmente, é descrito o problema que será utilizado como exemplo para ilustrar a aplicação do algoritmo.

5.2 Problema Exemplo

Considere o problema de determinar o número máximo de empréstimos (de livros) que João pode fazer na biblioteca de uma universidade, sendo João um membro dessa universidade. Na biblioteca da universidade são realizados empréstimos apenas para três tipos de usuários: professores, alunos e funcionários da universidade. Um professor da universidade pode emprestar simultaneamente até 10 livros. O número de empréstimos dos alunos depende se os mesmos são graduandos ou pós-graduandos na universidade. Alunos no nível de graduação podem emprestar até 4 livros simultaneamente e alunos da pós-graduação, até 10 livros simultaneamente. Para os funcionários da universidade, o empréstimo permitido é de 5 livros simultaneamente. A universidade tem ainda uma regra que permite aos funcionários que possuem filhos em idade escolar emprestar 2

livros extras (ou seja, um funcionário que possui filhos em idade escolar pode emprestar até 7 livros simultaneamente).

O seguinte conjunto de cláusulas representa o cenário descrito acima. Essas cláusulas constituem, para o exemplo, a Base de Conhecimentos BC.

1. $prof(x) \vee alun(x) \vee func(x)$
   x é professor ou x é aluno ou x é funcionário

2. $\neg prof(x) \vee empr(x, 10)$
   se x é professor, então x pode emprestar até 10 livros simultaneamente

3. $\neg alun(x) \vee niv(x, Grad) \vee niv(x, Pos)$
   se x é aluno, então x é graduando ou x é pós-graduando

4. $\neg alun(x) \vee \neg niv(x, Grad) \vee empr(x, 4)$
   se x é aluno e x é graduando, então x pode emprestar até 4 livros simultaneamente

5. $\neg alun(x) \vee \neg niv(x, Pos) \vee empr(x, 10)$
   se x é aluno e x é pós-graduando, então x pode emprestar até 10 livros simultaneamente

6. $\neg func(x) \vee \neg possui(x, filhoIE) \vee empr(x, 7)$
   se x é funcionário e x possui filhos em idade escolar, então x pode emprestar até 7 livros simultaneamente

7. $\neg func(x) \vee possui(x, filhoIE) \vee empr(x, 5)$
   se x é funcionário e x não possui filhos em idade escolar, então x pode emprestar até 5 livros simultaneamente

Seja a consulta: determinar y tal que $empr(J, y)$. O conjunto de cláusulas correspondentes a consulta, CQ, é então constituído pela cláusula única:

8. $\neg empr(J, y) \vee resp(y)$

e faz-se $BC' = BC \cup CQ$. Adicionando-se a cláusula $\neg resp(y)$ ao conjunto:

9. $\neg resp(y)$

Das cláusulas de (1) a (9) geramos a seguinte dedução no formato de uma árvore de prova:

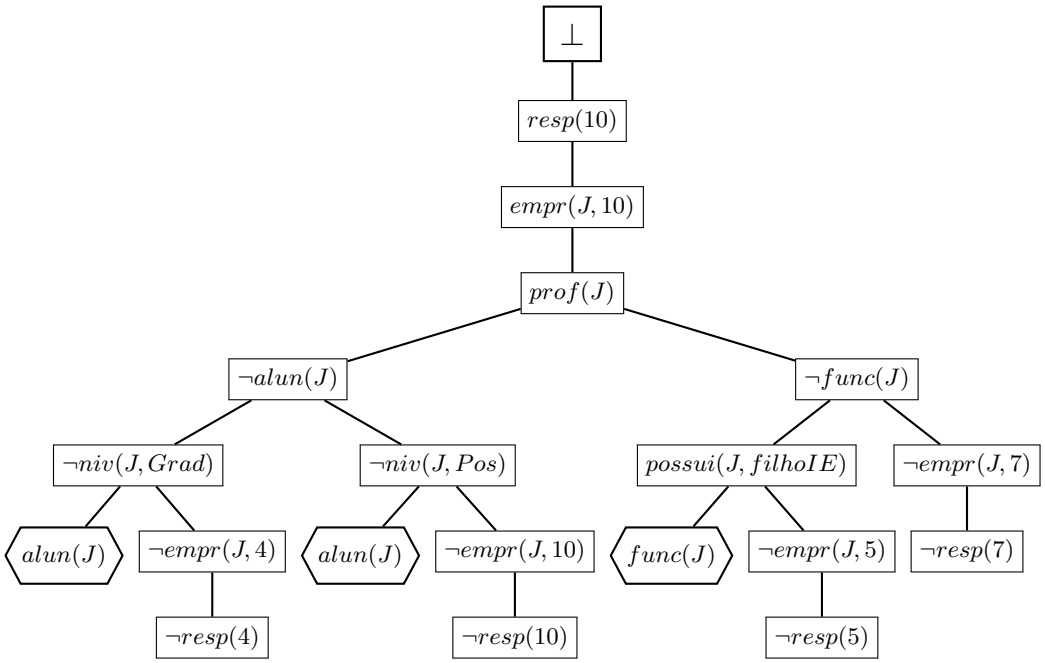


Figura 5.1: Árvore de prova gerada para o problema exemplo

A partir dessa árvore de prova extrai-se a resposta disjuntiva

$$resp(4) \vee resp(5) \vee resp(7) \vee resp(10)$$
$$\Downarrow$$
$$empr(J, 4) \vee empr(J, 5) \vee empr(J, 7) \vee empr(J, 10)$$

para a consulta do usuário. Essa resposta é da forma

$$P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k),$$

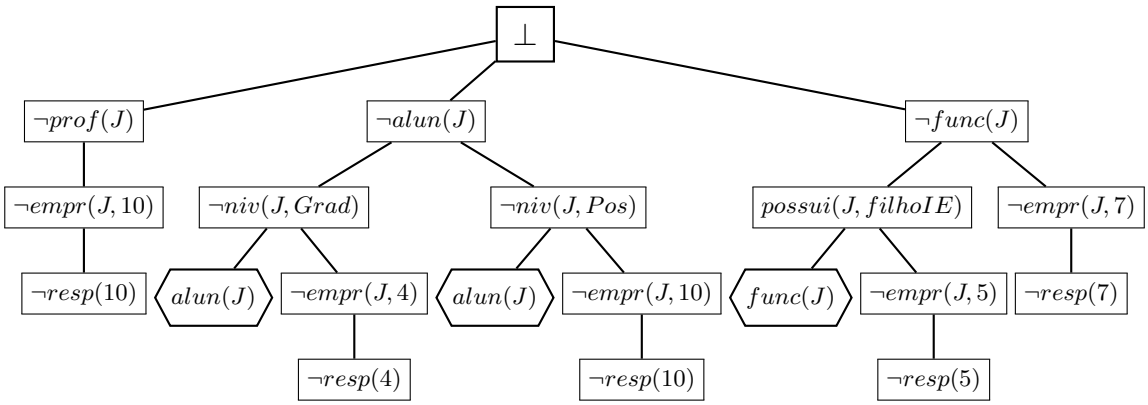
em que $k \geqslant 2$. Será mostrado como, a partir dessa árvore de prova, pode-se obter uma resposta baseada em casos para a consulta.

5.3 Algoritmo para Obtenção de Respostas Baseadas em Casos

O algoritmo proposto obtém uma resposta baseada em casos a partir da dedução gerada para $BC' \cup \{\neg resp(X)\}$ no formato de uma árvore de prova. Vamos denotar essa árvore

de prova por $\mathcal{A}_0$. Para o correto funcionamento do algoritmo é necessário, no entanto, que a cláusula codificada em $\mathcal{A}_0$ não seja $\neg resp(X)$. Isto não limita a aplicabilidade do algoritmo, uma vez que é possível transformar qualquer árvore de prova em outra equivalente onde a cláusula codificada é qualquer uma das utilizadas em expansões. Esta transformação é explicada em (Vieira, 1987) e pode ser realizada de uma maneira algorítmica.

Como exemplo, podemos transformar a árvore de prova ilustrada na Figura 5.1 na seguinte árvore de prova equivalente, onde a cláusula codificada é a cláusula (1) da base de conhecimentos. Para o exemplo, essa árvore de prova irá corresponder à árvore $\mathcal{A}_0$ do algoritmo.



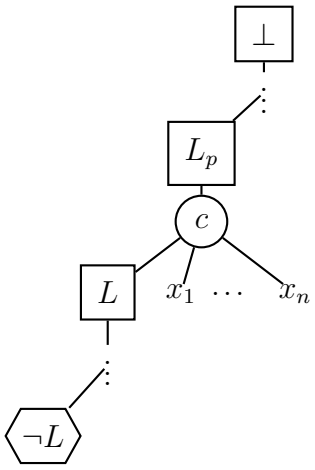
Árvore $\mathcal{A}_0$

O algoritmo é dividido em duas etapas. Na 1^a etapa, uma vez que literais originados a partir da fórmula da consulta não devem ser mencionados nos casos que constituem a resposta, exclui-se de $\mathcal{A}_0$ aqueles literais que constituam instâncias de cláusulas de CQ (cláusulas da consulta), com exceção dos literais de resposta. Para isto, inicialmente deve ser obtida uma árvore de prova $\mathcal{A}'_0$ equivalente a $\mathcal{A}_0$ e que não contenha reduções por literais que consistam em contribuições de cláusulas de CQ. E, posteriormente, tais literais (com exceção dos literais de resposta) devem ser excluídos de $\mathcal{A}'_0$, gerando-se a árvore $\mathcal{A}_1$, a qual é uma árvore de prova para $Th(BC' \cup \{\neg resp(X)\})$[1]. Ou seja, se L_0 é um literal em $\mathcal{A}_1$ e $L_1, L_2, \ldots, L_n$ são os seus filhos, então:

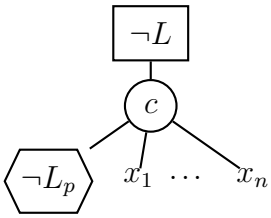
$$BC' \cup \{\neg resp(X)\} \models \overline{L_1} \vee \overline{L_2} \vee \ldots \vee \overline{L_n} \vee L_0.$$

Considere a árvore de prova ilustrada abaixo, onde L é um literal expandido por ou gerado por uma expansão de uma cláusula de CQ e na qual existem reduções por L.

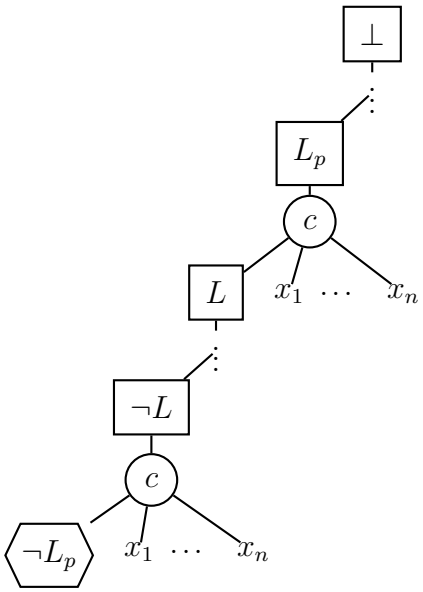
[1]Se S é um conjunto de fórmulas, então $Th(S)$ é o conjunto de todas as fórmulas dedutíveis a partir das fórmulas em S, ou seja, $Th(S) = \{\alpha \mid S \vdash \alpha\}$



Qualquer literal reduzido por L nessa árvore de prova pode ser substituído por

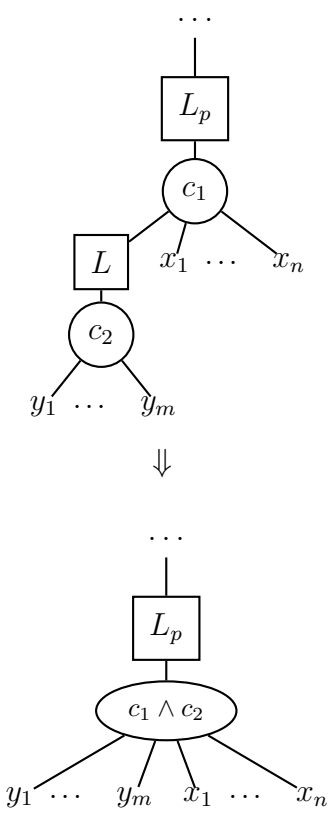


gerando a seguinte árvore de prova, equivalente à árvore de prova original:



Se na árvore de prova resultante ainda houver reduções por literais que façam parte de instâncias de cláusulas de CQ, então deve-se aplicar o mesmo raciocínio, até que seja obtida uma árvore de prova na qual não existam reduções por tais literais.

Dada uma árvore de prova onde não existam reduções por um literal L, mostramos como pode ser excluído o nó associado a L da árvore, de modo a obter uma árvore de prova consistente com o conjunto de cláusulas da base de conhecimentos em questão.



Sejam $X_1, \ldots, X_n, Y_1, \ldots, Y_m$ os literais nas raízes das subárvores $x_1, \ldots, x_n, y_1, \ldots, y_m$, respectivamente. Então, tem-se que:

- $\overline{L} \vee \overline{X_1} \vee \ldots \vee \overline{X_n} \vee L_p$ é uma instância de c_1;
- $\overline{Y_1} \vee \ldots \vee \overline{Y_m} \vee L$ é uma instância de c_2;
- $c_1 \wedge c_2 \models \overline{Y_1} \vee \ldots \vee \overline{Y_m} \vee \overline{X_1} \vee \ldots \vee \overline{X_n} \vee L_p$.

Isto mostra que a partir de uma árvore de prova $\mathcal{A}_0$ para $BC' \cup \{\neg resp(X)\}$ é possível gerar uma árvore de prova $\mathcal{A}_1$ para $Th(BC' \cup \{\neg resp(X\}$, na qual não figuram literais que constituam instâncias das cláusulas de CQ, com exceção dos literais de resposta.

Uma vez que na árvore de prova $\mathcal{A}_0$ da Figura 5.3 não existem reduções por literais gerados a partir de cláusulas de CQ, a árvore de prova $\mathcal{A}_1$ é obtida simplesmente excluindo-se os nós associados a tais literais da árvore, da maneira explicada anteriormente. A árvore $\mathcal{A}_1$ resultante é ilustrada na figura 5.2.

2^a Etapa do Algoritmo

A 2^a etapa do algoritmo tem início a partir da obtenção da árvore de prova $\mathcal{A}_1$, a qual é uma árvore de prova para $Th(BC' \cup \{\neg resp(X)\})$. Descrevemos essa etapa em quatro

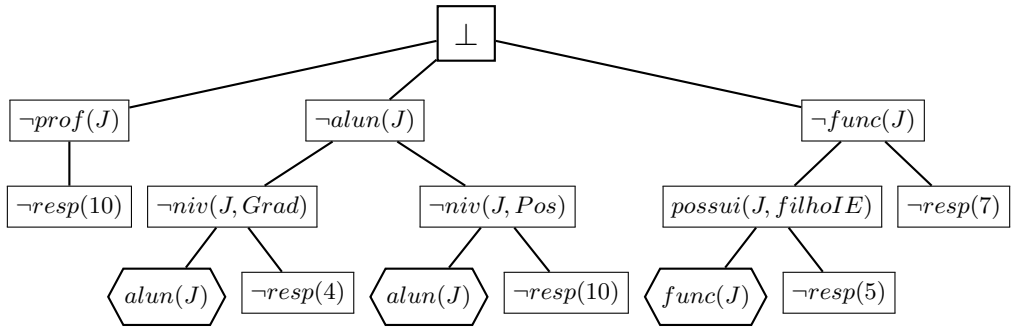


Figura 5.2: Árvore $\mathcal{A}_1$

passos, sendo que em cada passo tem-se como entrada uma árvore $\mathcal{A}_i$ e produz-se como saída uma árvore $\mathcal{A}_{i+1}$. Inicialmente, definimos formalmente o que é uma árvore $\mathcal{A}$ no contexto do algoritmo.

Seja Σ um conjunto de literais. Uma árvore $\mathcal{A}$ é uma quádrupla $(N, A, \mathcal{L}, r)$, onde:

- N é um conjunto finito de nós;
- $A \subseteq N \times N$ é um conjunto de arestas;
- $\mathcal{L}$ é uma função, $\mathcal{L} : N \rightarrow (L_1 \wedge L_2 \wedge \ldots \wedge L_n)$, tal que $L_j \in \Sigma$, para $1 \leqslant j \leqslant n$. Normalmente, tem-se que $n = 1$, ou seja, $\mathcal{L}$ associa a um nó um literal. Na árvore $\mathcal{A}_1$, $\mathcal{L}(\alpha)$ é o literal que rotula o nó α. Para o exemplo, é o literal que aparece explicitado na Figura 5.2.
- $r \in N$ é o nó raiz da árvore.

Um *caminho* do nó α para o nó β é uma seqüência $\nu_1, a_1, \nu_2, a_2, \nu_3, \ldots, \nu_{k-1}, a_{k-1}, \nu_k$ de nós e arestas tal que $\alpha = \nu_1$, $\beta = \nu_k$ e, para cada i, $a_i = (\nu_i, \nu_{i+1})$. O *tamanho* de um caminho é o número de arestas que ele possui. Se o caminho do nó raiz r a um nó α passa por um nó α' (que pode ser r ou α), diz-se que α' é *ancestral* de α e que α é *descendente* de α'. Nesse caso, se (α', α) é uma aresta da árvore, diz-se ainda que α' é o *ancestral imediato*, ou *pai*, de α e que α é um *descendente imediato*, ou *filho*, de α'. Dois nós que possuem o mesmo pai são ditos *irmãos*. Um nó que não possui descendentes é denominado *folha* e um nó que não é folha é denominado nó *interno*. A *altura de um nó* α é o tamanho do maior caminho de α até um de seus descendentes. As folhas têm altura 0.

A seguir, são descritos os quatro passos que constituem a segunda etapa do algoritmo.

Passo 1. Seja uma árvore de prova $\mathcal{A}_1 = (N, A, \mathcal{L}_1, r)$ para $Th(BC' \cup \{\neg resp(X)\})$. Obtenha, a partir de $\mathcal{A}_1$, a árvore $\mathcal{A}_2 = (N, A, \mathcal{L}_2, r)$, em que:

$$\mathcal{L}_2(\alpha) = \overline{\mathcal{L}_1(\alpha)}, \text{ para todo } \alpha \in N.$$

Para o exemplo, a árvore $\mathcal{A}_2$ obtida está ilustrada na Figura 5.3.

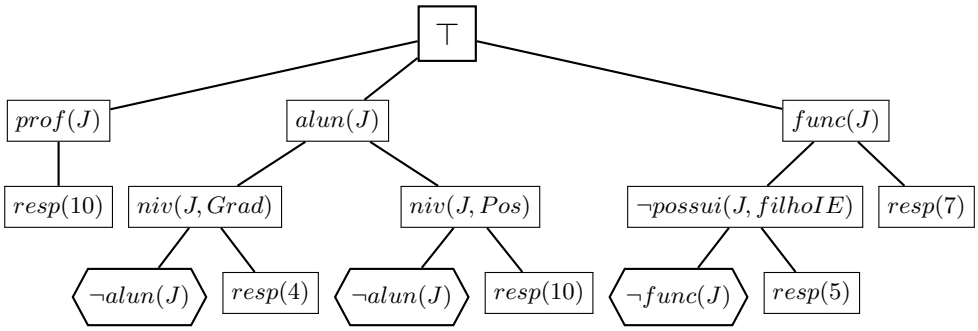


Figura 5.3: Árvore $\mathcal{A}_2$

Com o intuito de se mostrar que o algoritmo proposto é correto, enunciamos a seguinte proposição, a qual será provada verdadeira para toda árvore $\mathcal{A}_i$, $i \geqslant 2$, gerada pelo algoritmo.

Proposição 1. *Considere o conjunto $BC' = BC \cup CQ$. Seja $\mathcal{A}_i = (N, A, \mathcal{L}, r)$, $i \geqslant 2$, uma árvore gerada pelo algoritmo e sejam:*

- *um nó interno $\alpha_n \in N$;*
- *$\alpha_1, \alpha_2, \ldots, \alpha_{n-1} \in N$ os nós ancestrais ao nó α_n em $\mathcal{A}_i$;*
- *$\alpha_{n,1}, \alpha_{n,2}, \ldots, \alpha_{n,m} \in N$ os nós filhos de α_n em $\mathcal{A}_i$.*

Então

$$BC' \models \mathcal{L}(\alpha_1) \wedge \mathcal{L}(\alpha_2) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \mathcal{L}(\alpha_{n,2}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

A prova de que proposição 1 é válida para a árvore $\mathcal{A}_2$ é feita considerando a obtenção de $\mathcal{A}_2$ a partir da árvore de prova $\mathcal{A}_1$.

Demonstração. Uma vez que a árvore $\mathcal{A}_1 = (N, A, \mathcal{L}_1, r)$ é uma árvore de prova para $Th(BC' \cup \{\neg resp(X)\})$ e a cláusula codificada em $\mathcal{A}_1$ não é $\neg resp(X)$, se $\alpha_n \in N$ é um nó interno em $\mathcal{A}_1$ e $\alpha_{n,1}, \alpha_{n,2}, \ldots, \alpha_{n,m}$ são os seus filhos, então

$$BC' \models \overline{\mathcal{L}_1(\alpha_{n,1})} \vee \overline{\mathcal{L}_1(\alpha_{n,2})} \vee \ldots \vee \overline{\mathcal{L}_1(\alpha_{n,m})} \vee \mathcal{L}_1(\alpha_n).$$

A fórmula

$$\overline{\mathcal{L}_1(\alpha_{n,1})} \vee \overline{\mathcal{L}_1(\alpha_{n,2})} \vee \ldots \vee \overline{\mathcal{L}_1(\alpha_{n,m})} \vee \mathcal{L}_1(\alpha_n)$$

é equivalente a

$$\overline{\mathcal{L}_1(\alpha_n)} \rightarrow \overline{\mathcal{L}_1(\alpha_{n,1})} \vee \overline{\mathcal{L}_1(\alpha_{n,2})} \vee \ldots \vee \overline{\mathcal{L}_1(\alpha_{n,m})}$$

e, como $\mathcal{A}_2 = (N, A, \mathcal{L}_2, r)$ e, para todo $\alpha \in N$, $\mathcal{L}_2(\alpha) = \overline{\mathcal{L}_1(\alpha)}$, conclui-se que

$$BC' \models \mathcal{L}_2(\alpha_n) \rightarrow \mathcal{L}_2(\alpha_{n,1}) \vee \mathcal{L}_2(\alpha_{n,2}) \vee \ldots \vee \mathcal{L}_2(\alpha_{n,m}).$$

Como se $p \rightarrow q$, então $p \wedge f \rightarrow q$, para uma fórmula f qualquer, segue-se que

$$BC' \models \mathcal{L}_2(\alpha_1) \wedge \mathcal{L}_2(\alpha_2) \wedge \ldots \wedge \mathcal{L}_2(\alpha_n) \rightarrow \mathcal{L}_2(\alpha_{n,1}) \vee \mathcal{L}_2(\alpha_{n,2}) \vee \ldots \vee \mathcal{L}_2(\alpha_{n,m}).$$

□

Passo 2. Exclua todo nó (e as arestas associadas a tais nós) em $\mathcal{A}_2$ que não seja ancestral de um literal de resposta, obtendo como resultado a árvore $\mathcal{A}_3$. A árvore $\mathcal{A}_3$ obtida para o exemplo está ilustrada na Figura 5.4.

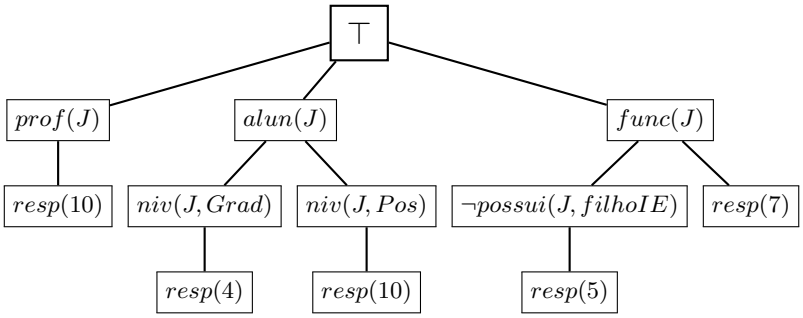


Figura 5.4: Árvore $\mathcal{A}_3$

A seguir, é apresentada a prova de que a proposição 1 é válida para a árvore $\mathcal{A}_3$.

Demonstração. No passo 1 do algoritmo foi provado que se α_n é um nó interno de $\mathcal{A}_2 = (N, A, \mathcal{L}, r)$, então

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,j}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}) = \gamma_1.$$

Vamos agora provar que se $\alpha_{n,j}$ não é ancestral de um literal de resposta, então podemos concluir que

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,j-1}) \vee \mathcal{L}(\alpha_{n,j+1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}) = \gamma_2.$$

Inicialmente, provamos que a conclusão é válida se o nó $\alpha_{n,j}$ possui altura igual a 0 em $\mathcal{A}_2$. Dois casos são considerados:

- o nó $\alpha_{n,j}$ foi gerado pela redução de um literal em $\mathcal{A}_1$. Nesse caso, existe um nó ancestral de $\alpha_{n,j}$ em $\mathcal{A}_2$ cujo rótulo é $\overline{\mathcal{L}(\alpha_{n,j})}$, ou seja, existe i tal que $\mathcal{L}(\alpha_i) = \overline{\mathcal{L}(\alpha_{n,j})}$. Logo, $\gamma_1 \equiv \gamma_2$ e, portanto, se $BC' \models \gamma_1$, então

  $$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,j-1}) \vee \mathcal{L}(\alpha_{n,j+1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

- o nó $\alpha_{n,j}$ foi gerado pela expansão de um literal em $\mathcal{A}_1$. Nesse caso, $\overline{\mathcal{L}(\alpha_{n,j})}$ é um literal expandido não correspondente a um literal de resposta em um nó folha da árvore de prova $\mathcal{A}_1$ e, portanto, tem-se que

  $$BC' \models \overline{\mathcal{L}(\alpha_{n,j})}.$$

  Assim, se $BC' \models \gamma_1$, então

  $$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,j-1}) \vee \mathcal{L}(\alpha_{n,j+1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

Suponha como hipótese de indução que a conclusão é válida para todo nó $\alpha_{n,j}$ que possui altura menor ou igual a k em $\mathcal{A}_2$, sendo $k \geqslant 0$. Vamos mostrar que o resultado é válido para um nó $\alpha_{n,j}$ de altura $k+1$ em $\mathcal{A}_2$. Supondo que

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \wedge \mathcal{L}(\alpha_{n,j}) \rightarrow \mathcal{L}(\alpha_{(n,j),1}) \vee \ldots \vee \mathcal{L}(\alpha_{(n,j),m}),$$

uma vez que todo nó $\alpha_{(n,j),i}$ possui altura menor ou igual a k em $\mathcal{A}_2$, aplicando-se a hipótese de indução m vezes conclui-se que

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \wedge \mathcal{L}(\alpha_{n,j}) \rightarrow \bot,$$

que é equivalente a

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \overline{\mathcal{L}(\alpha_{n,j})}.$$

Logo, se $BC' \models \gamma_1$, então

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,j-1}) \vee \mathcal{L}(\alpha_{n,j+1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

□

Passo 3. Seja α_j um nó em $\mathcal{A}_3$, tal que α_j é o único filho do nó α_{j-1} e $\mathcal{L}(\alpha_j)$ não é um literal de resposta. Exclua o nó α_j da árvore, tornando α_{j-1} o nó pai dos filhos de α_j na árvore resultante. Repita até se obter uma árvore $\mathcal{A}_4$ em que todo nó interno, ou se ramifica em duas ou mais subárvores, ou possui como filho um literal de resposta. Note que para o exemplo tem-se $\mathcal{A}_3 = \mathcal{A}_4$, como ilustrado na Figura 5.5.

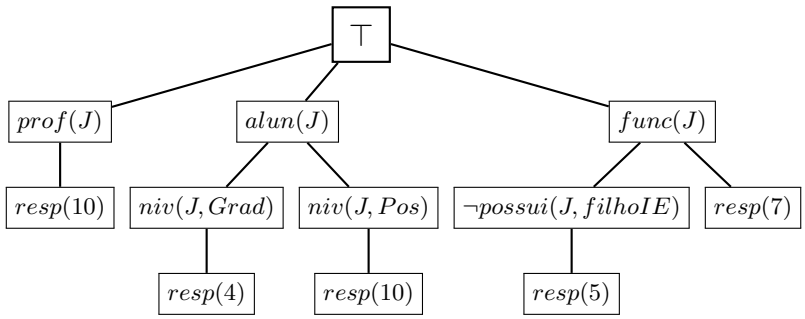


Figura 5.5: Árvore $\mathcal{A}_4$

A seguir, apresentamos a prova de que a proposição 1 se verifica para a árvore $\mathcal{A}_4$.

Demonstração. Neste passo estamos excluindo apenas nós internos de $\mathcal{A}_3 = (N, A, \mathcal{L}, r)$. Foi provado anteriormente que

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_j) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \mathcal{L}(\alpha_{n,2}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

Se α_j é o único filho do nó α_{j-1} em $\mathcal{A}_3$, então segue-se que

$$BC' \models \mathcal{L}(\alpha_1) \wedge \mathcal{L}(\alpha_2) \wedge \ldots \wedge \mathcal{L}(\alpha_{j-1}) \rightarrow \mathcal{L}(\alpha_j),$$

logo

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_{j-1}) \wedge \mathcal{L}(\alpha_{j+1}) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

□

Passo 4. Considere que todo nó interno α_n em $\mathcal{A}_4$ tenha como filhos os nós $\alpha_{n,1}, \ldots, \alpha_{n,m_1}$, $\tau_{n,1}, \ldots, \tau_{n,m_2}$, tal que, para todo i, $\mathcal{L}(\alpha_{n,i})$ não é um literal de resposta e, para todo j, $\mathcal{L}(\tau_{n,j})$ é um literal de resposta. Se $m_1, m_2 \geqslant 1$, então crie um nó α_{n,m_1+1} em $\mathcal{A}_4$ no caminho de α_n a cada nó $\tau_{n,j}$, com o rótulo $\mathcal{L}(\alpha_{n,m_1+1}) = \overline{\mathcal{L}(\alpha_{n,1})} \wedge \ldots \wedge \overline{\mathcal{L}(\alpha_{n,m_1})}$. Assim, o nó α_n passa a ter como filhos os nós $\alpha_{n,1}, \ldots, \alpha_{n,m_1}, \alpha_{n,m_1+1}$ e os nós $\tau_{n,1}, \ldots, \tau_{n,m_2}$ passam a ser filhos de α_{n,m_1+1}. Para o exemplo, a árvore $\mathcal{A}_5$ obtida é ilustrada na Figura 5.6.

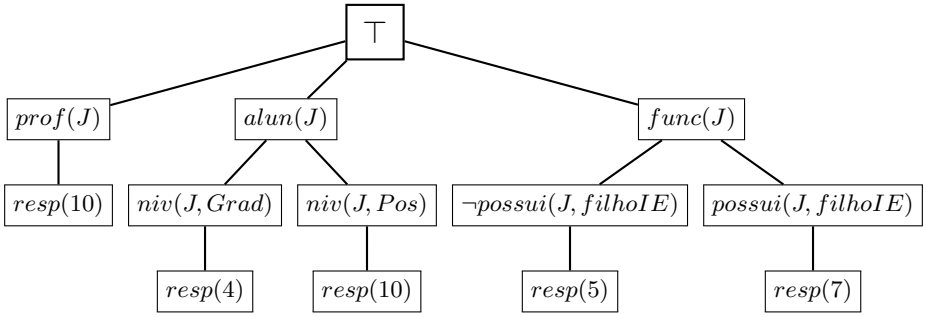


Figura 5.6: Árvore $\mathcal{A}_5$

A prova de que a proposição 1 é válida para a árvore $\mathcal{A}_5$ é apresentada a seguir.

Demonstração. Foi provado para a árvore $\mathcal{A}_4 = (N, A, \mathcal{L}, r)$ que, se $\alpha_n \in N$ tem como filhos os nós $\alpha_{n,1}, \ldots, \alpha_{n,m_1}$, $\tau_{n,1}, \ldots, \tau_{n,m_2} \in N$, então

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee \mathcal{L}(\tau_{n,1}) \vee \ldots \vee \mathcal{L}(\tau_{n,m_2}).$$

Uma vez que se $m_1, m_2 \geqslant 1$, o nó α_n passa a ter como filhos os nós $\alpha_{n,1}, \ldots, \alpha_{n,m_1}, \alpha_{n,m_1+1}$ e os nós $\tau_{n,1}, \ldots, \tau_{n,m_2}$ passam a ser filhos de α_{n,m_1+1}, vamos mostrar que, tendo-se

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee \mathcal{L}(\tau_{n,1}) \vee \ldots \vee \mathcal{L}(\tau_{n,m_2})$$

podemos concluir

1) $BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee \mathcal{L}(\alpha_{n,m_1+1})$

2) $BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \wedge \mathcal{L}(\alpha_{n,m_1+1}) \rightarrow \mathcal{L}(\tau_{n,1}) \vee \ldots \vee \mathcal{L}(\tau_{n,m_2})$

sendo $\mathcal{L}(\alpha_{n,m_1+1}) = \overline{\mathcal{L}(\alpha_{n,1})} \wedge \ldots \wedge \overline{\mathcal{L}(\alpha_{n,m_1})}$.

A prova de 1 é bastante simples, sendo apresentada a seguir. Tem-se que

$$\mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee \mathcal{L}(\alpha_{n,m_1+1}) = \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee (\overline{\mathcal{L}(\alpha_{n,1})} \wedge \ldots \wedge \overline{\mathcal{L}(\alpha_{n,m_1})}),$$

que é uma tautologia. Logo,

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee \mathcal{L}(\alpha_{n,m_1+1})$$

E, por último, apresentamos a prova de 2. Supondo

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m_1}) \vee \mathcal{L}(\tau_{n,1}) \vee \ldots \vee \mathcal{L}(\tau_{n,m_2}),$$

então

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \wedge \overline{\mathcal{L}(\alpha_{n,1})} \wedge \ldots \wedge \overline{\mathcal{L}(\alpha_{n,m_1})} \rightarrow \mathcal{L}(\tau_{n,1}) \vee \ldots \vee \mathcal{L}(\tau_{n,m_2}).$$

Como $\mathcal{L}(\alpha_{n,m_1+1}) = \overline{\mathcal{L}(\alpha_{n,1})} \wedge \ldots \wedge \overline{\mathcal{L}(\alpha_{n,m_1})}$, conclui-se que

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \wedge \mathcal{L}(\alpha_{n,m_1+1}) \rightarrow \mathcal{L}(\tau_{n,1}) \vee \ldots \vee \mathcal{L}(\tau_{n,m_2}).$$

□

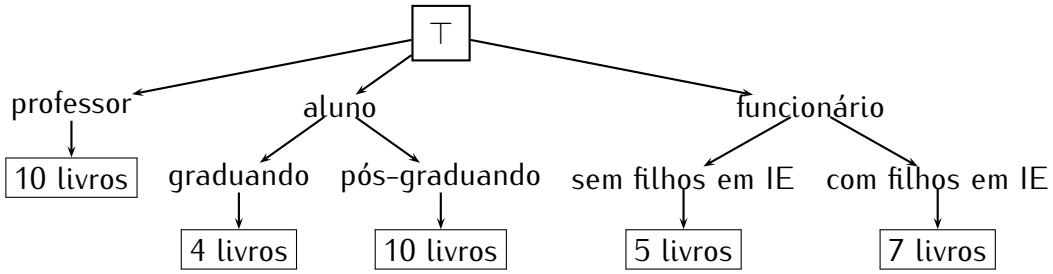
É fácil perceber que após a aplicação dos quatro passos descritos acima, a árvore $\mathcal{A}_5 = (N, A, \mathcal{L}, r)$ resultante possui o seguinte formato:

1. O nó raiz da árvore tem como rótulo o literal $\top$, ou seja, $\mathcal{L}(r) = \top$.
2. Todo nó folha da árvore tem como rótulo um literal de resposta;
3. Todo nó interno da árvore ou se ramifica em duas ou mais subárvores ou possui como filho um literal de resposta;
4. Se α_1 e α_2 são nós irmãos em $\mathcal{A}_5$ e $\mathcal{L}(\alpha_1)$ é um literal de resposta, então $\mathcal{L}(\alpha_2)$ também é um literal de resposta.

E, como provado, se α_n é um nó interno em $\mathcal{A}_5$, $\alpha_1, \ldots, \alpha_{n-1}$ são os nós ancestrais de α_n em $\mathcal{A}_5$ e $\alpha_{n,1}, \ldots, \alpha_{n,m}$ são os nós filhos de α_n em $\mathcal{A}_5$, então

$$BC' \models \mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n) \rightarrow \mathcal{L}(\alpha_{n,1}) \vee \ldots \vee \mathcal{L}(\alpha_{n,m}).$$

Dessa forma, caso não hajam literais de resposta irmãos em $\mathcal{A}_5$, então a árvore $\mathcal{A}_5$ pode ser vista convenientemente como uma árvore de decisão. Para que o leitor possa perceber tal fato, para a árvore $\mathcal{A}_5$ obtida no exemplo, ilustrada na Figura 5.6, apresentamos na figura abaixo uma visão bastante intuitiva dessa árvore como uma árvore de decisão, utilizando a linguagem natural, no lugar da linguagem lógica. Observe que a resposta para a consulta pode ser determinada simplesmente "caminhando" nessa árvore, até se atingir um nó folha ou nó de resposta.



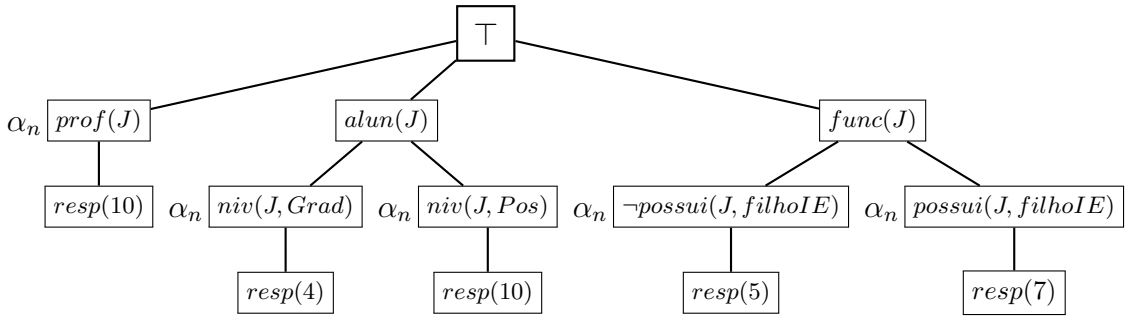
Vamos agora mostrar formalmente como uma resposta baseada em casos pode ser obtida a partir da árvore $\mathcal{A}_5$. Uma resposta baseada em casos possui a forma geral:

$$\begin{gathered} v_1 \rightarrow r_1 \\ v_2 \rightarrow r_2 \\ \ldots \\ v_m \rightarrow r_m \end{gathered}$$

onde v_j é um caso que implica na veracidade de r_j. Para todo nó α_n em $\mathcal{A}_5$, tal que α_n possua como filho um literal de resposta (se um nó possui como filho um literal de resposta, então todos seus filhos correspondem a literais de resposta), a fórmula

$$v_j = (\mathcal{L}(\alpha_1) \wedge \ldots \wedge \mathcal{L}(\alpha_n)) \rightarrow r_j = (\mathcal{R}(\mathcal{L}(\alpha_{n,1})) \vee \ldots \vee \mathcal{R}(\mathcal{L}(\alpha_{n,m}))).$$

constitui a resposta baseada em casos obtida a partir de $\mathcal{A}_5$. Como exemplo, para a árvore $\mathcal{A}_5$ ilustrada na Figura 5.6 indicamos os nós α_n que possuem como filhos pelo menos um literal de resposta.



A resposta baseada em casos obtida a partir dessa árvore é:

$$prof(J) \vee alun(J) \vee func(J)$$

$$prof(J) \rightarrow empr(J, 10)$$
$$alun(J) \wedge niv(J, Grad) \rightarrow empr(J, 4)$$
$$alun(J) \wedge niv(J, Pos) \rightarrow empr(J, 10)$$
$$func(J) \wedge \neg possui(J, filhoIE) \rightarrow empr(J, 5)$$
$$func(J) \wedge possui(J, filhoIE) \rightarrow empr(J, 7)$$

A disjunção na 1^a linha da resposta, formada pelos literais filhos de $\top$ em $\mathcal{A}_5$, pode ser exibida opcionalmente. Segue a resposta acima escrita em linguagem natural:

João é professor ou João é aluno ou João é funcionário

- se João é professor, então João pode emprestar até 10 livros simultaneamente;
- se João é aluno e João é graduando, então João pode emprestar até 4 livros simultaneamente;
- se João é aluno e João é pós-graduando, então João pode emprestar até 10 livros simultaneamente;
- se João é funcionário e João não possui filhos em idade escolar, então João pode emprestar até 5 livros simultaneamente;
- se João é funcionário e João possui filhos em idade escolar, então João pode emprestar até 7 livros simultaneamente.

Capítulo 6

Conclusão

As principais contribuições deste trabalho consistiram na proposta de uma resposta alternativa - denominada resposta baseada em casos - para as situações em que o sistema de consulta-resposta gera uma resposta disjuntiva e na descrição de um algoritmo para a obtenção de tais respostas a partir da dedução na forma de árvore de prova.

Para que esses objetivos pudessem ser alcançados, inicialmente era necessário definir em que consiste uma resposta no contexto de um sistema de prova baseado em árvores de prova. Assim, no capítulo 4 deste trabalho, foi apresentada uma definição formal de respostas nesse contexto, a qual é uma adaptação da definição de respostas feita por Burhans (2002) para o contexto de um mecanismo de prova por resolução. A definição de respostas dada difere das definições usuais, as quais vinculam a obtenção de uma resposta à existência de uma prova, na medida que considera-se que mesmo árvores de prova geradas em passos intermediários da dedução (ou seja, árvores de prova que não são fechadas) produzem informações relevantes para a consulta na forma de respostas intensionais ou hipotéticas.

Uma vez definido em que consiste uma resposta, o passo seguinte foi investigar o principal objeto de pesquisa deste trabalho, que diz respeito à obtenção de respostas disjuntivas em um sistema de consulta-resposta baseado em árvores de prova. Qualquer um dos três tipos de respostas - extensionais, intensionais e hipotéticas - pode corresponder a uma resposta disjuntiva para a consulta. Em particular, a investigação deste trabalho diz respeito à obtenção de respostas disjuntivas extensionais, da forma $P(A_1) \vee P(A_2) \vee \ldots \vee P(A_k)$. Uma maneira de reduzir a imprecisão existente a uma resposta disjuntiva dessa forma consiste em determinar os casos em que cada elemento da disjunção se verifica. Ou, de forma alternativa, pode-se considerar um conjunto exaustivo de casos no domínio do problema, sendo que cada caso implica em uma resposta

$P(A_i)$, $1 \leqslant i \leqslant k$, para a consulta. Com este intuito, no capítulo 5 do trabalho, foi proposto um algoritmo que, a partir da dedução na forma de árvore de prova, gera uma resposta baseada em casos para a consulta, da forma

$$
\begin{aligned}
&v_1 \rightarrow P(A_{i_1}) \\
&v_2 \rightarrow P(A_{i_2}) \\
&\ldots \\
&v_m \rightarrow P(A_{i_m})
\end{aligned}
$$

onde $1 \leqslant i_1, i_2, \ldots, i_m \leqslant k.$, $v_1 \vee v_2 \vee \ldots \vee v_m$ é uma conseqüência lógica da base de conhecimentos e v_j é um caso que implica na veracidade de $P(A_{i_j})$.

A vantagem em se fornecer respostas baseadas em casos, no lugar de respostas disjuntivas, é que o conhecimento de que o caso v_j é verdadeiro no contexto do problema considerado permite ao usuário deduzir a resposta $P(A_{i_j})$ para a sua consulta. Assim, em aplicações em que respostas disjuntivas são inaceitáveis, uma resposta baseada em casos pode ser apropriada.

Não foram encontrados trabalhos na literatura que lidam com o problema referente à obtenção de respostas disjuntivas da maneira como foi tratado aqui: fornecendo uma resposta baseada em casos, ao invés. Dessa forma, pode-se dizer que o presente trabalho estende trabalhos anteriores, propondo uma solução nova para as situações em que respostas disjuntivas são obtidas. E essa solução consiste na definição de um novo tipo de resposta para sistemas de consulta-respostas: uma resposta baseada em casos.

Como trabalhos futuros, propõe-se tonar o algoritmo não determinístico e estudar a aplicação do algoritmo a partir de um modelo de usuário, visando a possibilidade de escolha do vocabulário apropriado para expressar os casos que constituem a resposta.

Referências Bibliográficas

Bessie, J. e Glennan, S. (2000). *Elements of Deductive Inference: An Introduction to Symbolic Logic*. Wadsworth Publishing.

Burhans, D. T. (2002). *A Question Answering Interpretation of Resolution Refutation*. PhD thesis, State University of New York at Buffalo.

Burhans, D. T. e Shapiro, S. C. (2007). Defining answer classes using resolution refutation. *J. Applied Logic*, 5(1):70–91.

Chang, C.-L. e Lee, R. C.-T. (1973). *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc., Orlando, FL, USA.

Cholvy, L. (1990). Answering queries addressed to a rule base. *Revue d'Intelligence Artificielle*, 4(1):79–98.

Cholvy, L. e Demolombe, R. (1986). Querying a rule base. In *Proc. from the First International Conference on Expert Database Systems*.

Copi, I. M. (1978). *Introdução à Lógica*. Mestre Jou.

Demolombe, R. (1992). A strategy for the computation of conditional answers. In *ECAI '92: Proceedings of the 10th European conference on Artificial intelligence*.

Demolombe, R. (1997). Uncertainty in intelligent databases. In *Uncertainty Management in Information Systems*, pp. 89–126. Kluwer Academic Publishers.

Gallaire, H.; Minker, J. e Nicolas, J.-M. (1984). Logic and databases: A deductive approach. *ACM Comput. Surv.*, 16(2):153–185.

Green, C. C. (1969). *The application of theorem proving to question-answering systems*. PhD thesis, Stanford, CA, USA.

Green, C. C. (1990). Application of theorem proving to problem solving. In Allen, J.; Hendler, J. e Tate, A., editores, *Readings in Planning*, pp. 67–87. Kaufmann, San Mateo, CA.

Herbrand, J. (1967). Investigations in proof theory: The properties of true propositions. In *From Frege to Gödel. A Source Book in Mathematical Logic 1879–1931*, pp. 525–81.

Huth, M. R. A. e Ryan, M. D. (2000). *Logic in Computer Science: Modelling and Reasoning about Systems*. Cambridge University Press, Cambridge, England.

Luckham, D. C. e Nilsson, N. J. (1971). Extracting information from resolution proof trees. *Artificial Intelligence*, 2(1):27–54.

Motro, A. (1994). Intensional answers to database queries. *IEEE Trans. on Knowl. and Data Eng.*, 6(3):444–454.

Reiter, R. (1977). An approach to deductive question-answering systems. *SIGART Bull.*, (61):41–43.

Rich, E. e Knight, K. (1990). *Artificial Intelligence*. McGraw-Hill Higher Education.

Vieira, N. J. (1987). *Máquinas de Inferência para Sistemas Baseados em Conhecimento*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro.

Vieira, N. J. (1996). Processamento de conhecimento. Relatorio Técnico 1, UFMG.