# PROLOG MODAL DE AÇÃO E REVISÃO DE CRENÇAS EM CONJUNTOS DEFINIDOS

Odinaldo Teixeira Rodrigues

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO EM ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovado por :

Mario Roberto Folhadela Benevides, Ph.D.
(Presidente)

Gerson Zaverucha, Ph.D.

Edward Hermann Haeusler, Doutor em Computação

RIO DE JANEIRO, RJ – BRASIL
Outubro de 1993

RODRIGUES, ODINALDO TEIXEIRA

PROLOG MODAL DE AÇÃO E REVISÃO DE CRENÇAS EM CONJUNTOS DEFINIDOS [Rio de Janeiro] 1993.
X, 141 p., 29,7cm (COPPE/UFRJ, M.Sc., Engenharia de Sistemas e Computação, 1993
Tese – Universidade Federal do Rio de Janeiro, COPPE.

1 – Lógica Modal de Ação
2 – Programação em Lógica
3 – Revisão de Crenças
4 – Problema da Geração de Planos
I. COPPE/UFRJ
II. Título (série).

A minha família,
sem cujo apoio este trabalho
não seria possível.

**** ****

# Agradecimentos

Ao professor e amigo Mario Benevides. Que sempre presente, dedicado e positivamente influente, soube manter-se firme no tênue limite entre uma amizade marcante e um profissionalismo competente e paciente.

Ao professor Gerson Zaverucha, por inúmeras discussões realizadas sobre o assunto da tese, pela amizade e pela colaboração incansável nos assuntos acadêmicos.

Ao professor Edward Hermann Haeusler, por sua cordialidade e participação.

Aos demais professores da linha de Inteligência Artificial, pela dedicação, incentivo e pelo ambiente estimulador proporcionado.

Aos amigos Lilian Freitas, Denise Carneiro, Claudia Ceci, Fabio Marques, Marcelo Aragão e Evande Araújo, o meu agradecimento especial pelo suporte emocional, carinho e dedicação.

A todos aqueles, em especial aos colegas e funcionários da COPPE/Sistemas, que de alguma forma contribuíram para a elaboração desta tese.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

# PROLOG MODAL DE AÇÃO E REVISÃO DE CRENÇAS EM CONJUNTOS DEFINIDOS

**Odinaldo Teixeira Rodrigues**

**Outubro, 1993**

Orientador : Mario Roberto Folhadela Benevides
Programa de Engenharia de Sistemas e Computação

Neste trabalho nós apresentamos o Prolog Modal de Ação, um formalismo para representação do problema da geração de planos. Baseado numa Lógica Modal de Ação, o mesmo possui uma semântica procedural similar à da Linguagem de Programação PROLOG.

O Prolog Modal de Ação pode ser visto como uma extensão do método da Resolução-LSD que permite a presença de literais com modalidades nas regras. Além de efetuar provas de propriedades de cenários específicos é possível também investigar cenários onde tais propriedades sejam verificadas.

Assim como na maioria dos formalismos lógicos, nós enfrentamos o problema de *Frame*. De modo a evitá-lo, nós propusemos uma extensão do nosso formalismo com operadores de Revisão de Crenças baseados na Teoria de Revisão de Crenças de Gärdenfors.

Como a Teoria de Revisão de Crenças não foi originalmente desenvolvida para uso em Programação em Lógica, a mesma foi adaptada e foram sugeridas diversas alternativas para obtençao das mudanças epistemológicas nesse contexto. Dessa forma, três critérios para determinar funções de Contração e dois para Expansão foram apresentados. Foi também analisado o efeito das Revisões sobre Conjuntos Definidos usando a regra da Negação por Falha Finita para obter negação.

Uma versão do Prolog Modal de Ação usando Revisão de Crenças foi então apresentada. Como antes, a informação acerca do estado inicial do sistema pode ser definida por um conjunto de fatos sobre os quais algumas regras podem ser aplicadas. As novas regras modais introduzidas torna possível a representação dos operadores de Expansão e Contração e determinam o modo pelo qual as mudanças nos cenários ocorrem.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

# PROLOG MODAL DE AÇÃO E REVISÃO DE CRENÇAS EM CONJUNTOS DEFINIDOS

**Odinaldo Teixeira Rodrigues**

**October, 1993**

Thesis Supervisor : Mario Roberto Folhadela Benevides
Department : Computing and Systems Engineering

In this work we present the Modal Action Prolog, a formalism for representing the planning problem. Based on the concepts of Modal Action Logic, it has a procedural semantics similar to the logic programming language PROLOG.

Modal Action Prolog may be regarded as an extension to SLD-Resolution which allows the rules of a Definite Set to have literals with modalities. Besides accomplishing proofs of specific sceneries properties it is possible to search sceneries which have desired properties.

Similarly to most of the logic formalisms, we faced the so-called frame problem. In order to avoid it, we proposed an extension of our Modal Action Prolog with Belief Revision operators which are based on Gardenfors' Theory of Belief Revision.

Since the Theory of Belief Revision was not originally developed for use in Logic Programming, we adapted it and suggested several ways in which the epistemological changes could be reached in such context. Thus, we presented three criteria for determining functions for Contraction and two ones for Expansion. We also analyzed the effect of doing Revisions in Definite Sets using the Negation as Failure Rule as a way to obtain negation.

Then we presented the Belief Revision version of our Modal Action Prolog. As before, the information in the initial state can be defined by a set of PROLOG facts over which some rules can be applied. The new modal rules introduced makes it possible to state the operators of Contractions e Expansions and determine the way in which the changes in the sceneries occur.

# ÍNDICE

# Capítulo 1

# INTRODUÇÃO

Nesse trabalho, foram analisados meios de representação e obtenção de soluções para o problema da geração de planos. Através da Programação em Lógica, nós gostaríamos de utilizar conceitos subjacentes da Lógica Modal de Ação para representar a dinâmica das informações acerca de estados de um sistema.

Este trabalho propõe a utilização de um formalismo lógico para a representação do problema da geração de planos, baseado numa Lógica Modal de Ação, com semântica procedural semelhante à do Prolog. Para isso, nós estendemos a noção de Conjuntos quase-Definidos de forma a permitir-lhes a presença de literais com modalidades nas cláusulas.

Um literal com modalidade associa a propriedade correspondente a ele, ao cenário correspondente à modalidade. Nesse formalismo, a noção de mudança de estado é representada quando o literal no conseqüente de uma regra possui uma modalidade. Intuitivamente, a execução da(s) ação(ões) da parte modal desse conseqüente num cenário onde o antecedente seja verificado leva o sistema a um novo cenário onde a propriedade do literal do conseqüente é verificada. Os novos Conjuntos assim definidos foram chamados de Conjuntos quase-Definidos Modais.

Adaptando a regra da Extensão do método de Resolução-LSD, foi desenvolvido um método similar capaz de encontrar refutações a partir de Conjuntos quase-Definidos Modais. Nesse método, a cláusula objetivo representa as propriedades de cenários para as quais se pretende obter uma prova. Quando é possível encontrar uma refutação a partir do Conjunto quase-Definido Modal, então os literais da cláusula objetivo são provados nos cenários relativos às suas modalidades.

Sob o ponto de vista do problema da geração de planos, seria mais interessante descobrir em que cenários as propriedades da cláusula objetivo são verificadas, pois dessa

forma um plano que levasse o sistema do cenário inicial a um cenário com tais propriedades poderia ser fornecido. Isso foi atingido através da introdução de uma nova regra que em adição à anterior possibilita refutações de propriedades de cenários genéricos.

Apesar de extremamente simples, o método enfrenta o assim chamado *Problema de Frame*. Além das propriedades que mudam pela execução das ações, aquelas que permanecem inalteradas precisam ser especificadas de modo a continuarem a serem verificadas nos cenários atingidos por essa execução. O que se gostaria era simplesmente de atualizar um cenário de acordo com as mudanças sugeridas pela execução das ações deixando inalteradas as propriedades não afetadas pela execução.

Com esse objetivo, foi feita uma analogia entre as informações de um cenário dedutíveis a partir de um Conjunto Definido e um Conjunto de Crenças (que é uma representação das convicções e crenças de um indivíduo num determinado ponto no tempo). Assim, uma ação estaria associada a uma entrada epistêmica e sua execução à mudança epistêmica ocasionada por aquela entrada.

Utilizando o formalismo de Revisão de Crenças proposto por [Gär88], as Mudanças Epistemológicas seriam utilizadas de modo a obter as atualizações desejadas nos cenários. A verificação de novas propriedades no cenário resultante seriam obtidas por Expansões e aquelas propriedades que deixariam de ser verificadas seriam contraídas.

Foi então analisada a aplicabilidade dessas mudanças a Conjuntos Definidos e meios pelos quais elas poderiam ser obtidas. Alguns critérios para obtenção de Contrações e Expansões foram sugeridos e suas conseqüências nos Conjuntos resultantes. Além disso, os postulados propostos originalmente por [Gär88] foram analisados tendo em vista o novo contexto.

Achamos que seria interessante também algum mecanismo para obtenção de negação, quando então Revisões também seriam possíveis. A Revisão foi analisada segundo a negação fornecida pelo método da Negação por Falha Finita.

A partir dos critérios assim definidos para obtenção de Contrações e Expansões, passamos a visualizar um cenário como um Conjunto Definido. Novamente, regras modais descrevem o mecanismo de atuação das ações, fornecendo um meio para representação dos operadores das mudanças epistemológicas.

Foi então definido um provador no qual as propriedades que se pretende que sejam verificadas em algum cenário são representadas por cláusulas objetivo. Os cenários são representados por Conjuntos Definidos sobre os quais as mudanças epistemologicas

relacionadas às ações são executadas gerando cenários. Quando algum cenário possui as propriedades da cláusula objetivo, a seqüência de ações executadas desde o cenário inicial é tomada como um plano para a solução do problema.

A primeira parte da tese contem uma revisão bibliografica acerca dos principais conceitos utilizados no trabalho. No Capítulo 2, uma Lógica Modal de Ação é apresentada a fim de introduzir as características desse formalismo relevantes ao problema.

O Capítulo 3 faz uma introdução ao processo de Revisão de Crenças. Esse formalismo tem por objetivo caracterizar os diferentes estados de crença de um indivíduo e formas pelas quais as mudanças epistemológicas podem ser realizadas. Essas mudanças se constituem basicamente na Expansão, quando o indivíduo passa a acreditar em proposições em que não acreditava antes; na Contração, quando o indivíduo deixa de acreditar em algumas proposições; e por fim na Revisão, que é quando o indivíduo passa a acreditar em proposições que contradizem proposições em que antes acreditava.

O Capítulo 4 contem uma revisão acerca dos principais conceitos de Programção em Lógica e, em particular, da Linguagem das Cláusulas de Horn. O Método da Resolução-LSD e da Resolução-LSDNF são discutidos. Além disso, esse capítulo apresenta algumas novas definições que se mostrarão úteis em capítulos posteriores.

O Capítulo 5 contem algumas considerações levantadas sobre o problema da geração de planos. Os principais tipos de sistemas utilizados nesse processo são apresentados e é efetuada uma análise de dois formalismos existentes.

O Capítulo 6 apresenta uma nova Linguagem, o Prolog Modal de Ação, para a qual foi desenvolvido o mecanismo de refutação citado anteriormente.

O Capítulo 7 propõe formas pelas quais os processos de Contração e de Expansão podem ser realizados; considerações sobre os postulados originais para Conjuntos Definidos e ainda uma análise da mudança epistemológica Revisão em Conjuntos pseudo-Definidos com Negação por Falha Finita.

O Capítulo 8 introduz o Prolog Modal de Ação com Revisão de Crenças, em cuja linguagem fornecemos um meio para representação dos operadores de Revisão. Esse capítulo pode ser visto como uma aplicação das propriedades analisadas no capítulo anterior para o problema da geração de planos.

O Capítulo 9 encerra o texto com algumas conclusões obtidas e efetuando propostas para estender o trabalho realizado.

# Capítulo 2

# LÓGICA MODAL DE AÇÃO

## 2.1 Introdução

A Lógica Modal de Ação foi proposta tendo em vista a necessidade de uma representação clara e formal das informações a respeito de um sistema de computação. Esse capítulo é inteiramente baseado nos trabalhos de [Kho88] e [Mai87].

O termo Modal procura dar a intuição de dinâmica. Ou seja, que a Lógica pretende capturar o modo pelo qual o sistema muda de um estado a outro. Essas mudaças são ocasionadas pela execução de ações.

Em virtude de o sistema ser composto por uma série de objetos desempenhando papéis específicos, uma Lógica poli-sortida é utilizada como formalismo subjacente para explicitar a multiplicidade de categorias a que um objeto em particular pode pertencer.

Essas categorias são definidas por um conjunto de *Sortes*. Um *Tipo* é constituído por uma seqüência não vazia de sortes. A idéia é fazer com que um objeto do discurso possua um Sorte associado a ele e que as propriedades do sistema estejam associadas a um Tipo. Um tipo especial de Sorte é reservado para denotar as *Ações*.

Dessa forma, existe um conjunto de variáveis e constantes para cada Sorte e cada predicado está associado a um Tipo. Um símbolo funcional está associado a um Tipo, que consiste de uma seqüência de sortes e um Sorte (a do objeto do discurso denotado pela função).

Um conjunto de axiomas extra-lógicos denominado Especificação procura estabelecer as propriedades intrínsecas do sistema, isto é, que descrevem as características gerais de comportamento do sistema. Um outro conjunto procura estabelecer as in-

formações relativas apenas ao estado inicial.

Numa analogia com uma Lógica Modal, a modalidade $\Box$ é substituída por uma família de modalidades, uma associada a cada ação. Uma regra similar à regra da Necessitação assegura que as propriedades do sistema descritas na Especificação sejam verificadas em todos os estados.

## 2.2 Linguagem de Primeira Ordem

DEFINIÇÃO 2.1 : ***Alfabeto***

- *Um Sorte não-vazio de ações Ac*
- *Uma Coleção finita de Sortes* $S = Ac \bigcup \{s_1, s_2, \ldots\}$
- *Constantes: para cada Sorte* $s \in S$ *existe um conjunto possivelmente vazio de símbolos de constante cada qual dita ser de tipo* $s$
- *Nomes de Ação: para cada* $n > 0$ *e cada ênupla* $< s_1, s_2, \ldots, s_n >$ *tal que* $s_1 \in S$, $s_2 \in S$, ..., $s_n \in S$, *existe um conjunto possivelmente vazio de nomes de ação n-ários, pertencendo a Ac cada qual dito ser de tipo* $< s_1, s_2, \ldots, s_n >$
- *Símbolos Predicativos: para cada* $n > 0$ *e cada ênupla* $< s_1, s_2, \ldots, s_n >$ *tal que* $s_1 \in S$, $s_2 \in S$, ..., $s_n \in S$, *existe um conjunto possivelmente vazio de símbolos predicativos n-ários, cada qual dito ser de tipo* $< s_1, s_2, \ldots, s_n >$
- *Símbolos Funcionais: para cada* $n > 0$ *e cada* $n+1$*-tupla* $< s_1, s_2, \ldots, s_{n+1} >$ *tal que* $s_1 \in S$, $s_2 \in S$, ..., $s_{n+1} \in S$, *existe um conjunto possivelmente vazio de símbolos funcionais n-ários, cada qual dito ser de tipo* $< s_1, s_2, \ldots, s_{n+1} >$
- *Símbolos de Igualdade: para alguns Sortes* $s \in S$ *(possivelmente todos), pode existir um símbolo predicativo especial* $=_s$ *de Sorte* $< s, s >$ *representado igualdade entre objetos do* Sort $s$
- *Variáveis: Uma Coleção infinita de variáveis distintas para cada Sorte* $s \in S$
- *Quantificadores: para cada Sorte* $s \in S$ *o quantificador universal* $\forall_s$ *e o quantificador existencial* $\exists_s$

- *Símbolos Lógicos:* $\neg$, $\leftrightarrow$, $\rightarrow$, $\wedge$, $\vee$, [ e ].
- *Símbolos de Pontuação:* (, ) e ,.

### *Símbolos da Linguagem*

- *termo:*
  - *Se* $s \in S$, *então toda variável ou constante de Sorte* $s$ *é um termo*
  - *Se* $t_1,t_2,\ldots,t_n$ *são termos de Sortes* $s_1$, $s_2$, ..., $s_n$ *respectivamente,* $s_1$, $s_2$, ..., $s_n$ *pertencem todos a* $S$ *e* $f$ *é um símbolo funcional de tipo* $< s_1,s_2,\ldots,s_{n+1} >$ *então* $f(t_1,t_2,\ldots,t_n)$ *é um termo de Sorte* $s_{n+1}$
  - *Se* $t_1,t_2,\ldots,t_n$ *são termos de Sortes* $s_1$, $s_2$, ..., $s_n$ *respectivamente,* $s_1$, $s_2$, ..., $s_n$ *pertencem todos a* $S$ *e* $a$ *é um nome de ação de tipo* $< s_1,s_2,\ldots,s_n >$ *então* $a(t_1,t_2,\ldots,t_n)$ *é um termo de Sorte* $Ac$
  - *Nada além disso é um termo*
- *Átomos:*
  - *Se* $t_1,t_2,\ldots,t_n$ *são termos de Sortes* $s_1$, $s_2$, ..., $s_n$ *respectivamente,* $s_1$, $s_2$, ..., $s_n$ *pertencem todos a* $S$ *e* $p$ *é um símbolo predicativo de tipo* $< s_1,s_2,\ldots,s_n >$ *então* $p(t_1,t_2,\ldots,t_n)$ *é um átomo*
  - *Para cada Sorte* $s \in S$, *dados dois termos* $t_1$ *e* $t_2$ *de* $s$, *se* $=_s$ *estiver na Linguagem então* $t_1 = t_2$ *é um átomo*
  - *Nada além disso é um átomo*
- *fórmulas bem-formadas* (wffs)*:*
  - *um átomo é uma fórmula*
  - *Se* $\varphi$ *é uma fórmula, então assim o é* $\neg\varphi$
  - *Se* $\varphi$ *e* $\psi$ *são fórmulas, então assim o são* $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $(\varphi \rightarrow \psi)$, $(\varphi \leftrightarrow \psi)$
  - *Se* $\alpha$ *é um termo de Sorte* $Ac$ *e* $\varphi$ *é uma fórmula, então assim o é* $[\alpha](\varphi)$
  - *Para cada Sorte* $s \in S$, *se* $x$ *é uma variável de Sorte* $s$ *e* $\varphi$ *é uma fórmula então* $\forall_s x\varphi$ *e* $\exists_s x\varphi$ *são fórmulas*
  - *Nada além disso é uma fórmula*

## 2.3 Sistema Dedutivo

O Sistema Dedutivo será apresentado através da introdução de um sistema axiomático. Para simplificar, inicialmente serão descritos apenas os axiomas que definem a atuação dos operadores lógicos de Implicação ($\rightarrow$) e Negação ($\neg$) e do quantificador universal ($\forall$). Os outros operadores e o quantificador existencial ($\exists$) serão definidos em função destes e em seguida serão apresentados os axiomas que descrevem a interação entre as modalidades e o conjunto de axiomas básicos apresentado. No Esquema abaixo, $\varphi$, $\gamma$ e $\psi$ são fórmulas da Linguagem, $x$ e $y$ são variáveis e $\alpha$ é um termo de Sorte $Ac$.

***Axioma Schema* para os operadores lógicos básicos**

- $\varphi \rightarrow (\psi \rightarrow \varphi)$
- $(\varphi \rightarrow (\psi \rightarrow \gamma)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \gamma))$
- $(\neg\psi \rightarrow \neg\varphi) \rightarrow ((\neg\psi \rightarrow \varphi) \rightarrow \psi)$
- $(\forall x(\varphi \rightarrow \psi)) \rightarrow (\varphi \rightarrow \forall x\psi)$, onde $x$ não está livre em $\varphi$
- $(\forall x\varphi(x)) \rightarrow \varphi(t)$, onde $x$ é substituível por $t$

Os operadores $\vee$, $\wedge$, $\leftrightarrow$, e o quantificador existencial ($\exists$) são agora introduzidos por definição:

DEFINIÇÃO 2.2 :

$$\begin{aligned} \varphi \vee \psi &\cong (\neg\varphi) \rightarrow \psi \\ \varphi \wedge \psi &\cong \neg(\varphi \rightarrow \neg\psi) \\ \varphi \leftrightarrow \psi &\cong (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi) \\ \exists x\varphi &\cong \neg\forall x\neg\varphi \end{aligned}$$

Até então, ainda não foi descrito a relação entre as modalidades e os símbolos lógicos da linguagem. O seguinte Axioma Schema apresenta essa interação e introduz um axioma para a igualdade que permite a existência de designadores não-rígidos nessa Lógica.

***Axioma Schema* para as Modalidades**

1. $[\alpha]\top$
2. $([\alpha](\varphi \rightarrow \psi)) \rightarrow (([\alpha]\varphi) \rightarrow ([\alpha]\psi))$
3. $([\alpha]\neg\varphi) \rightarrow (\neg[\alpha]\varphi)$
4. $\forall x([\alpha]\varphi) \leftrightarrow ([\alpha]\forall x\varphi)$, onde $\alpha$ não está livre para $x$
5. $([\alpha]\varphi \wedge \psi) \leftrightarrow ([\alpha]\varphi \wedge [\alpha]\psi)$
6. $([\alpha]\varphi \vee [\alpha]\psi) \rightarrow [\alpha](\varphi \vee \psi)$
7. $\exists x([\alpha]\varphi) \rightarrow ([\alpha]\exists x\varphi)$, onde $\alpha$ não está livre para $x$
8. $(EQ)$ $\forall x, y((x = y \wedge \varphi(x)) \rightarrow \varphi(y))$ onde $x$ não aparece no escopo de um operador modal

No axioma (1), $\top$ representa qualquer tautologia da Lógica apresentada e assegura que apenas extensões consistentes de uma Especificação sejam usadas como cenários atingidos pelas ações. O axioma (2) preserva o sentido da regra de Modus Ponens através dos cenários. (3) é uma componente deôntica do sistema. Note que se fosse definida uma implicação também no sentido recíproco, o sistema pressuporia informação completa nos estados, pois para qualquer fórmula $\varphi$ se $\varphi$ não é verificada num estado, então sua negação o é. Claramente, isso não é uma situação desejada pois a motivação inicial pressupunha que os estados poderiam estar descritos de forma incompleta. O axioma (4) representa a fórmula de Barcan e sua recíproca. Portanto o domínio é considerado constante. Apesar disso, esse domínio é apenas um conjunto dos objetos potenciais do sistema. A mudança entre objetos nos estados (por exemplo numa operação de abrir uma conta de cliente), é modelada por predicados de existência cuja extensão pode variar de um estado a outro. Os axiomas (5) e (6) estabelecem a distributividade da modalidade sobre a conjunção e a disjunção. O axioma (7) não seria necessário caso (3) não fosse apenas unilateral. O axioma (8) fornece a mecânica usual da igualdade, mas só afirma que essa mecânica é válida para objetos de um mesmo cenário (por isso a restrição de a variável não ocorrer no escopo de um operador modal).

### Regras de Inferência

### Modus Ponens

Dada uma prova para $\varphi$ e para $\varphi \rightarrow \psi$ é possível inferir $\psi$

**Generalização**

Dada uma prova para $\varphi$ é possível inferir $\forall x\varphi$

A partir dos axiomas e regras de inferência acima é possível definir a relação de derivabilidade $\vdash_L$. $\vdash$ define a noção de derivabilidade de uma fórmula a partir de um conjunto de fórmulas, é portanto uma relação entre Conjuntos de Fórmulas e Fórmulas. Dessa forma, se $Form(L)$ é o conjunto de todas as fórmulas da Linguagem $L$, então $\vdash_L \subseteq 2^{Form(L)} X Form(L)$. O subscrito será omitido quando o contexto não oferecer dúvida.

Sejam $\Gamma$ um conjunto de fórmulas e $\varphi$ uma fórmula da linguagem $L$. Dizemos que $\Gamma \vdash_L \varphi$ sse existir uma seqüência finita de fórmulas de $L$, tais que cada uma é uma instância de um axioma, uma fórmula de $\Gamma$, ou foi obtida através destas pela aplicação de uma das regras de inferência. Além disso, se uma fórmula na seqüência foi obtida pela regra da Generalização, então a variável correspondente não pode ter ocorrido livre em nenhuma fórmula em posição anterior na seqüência.

Os Teoremas da Lógica são todas as fórmulas $\varphi$ tais que $\Gamma \vdash \varphi$, onde $\Gamma = \emptyset$. Um sistema de prova do tipo Tableau para a Lógica acima pode ser encontrado em [CdC87].

Usando a relação $\vdash$, as regras de inferência podem ser mais formalmente estabelecidas da seguinte forma:

**Regras de Inferência**

**Modus Ponens**

$$\frac{\vdash \varphi \text{ e } \vdash \varphi \rightarrow \psi}{\psi}$$

**Generalização**

$$\frac{\vdash \varphi}{\forall x}$$

Uma Apresentação de Teoria $AT$ é um par $<\mathbf{L}_{AT}, \mathbf{Ax}_{AT}>$, onde $\mathbf{L}_{AT}$ é a Linguagem extralogica da Teoria, na qual são expressos os axiomas específicos da Teoria $\mathbf{Ax}_{AT}$. Uma Teoria para uma Apresentação de Teoria $AT$ é definida por

$$TH(AT) = \{\varphi : \mathbf{Ax} \vdash \varphi\}$$

A informação acerca de um estado do sistema é representada por uma Teoria. Essa Teoria pode ser gerada a partir de uma Apresentação de Teoria do tipo definida acima. Ao estado inicial corresponde uma Apresentação de Teoria cuja Teoria representa esse cenário específico. Por sua vez, a Especificação do funcionamento do sistema é efetuada também em termos de uma Teoria. O que se pretende é que a Especificação capture

de modo intensional toda a dinâmica do sistema, não sendo os cenários apresentados como um conjunto de Apresentações de Teoria mas sim como extensões da Especificação.

As regras de inferência e os axiomas definidos acima não são suficientes para expressar o mecanismo de funcionamento de um sistema. É necessário que a especificação realizada para o sistema seja verificada em todo os estados, uma vez que é ela quem determina o comportamento do mesmo. Novamente fazendo uma analogia com uma Lógica Modal é necessário uma regra do tipo da regra da Necessitação para obter esse efeito. Vamos chamar essa regra de $(NEC)$.

É então que surge um problema: tudo que for descrito a respeito do sistema deve ser expresso através de axiomas extra-lógicos, inclusive as propriedades do estado inicial. Como não se deseja que as propriedades do estado inicial sejam verificadas nos outros estados, o que se tenciona fazer é restringir a aplicação de $(NEC)$ apenas aos Teoremas Lógicos ou da Especificação. Uma forma de efetuar isso é tomar um subconjunto da Teoria sobre o qual a regra da Necessitação possa ser aplicada. Se $ESP$ é a Apresentação de Teoria da Especificação, esse conjunto é tal que possui todos os Teoremas Lógicos e os Teoremas da Especificação e é fechado sobre $(NEC)$. Dessa forma,

$$\text{Se } \varphi \in \Gamma \text{ então } [\alpha]\varphi \in \Gamma \quad (NEC)$$

onde $\Gamma$ é o menor conjunto de sentenças tal que $TH(ESP) \subseteq \Gamma$, $\Gamma$ é fechado sobre $(NEC)$ e $\alpha \in Ac$.

[Kho88] introduz uma Lógica Modal de Ação, chamada de **SFOAL**, onde uma descrição mais extensa dos conceitos aqui apresentados e exemplos de especificação podem ser encontrados; além de uma Semântica para a Lógica. [Mai87] é um relatório que faz parte de um projeto de Engenharia de Software, onde uma Lógica similar voltada para a especificação de sistemas de tempo real é apresentada. Essa Lógica permite também que os agentes executores das ações sejam expressos e pretende modelar a noção de tempo em sistemas distribuídos.

# Capítulo 3

# RESUMO DE REVISÃO DE CRENÇAS

## 3.1 Introdução

Uma **Teoria Epistemológica** é um aparato utilizado para modelar a dinâmica de crenças e conhecimentos de um indivíduo. Para tanto, deve fornecer meios de representação dos elementos epistêmicos e critérios que guiem as variações desses elementos.

Um **Estado Epistêmico** (ou estado de crenças) é uma representação das convicções ou conhecimentos de um indivíduo num certo ponto do tempo. O comportamento do mesmo com relação a uma nova informação é denominado **Atitude Epistêmica.**

Quando uma informação provoca num indivíduo uma reação, essa informação é chamada de **Entrada Epistêmica** e a reação denominada **Mudança Epistêmica** ou simplesmente **Mudança de Crenças.**

Um estado epistêmico de um indivíduo é representado por um conjunto de sentenças que correspondem àquelas em que ele acredita no estado de crenças modelado.

Uma sentença $A$ induz num indivíduo os seguintes tipos de atitudes epistêmicas:

- Aceitação
- Rejeição
- Indeterminação

Se considerarmos uma sentença $A$ como aceita sse $\neg A$ for rejeitada, é possível reduzir os tipos de atitudes acima a apenas dois.

**Conjuntos de Crenças** são conjuntos de sentenças que podem ser racionalmente suportadas por um indivíduo. Dessa forma, dois critérios de racionalidade possíveis para esses conjuntos são:

i) consistência do conjunto

ii) fechamento em relação a conseqüência lógica

É assumido a existência de uma relação de conseqüência $\vdash$ entre as sentenças de um Conjunto de Crenças a qual define uma Lógica que:

i) $\vdash_L A$, onde $A$ é uma tautologia

ii) é fechada sobre Modus Ponens

iii) é consistente, ou seja, $\nvdash_L \bot$

Além disso, assume-se que a Lógica acima é compacta e que $\vdash_L$ satisfaz o Teorema da Dedução. Um Conjunto de Crenças pode agora ser definido mais precisamente da seguinte forma:

Um conjunto de sentenças $K$ é um Conjunto de Crenças (não-absurdo) sse:

- $\bot$ não é uma conseqüência lógica das sentenças em $K$
- Se $K \vdash B$ então $B \in K$

O conjunto de todas as conseqüências lógicas de um conjunto $K$ $(A : K \vdash A)$ é denotado por $Cn(K)$ e é chamado de conjunto conseqüência de K. Logo,

**(Cn)** Se $K$ é um Conjunto de Crenças então $K = Cn(K)$

O conjunto $L$ de todas as sentenças da linguagem é considerado um Conjunto de Crenças e chamado de Conjunto de Crenças *Absurdo* $(K_\bot)$. O menor Conjunto de Crenças é o conjunto com apenas as sentenças válidas, isto é, $Cn(\emptyset)$.

**Definição 3.1** *Uma linguagem $L$ é completa sse para toda seqüência $(A_i)_{i\in I}$, onde $I$ é um conjunto índice, existem sentenças $\bigcup_{i\in I}(A_i)$ e $\bigcap_{i\in I}(A_i)$ em $L$ representando respectivamente as disjunções e conjunções de todas as sentenças em $(A_i)_{i\in I}$.*

*Uma relação de conseqüência lógica $\vdash$ é completa sse esta satisfizer as seguintes condições:*

*i) para todo* $A_i$, $A_i \vdash \bigcup(A_i)$

*ii) para todo* $A_i$, $\bigcap A_i \vdash A_i$

*iii) Se* $A_i \vdash B$, *para todo* $A_i$, *então* $\bigcup(A_i) \vdash B$

*iv) Se* $C \vdash A_i$, *para todo* $A_i$, *então* $C \vdash \bigcap(A_i)$

Um Conjunto de Crenças $K$ é completo sse ele é fechado sobre uma lógica completa. A conjunção de todas as sentenças em $K$ é também uma sentença em $K$ e é denominada *Determinador de* $K$ (denotada por $\bigcap K$), uma vez que para qualquer sentença $A$, $A \in K$ sse $\bigcap K \vdash A$.

## 3.2 Expansões, Revisões e Contrações

Expansões, Revisões e Contrações são operações que determinam a dinâmica de um Conjunto de Crenças. Se $K$ é um Conjunto de Crenças, então para qualquer sentença $A$:

i) Se $A \in K$ então $A$ é aceita; ou

ii Se $\neg A \in K$ então $A$ é rejeitada; ou

iii) Se $A \notin K$ e $\neg A \notin K$ então $A$ é indeterminada.

As *Expansões* ocorrem para as sentenças indeterminadas, mudando a atitude epistêmica em relação à sentença para *Aceita* ou *Rejeitada* e correspondem ao resultado de observações ou de aceitação de informação transmitida lingüisticamente.

As *Contrações* consistem em mudar a atitude em relação a uma sentença $A$, de i) ou ii) para iii), ou seja, uma determinada crença em $A$ ou em $\neg A$ é abandonada.

O terceiro tipo de operação é a *Revisão*, e consiste em mudar a atitude com relação a uma sentença $A$ de i) para ii) ou vice-versa.

Ao contrário das Revisões, as Expansões e Contrações são mudanças consistentes em um estado de crenças, pois nenhuma crença de uma Expansão ou Contração contradiz qualquer outra de um estado de crenças anterior. Ou seja, a mesma atitude epistêmica com relação às sentenças do conjunto anterior é mantida no novo conjunto.

### 3.2.1 Expansões

As Expansões correspondem ao ato de aprender algo. Se $K$ é o Conjunto de Crenças inicial, a Expansão de $K$ por $A$ é denotada por $K_A^+$. $+$ é uma função de pares de Conjuntos de Crenças e sentenças ($\mathbf{K} \times \mathbf{L}$) em Conjuntos de Crenças ($\mathbf{K}$). Os postulados para a Expansão são os seguintes:

($K^+1$) $K_A^+$ é um Conjunto de Crenças

($K^+2$) $A \in K_A^+$

($K^+3$) $K \subseteq K_A^+$

($K^+4$) Se $A \in K$, então $K_A^+ = K$

($K^+5$) Se $K \subseteq H$, então $K_A^+ \subseteq H_A^+$

($K^+6$) Para todos os Conjuntos de Crenças $K$ e sentenças $A$, $K_A^+$ é o menor conjunto que satisfaz ($K^+1$) a ($K^+5$)

O primeiro postulado assegura que o resultado da Expansão é também um Conjunto de Crenças. O segundo que a sentença expandida está na Expansão. ($K^+3$) afirma que a Expansão preserva tudo que havia antes no conjunto expandido e ($K^+4$) que nada lhe é acrescentado quando a sentença a expandir já lhe pertencia. ($K^+5$) é um postulado que garante a monotonicidade entre expansões de conjuntos e ($K^+6$) assegura que apenas as crenças relacionadas à sentença a expandir são acrescentadas durante a Expansão.

### 3.2.2 Revisões

As Revisões correspondem ao ato de adicionar uma nova crença quando esta contradiz crenças anteriores. Uma das principais dificuldades desse processo é se determinar de quais crenças deve-se abrir mão. O projeto de modelagem das Revisões é muito importante no estudo da teoria do raciocínio não-monotônico.

Pelo critério de economia informativa, apenas o menor número de crenças possível deve ser descartado, sugerindo uma mudança mínima em $K$ para acomodar o novo conceito. Essa restrição de minimalidade contudo, pode ser satisfeita de diversas maneiras.

É assumido que para qualquer Conjunto de Crenças $K$ e qualquer sentença $A$, a Revisão representando a mudança mínima selecionada é efetuada de forma única. Os postulados básicos são os seguintes:

($K$*1) $K_A^*$ é um Conjunto de Crenças

($K$*2) $A \in K_A^*$

($K$*3) $K_A^* \subseteq K_A^+$

($K$*4) Se $\neg A \notin K$, então $K_A^+ \subseteq K_A^*$

($K$*5) $K_A^* = K_\perp$ sse $\vdash \neg A$

($K$*6) Se $\vdash A \leftrightarrow B$, então $K_A^* = K_B^*$

($K$*7) $K_{A \wedge B}^* \subseteq (K_A^*)_B^+$

($K$*8) Se $\neg B \notin K_A^*$, então $(K_A^*)_B^+ \subseteq K_{A \wedge B}^*$

O primeiro postulado, como no caso das expansões, garante que a função de Revisão leva a um Conjunto de Crenças . ($K$*2) afirma que uma sentença $A$ pertence à Revisão de $K$ por $A$. O terceiro e quarto postulados sugerem que as Expansões são um caso especial das Revisões (quando não é necessário se remover nada). ($K$*5) assegura que o Conjunto de Crenças revisado é consistente a menos que a negação da proposição a revisar seja uma tautologia. O sexto postulado determina que as Revisões de proposições logicamente equivalentes em relação a um mesmo Conjunto de Crenças são iguais. Os dois últimos postulados dizem respeito a mudanças sucessivas em Conjuntos de Crenças.

A Revisão é uma mudança de crenças não-monotônica com relação ao item ($K^+$5), significando que se um conjunto é subconjunto de outro, a Revisão do primeiro em relação a uma sentença não necessariamente estará contida na Revisão do segundo em relação à mesma sentença. Dessa forma, o seguinte postulado, denominado *Princípio da Monotonicidade* não pode ser aceito como necessariamente válido:

($K$*$M$) **Se** $K \subseteq H$, **então** $K_A^* \subseteq H_A^*$.

**Exemplo 3.1** *Um contra-exemplo de [Gär88] para* ($K$*$M$).

*Sejam duas sentenças* $B$ *e* $C$ *tais que* $B \nvdash C$ *e* $C \nvdash B$, *e conseqüentemente,* $\nvdash B \wedge C$ *e dois Conjuntos de Crenças* $K$ *e* $K'$ *tais que* $K = Cn(B)$ *e* $K' = Cn(C)$. *Além*

*disso, tome um Conjunto de Crenças $H$ tal que $H = Cn(B \wedge C)$. Claramente, $K \subseteq H$ e $K' \subseteq H$. A Revisões de $K$ e $K'$ por $\neg(B \wedge C)$ são respectivamente $Cn(B \wedge \neg C)$ e $Cn(\neg B \wedge C)$. Assumindo-se $(K^*M)$ como válido então a Revisão de $H$ por $\neg(B \wedge C)$ deveria conter as sentenças $B$, $\neg C$, $\neg B$ e $C$, o que é um absurdo por $(K^*5)$ pois $\nvdash (B \wedge C)$.*

### 3.2.3 Contrações

O principal problema com relação às Contrações é que ao se remover uma crença $A$ de um Conjunto de Crenças $K$, podem haver outras crenças em $K$ que a derivem (ou outras crenças que apenas juntas a derivem). Dessa forma, outras crenças têm eventualmente que ser removidas a fim de se manter a consistência.

Novamente temos o problema de se determinar quais crenças devem ser removidas e o critério da economia informativa pode outra vez ser aplicado.

É assumido que para qualquer Conjunto de Crenças $K$ e qualquer sentença $A$, a Contração de $K$ por $A$ é realizada de modo único de forma que $K_A^-$ é bem definida. Isso é feito assumindo-se que existe uma função de Contração $-$ de $\mathbf{K} \times \mathbf{L}$ em $\mathbf{K}$, que leva um Conjunto de Crenças $K$ e uma sentença $A$ num novo Conjunto de Crenças $K_A^-$. Para as Contrações seguem-se os seguintes postulados:

$(K^-1)$ $K_A^-$ é um Conjunto de Crenças

$(K^-2)$ $K_A^- \subseteq K$

$(K^-3)$ Se $A \notin K$, então $K_A^- = K$

$(K^-4)$ Se $\nvdash A$ então $A \notin K_A^-$

$(K^-5)$ Se $A \in K$ então $K \subseteq (K_A^-)_A^+$

$(K^-6)$ Se $\vdash A \leftrightarrow B$, então $K_A^- = K_B^-$

$(K^-7)$ $K_A^- \bigcap K_B^- \subseteq K_{A \wedge B}^-$

$(K^-8)$ Se $A \notin K_{A \wedge B}^-$ então $K_{A \wedge B}^- \subseteq K_A^-$

O primeiro postulado assegura que o resultado de uma Contração é um Conjunto de Crenças. O segundo que nada é adicionado numa Contração. $(K^-3)$ que se a proposição a contrair não pertence ao conjunto também nada é retirado. $(K^-4)$ afirma

que a menos que a proposição a ser contraída seja uma tautologia a mesma não pertence à Contração. O quinto postulado das Contrações sugere um princípio de recuperação entre Contrações e Expansões sucessivas de uma mesma proposição. Ou seja, se um Conjunto for contraído e imediatamente expandido em relação à mesma proposição tudo que havia antes no mesmo é recuperado. $(K^-6)$ segue a mesma motivação do postulado equivalente da Revisão. $(K^-7)$ afirma que crenças que estão tanto na Contração de $K$ por $A$ quanto na de $K$ por $B$ também estão na Contração de $K$ por $A \wedge B$ e juntamente com $(K^-8)$ introduz o conceito de Mudança Mínima para as Contrações. $(K^-8)$ determina que apenas a crença em um dos componentes de uma conjunção deve ser abandonada quando a Contração dessa conjunção for requerida.

## 3.3 Mudança Mínima

Mudança Mínima corresponde à noção de que as mudanças epistêmicas de um indivíduo necessárias para acomodar um novo conceito devem ser mínimas. Isso é motivado pelo seguinte princípio da conservatividade:

**(Cons)** Na mudança de crenças em resposta a uma nova evidência, deve-se continuar a acreditar em tantas crenças antigas quanto possível.

O uso do tamanho do Conjunto de Crenças é utilizado para comparar a magnitude da mudança simplesmente porque é o único possível em virtude da fraca estrutura interna dos Conjuntos de Crenças.

## 3.4 Relações entre Contrações e Revisões

Revisões e Contrações podem ser definidas uma em função da outra, acarretando numa simplificação das Teorias Epistemológicas.

Segundo [Lev7a], as únicas mudanças de crenças legítimas são a Contração e a Expansão, pois a Revisão poderia ser analisada como uma seqüência de Contrações e Expansões. A Revisão de um Conjunto de Crenças $K$ por uma sentença $A$, $K_A^*$ é definida em termos de Expansões e Contrações por:

$$K_A^* = (K_{\neg A}^-)_A^+$$

A correspondência acima é chamada de **Identidade de Levi**.

A motivação por trás dessa identidade é que para se fazer a Revisão de uma proposição é necessário inicialmente preparar o conjunto, retirando dele tudo o que contradiz a proposição a ser adicionada.

Similarmente, Harper propôs a definição da Contração em função da Revisão da seguinte forma:

$$K_A^- = K \bigcap K^*_{\neg A}.$$

Essa correspondência é chamada de **Identidade de Harper.**

## 3.5 Construção de Funções de Contração

Conforme visto anteriormente (por $(K^*3)$ e $(K^*4)$), a Expansão pode ser vista como um caso especial da Revisão, que por sua vez pode ser determinada em função da Contração via Identidade de Levi. Dessa forma, a definição de mecanismos que governem as Contrações é suficiente para se determinar os três tipos de mudança epistêmica existentes. Seria possível também analisarmos apenas as Revisões, obtendo-se as Contrações pela Identidade de Harper, mas isso não será efetuado aqui.

Pelo princípio da economia informativa a Contração de um Conjunto de Crenças $K$ em relação à proposição $A$, $K_A^-$, deve conter tanto quanto possível de $K$ sem contudo derivar $A$.

Se for possível especificar os Entrincheiramentos Epistêmicos relativos às proposições de um Conjunto de Crenças, isto é, se for possível especificar alguma ordenação na importância das proposições, a Contração pode também basear-se nesse princípio no momento de decidir de quais proposições deve-se abrir mão.

### 3.5.1 Funções de Contração *MaxiChoice*

Seguindo o critério da economia informativa, $K_A^-$ deve ser um subconjunto de $K$ que não derive $A$, tão grande quanto possível. Uma das formas de se determinar uma função de Contração é portanto se escolher dentre os maiores subconjuntos de $K$ que não derivam $A$.

Mais precisamente, um Conjunto de Crenças $K'$ é um subconjunto maximal de $K$ que não deriva $A$ sse:

i) $K' \subseteq K$,

ii) $A \notin K'$,

iii) para toda proposição $B$ que está em $K$ mas não em $K'$, $B \rightarrow A$ está em $K'$.

O conjunto de todos os subconjuntos maximais de $K$ que não derivam $A$ é denotado por $K \perp A$. A função de Contração *Maxichoice* é definida da seguinte forma:

**Definição 3.2** *Função* Maxichoice

$$K_A^- = \begin{cases} S(K \perp A) & \textit{quando } \nvdash A, \textit{ onde } S \textit{ é uma função de seleção} \\ K & \textit{caso contrário} \end{cases}$$

Como em geral existem diversos subconjuntos maximais de $K$ que não derivam $A$, a função de seleção $S$ é utilizada a fim de escolher um dentre eles.

Um efeito colateral da aplicação de funções de Contração *Maxichoice* a Conjuntos de Crenças é que os mesmos se tornam muito grandes, conforme pode ser melhor compreendido pelo lema abaixo, reproduzido de [Gär88]:

**Lema 3.1** *Se $A \in K$ e $K_A^-$ é definida por uma função de Contração* Maxichoice, *então para qualquer proposição $B$, ou $A \vee B \in K_A^-$, ou $A \vee \neg B \in K_A^-$.*

Esse resultado decorre do fato de que como $K$ é um Conjunto de Crenças, o mesmo contem todas as tautologias. Se $A \in K$ então $B \rightarrow A \in K$ e $\neg B \rightarrow A \in K$ para todo $B \in L$. Quando $K$ é contraído por uma função do tipo Maxichoice, o resultado é um subconjunto maximal de $K$ que não deriva $A$. Dessa forma, ou $B \rightarrow A$ ou $\neg B \rightarrow A$ para todo $B \in L$ é mantido em $K_A^-$ (ou ambos, se $B \notin A$ e $\neg B \notin A$), garantido assim a consistência e a maximilidade, e portanto $\neg B \vee A \in K_A^-$ ou $B \vee A \in K_A^-$.

**Definição 3.3** *Conjuntos de Crenças Maximais*

*Um Conjunto de Crenças $K$ é dito maximal se para toda proposição $B$ de sua linguagem, ou $B \in K$ ou $\neg B \in K$.*

Segue-se como corolário do lema acima que se uma função de Revisão $*$ for definida, pela Identidade de Levi, a partir de uma função de Contração $-$ *Maxichoice*, então para qualquer $A$ tal que $\neg A \in K$, $K_A^*$ será maximal.

Seguindo o mesmo raciocínio anterior, como $\neg A \in K$, então $B \rightarrow \neg A \in K^-_{\neg A}$ ou $\neg B \rightarrow \neg A \in K^-_{\neg A}$ para todo $B \in L$, e consequentemente $A \rightarrow \neg B \in K^-_{\neg A}$ ou $A \rightarrow B \in K^-_{\neg A}$, para todo $B \in L$. Ao se expandir $K^-_{\neg A}$ por $A$, uma das duas implicações será mantida e portanto para todo $B \in L$, $B \in K^*_A$ ou $\neg B \in K^*_A$.

### 3.5.2 Funções de Contração *Full Meet*

As funções de Contração *Full Meet* consideram $K^-_A$ como a interseção de todos os subconjuntos maximais de $K$ que não derivam $A$, ou seja,

**Definição 3.4** *Função* FullMeet

$$K^-_A = \begin{cases} \bigcap(K_\perp A) & \textit{Se } K_\perp A \textit{ é não-vazio} \\ K & \textit{caso contrário} \end{cases}$$

O problema com essa função é que ela produz conjuntos muito restringidos, como pode ser visto através do seguinte lema, também presente em [Gär88]:

**Lema 3.2** *Se $K^-_A$ é definida por meio de uma função de Contração* Full Meet *e $A \in K$, então $B \in K^-_A$ sse $B \in K$, e $\neg A \vdash B$.*

Ou seja, só resta no conjunto as proposições que são conseqüências lógicas de $\neg A$. Segue-se como corolário desse lema que se uma função de Revisão $*$ for definida, pela Identidade de Levi, a partir de uma função de Contração – *Full Meet*, então para qualquer $A$ tal que $\neg A \in K$, $K^*_A$ conterá apenas $A$ e suas conseqüências lógicas.

### 3.5.3 Funções de Contração *Partial Meet*

O princípio básico dessa função é usar alguns dos subconjuntos maximais em $K_\perp A$ na definição de $K^-_A$. A função de Contração é definida da seguinte forma:

**Definição 3.5** *Função* Partial Meet

***(Part)*** $K^-_A = \bigcap S(K_\perp A)$

A função de seleção $S$ apanha apenas os subconjuntos maximais de $K$ em $K_\perp A$ que são mais epistemologicamente entrincheirados. Dessa forma, uma proposição $B$ está

em $K_A^-$ sse ele é um elemento de todos os subconjuntos maximais de $K$ mais entrincheirados. A noção de Entrincheiramento Epistêmico será abordada mais detalhadamente na próxima seção.

Para se determinar os subconjuntos de $K_\perp A$ mais epistemologicamente entrincheirados, assume-se que existe uma ordenação de Entrincheiramento Epistêmico de todos os subconjuntos maximais de $K$. Tal ordenação deve ser independente da sentença a ser retirada. $M(K)$ é usado para denotar a união da família de todos os conjuntos $K_\perp A$, onde $A$ é qualquer proposição em $K$ que não é logicamente válida. Em seguida, é assumido que existe uma relação $\leq$ em $M(K)$. Se $\nvdash A$, então $K_\perp A$ é não-vazio e a função de seleção $S$ é definida através daquela relação:

**(Def S)** $S(K_\perp A) = \{K' \in K_\perp A : K'' \leq K' \text{ para todo } K'' \in K_\perp A\}$.

Esta identidade é chamada de identidade *Marking-Off*.

## 3.6 Entrincheiramento Epistêmico

O conceito de Entrincheiramento Epistêmico está relacionado às sentenças em $L$. É ele quem determina o que acontece com elas quando o Conjunto de Crenças é contraído ou revisado. É possível determinar-se o Entrincheiramento Epistêmico relativo das sentenças num Conjunto de Crenças $K$ independentemente do que acontece a $K$ nas Revisões e Contrações. As sentenças abandonadas em $K$ são aquelas com menor Entrincheiramento Epistêmico.

O critério de determinação do Entrincheiramento Epistêmico de uma sentença está relacionado ao seu poder explanatório e seu valor sob o ponto de vista informacional no Conjunto de Crenças. De modo geral, pode-se afirmar que quanto mais difícil for retirar uma sentença, mais epistemologicamente entrincheirada ela é.

Se $A$ e $B$ são sentenças em $L$, $A \leq B$ indica que $B$ é pelo menos tão epistemologicamente entrincheirado quanto $A$. Se $B$ é mais epistemologicamente entrincheirado que $A$ então indica-se por $A \leq B$ e $B \not\leq A$.

**(C$\leq$)** $B \leq A$ sse $B \notin K^-_{A \wedge B}$

Pelo critério da economia informativa, apenas uma sentença de uma conjunção precisa ser retirada do conjunto a fim de realizar a sua Contração. A propriedade acima

afirma que se uma sentença $B$ não está presente na Revisão de Crenças de um conjunto com relação à conjunção $A \wedge B$, é porque a retirada de $B$ foi preferida em relação à retirada de $A$ e portanto $A$ é pelo menos tão epistemologicamente entrincheirada quanto $B$.

**Postulados para determinação do Entrincheiramento Epistêmico**

($EE1$) Para quaisquer $A$, $B$ e $C$, se $A \leq B$ e $B \leq C$, então $A \leq C$. **(transitividade)**

($EE2$) Para quaisquer $A$ e $B$, se $A \vdash B$, então $A \leq B$. **(dominância)**

($EE3$) Para quaisquer $A$ e $B$ em $K$, $A \leq A \wedge B$ ou $B \leq A \wedge B$. **(conjuntividade)**

($EE4$) Quando $K \neq K_{\perp}$, $A \notin K$ sse $A \leq B$ para todo $B$. **(minimalidade)**

($EE5$) Se $B \leq A$ para todo $B$, então $\vdash A$.

O primeiro postulado assegura a transitividade da relação. O segundo afirma que se $B$ é uma conseqüência lógica de $A$ então $B$ é pelo menos tão epistemologicamente entrincheirado quanto $A$ (pois a retirada de $B$ implicaria necessariamente na retirada de $A$). ($EE3$) determina que qualquer conjunção é pelo menos tão epistemologicamente entrincheirada quanto qualquer um dos seus componentes e possibilitada a retirada de apenas um dos membros da mesma quando sua Contração é necessária. ($EE4$) é usado para generalizar a relação $\leq$ a todas as sentenças da linguagem, considerando as sentenças que não estão no conjunto como minimais em $\leq$. Similarmente, ($EE5$) determina que as sentenças logicamente válidas são maximais em $\leq$.

Uma ordenação $\leq$ que satisfaça ($EE1$) a ($EE3$) tem a propriedade de que para quaisquer $A$ e $B$, ou $A \leq B$ ou $B \leq A$ (conectividade). Segue-se de ($EE4$) que as sentenças mais epistemologicamente entrincheiradas são as logicamente válidas, pois essas nunca são descartadas. A recíproca de ($EE5$) segue de ($EE2$), pois se $\vdash A$ então $\vdash B \rightarrow A$ para todo $B$.

## 3.7 Contrações Seguras

O processo de Contração Segura foi proposto por [AM85] e consiste em:

Seja $K$ um Conjunto de Crenças e suponha que se deseje contraí-lo em relação a uma proposição $A$. Suponha que haja uma hierarquia $<$ em $K$ acíclica. Diz-se que um elemento $B$ é seguro em relação a $A$ sse $B$ não é um elemento mínimo (sob $<$) de nenhum

subconjunto minimal $K'$ de $K$ tal que $K' \vdash A$. Ou seja, todo subconjunto minimal $K'$ de $K$ tal que $K' \vdash A$ ou não contem $B$ ou então contem alguma sentença $C$ tal que $C < B$. A idéia por trás disso é que um elemento $B$ é seguro em relação a $A$, se não puder ser *culpado* por sua implicação. A hierarquia $<$ sobre $K$ pode ser vista como a ordenação de Entrincheiramento Epistêmico.

$K/A$ é o conjunto de todos os elementos de $K$ seguros com relação a $A$. A Contração segura de $K$ por $A$ é definida como $K_A^- = Cn(K/A)$.

## 3.8 Aplicações

As técnicas aqui apresentadas podem ser úteis em áreas como atualização de Bancos de Dados Lógicos e Códigos Legais.

### 3.8.1 Bancos de Dados Lógicos

Um Banco de Dados pode ser representado como um modelo de um estado epistêmico se pensarmos nos seus itens componentes como proposições ou sentenças. As modificações feitas por um usuário são então visualizadas como mudanças de crenças. Ao se remover um item pode-se alterar a consistência do Banco de Dados, pois o mesmo é visto como um conjunto de fatos a partir dos quais pode-se derivar outros.

Usando-se uma ordenação de Entrincheiramento Epistêmico, as atualizações podem ser efetuadas de modo seguro. [FUV83] introduziram a noção de sentença *marcada*, que consiste de um par $< i, A >$ onde $A$ é uma sentença e $i$ um número natural representando a prioridade de $A$ no Banco de Dados. Quanto menor o índice $i$, maior a prioridade. Ao se atualizar o Banco de Dados, as sentenças com índices maiores são removidas primeiro.

Outro modo de modelar as atualizações foi proposto por [FR86]. O Banco de Dados é construído como um conjunto de regras em PROLOG no qual certas sentenças são consideradas como informação negativa (que se sabe serem falsas). Essas sentenças são usadas para derivar proposições negadas (ao invés de usar negação por falha finita). Eles usam um conceito denominado *strata* de informação que é bastante similar ao de Entrincheiramento Epistêmico.

### 3.8.2 Códigos Legais

Um Código Legal pode também ser visto como um Conjunto de Crenças pois consiste de um conjunto de proposições e de suas conseqüências lógicas.

As Contrações aqui são denominadas de derrogação de código, as Revisões de emendas, e as Expansões são simplesmente as adições de novas leis. Os mesmos problemas lógicos encontrados nas Revisões e Contrações de Conjuntos de Crenças são encontrados aqui e podem ser resolvidos utilizando-se de uma ordenação de *Entrincheiramento Deôntico* juntamente com as técnicas apresentadas. æ

# Capítulo 4

# PROGRAMAÇÃO EM LÓGICA E CLÁUSULAS DE HORN

## 4.1 Introdução

O primeiro grande marco da história da Programação em Lógica pode ser considerado o trabalho desenvolvido em [Her67]. A partir desse trabalho surgiram outros de grande relevância ao campo da prova automática de teoremas, como a introdução da regra da Resolução em 1965 por Robinson ([Rob65]). A Importância dessa regra deve-se ao fato de a mesma ser especialmente adequada para implementação em computadores.

Posteriormente, Kowalski ([Kow74]) e Colmerauer ([CKRP73]), sugeriram o uso de Lógica como uma linguagem de programação culminando com o desenvolvimento do primeiro interpretador PROLOG em Marseille em 1972.

Esse capítulo faz uma breve revisão de literatura sobre programação em cláusulas de Horn. Uma análise mais aprofundada sobre o assunto pode ser encontrada em [CGF87], que possui uma excelente revisão de Lógica de Primeira Ordem e Prova Automática de Teoremas, e também em [Llo84], onde a maior parte desse texto é baseada.

A seguir será apresentada a Linguagem da Lógica de Primeira Ordem. Posteriormente, a Linguagem das Cláusulas será introduzida como uma classe especial de fórmulas dessa linguagem.

## 4.2 Linguagem da Lógica de Primeira Ordem

A linguagem da Lógica de Primeira Ordem é definida a partir de um conjunto de objetos distintos que constituem o seu alfabeto.

DEFINIÇÃO 4.1 :

***Alfabeto***

- *Um conjunto infinito de variáveis;*
- *Um conjunto de constantes;*
- *Um conjunto de símbolos funcionais n-ários;*
- *Um conjunto de símbolos predicativos n-ários;*
- *Símbolos Lógicos:* $\neg$, $\wedge$, $\vee$, $\rightarrow$ *e* $\leftrightarrow$*;*
- *Quantificadores:* $\forall$ *e* $\exists$*;*
- *Símbolos de Pontuação:* (, ) *e* ,.

***Símbolos da Linguagem***

- *termo: é uma variável, uma constante do alfabeto, ou uma expressão da forma* $f(t_1, t_2, ..., t_n)$*, onde* $f$ *é um símbolo funcional n-ário e* $t_1$,$t_2$,...,$t_n$ *são termos do alfabeto.*
- *fórmulas bem-formadas* (wffs)*:*

  *O conjunto das fórmulas bem-formadas, ou simplesmente fórmulas, sobre um alfabeto de primeira ordem é o menor conjunto que satisfaz às seguintes condições:*

  *i) Se p é um símbolo predicativo n-ário e* $t_1$,$t_2$,...,$t_n$ *são termos, então* $p(t_1,t_2,...,t_n)$ *é uma fórmula, chamada de* Fórmula Atômica *ou* Átomo.

  *ii) Se* $F$ *e* $G$ *são fórmulas, então* $\neg F$, $F \wedge G$, $F \vee G$, $F \rightarrow G$ *e* $F \leftrightarrow G$ *também são fórmulas.*

  *iii) Se* $F$ *é uma fórmula e* $x$ *é uma variável, então* $\forall x F$ *e* $\exists x F$ *são fórmulas.*

DEFINIÇÃO 4.2 :

*A* Linguagem de Primeira Ordem *sobre um alfabeto é constituída por todas as fórmulas construídas a partir de símbolos desse alfabeto.*

*O* Escopo *de* $\forall x$ *(ou* $\exists x$*) em* $\forall xF$ *(ou* $\exists xF$*) é* $F$*.*

*Uma ocorrência de variável é dita* Ligada *em uma fórmula se essa ocorrência estiver no escopo de algum quantificador seguido imediatamente por aquela variável. Caso contrário, a ocorrência é dita* Livre.

*Uma* Fórmula Fechada *é uma fórmula sem a ocorrência de variáveis livres.*

*O* Fecho Universal *de uma fórmula* $F$ *é obtido adicionando um quantificador universal ao início da mesma para cada variável com ocorrência livre em* $F$*. O* Fecho Existencial *de* $F$ *é obtido similarmente com quantificadores existenciais.*

DEFINIÇÃO 4.3 :

*Sejam* $E$ *e* $E'$ *duas seqüências de símbolos de um alfabeto de primeira ordem* $A$*. Suponha que as variáveis de* $A$ *sejam ordenadas* $x_1, x_2, \ldots$ .

*a)* $E'$ *é uma* variante *de* $E$ *sse existe uma renomeação* $\beta$ *tal que* $E' = E\beta$*.*

*b)* $E'$ *é a* variante canônica *de* $E$ *sse:*

*i)* $E'$ *é uma variante de* $E$

*ii) existe* $n \geq 0$ *tal que as variáveis ocorrendo em* $E'$ *são* $x_1, x_2, \ldots, x_n$

*iii) para todo* $i, j \in [1, n]$*, a ocorrência de* $x_i$ *mais à esquerda em* $E'$ *precede a ocorrência de* $x_j$ *mais à esquerda em* $E'$ *sse* $i < j$*.*

## 4.3 Linguagem das Cláusulas de Horn

Uma classe especial de fórmulas da Linguagem de Primeira Ordem apresentada anteriormente possui propriedades bastante interessantes sobretudo sob o aspecto computacional. Essas fórmulas possibilitam a definição de uma nova Linguagem formal, conhecida por Linguagem de Cláusulas, cujos símbolos são introduzidos a seguir.

DEFINIÇÃO 4.4 :

*Um* Literal *é uma fórmula atômica ou sua negação. Um* Literal Positivo *é uma fórmula atômica e um* Literal Negativo *é a negação de uma fórmula atômica.*

*Uma* Cláusula *é uma fórmula fechada do tipo* $\forall x_1 \ldots \forall x_m(L_1 \vee L_2 \vee \ldots \vee L_n)$, *onde cada* $L_i$ *é um literal e* $x_1 \ldots x_m$ *são todas as variáveis que ocorrem em* $L_1 \vee L_2 \vee \ldots \vee L_n$.

Uma cláusula do tipo

$$\forall x_1 \ldots \forall x_m(C_1 \vee C_2 \vee \ldots \vee C_j \vee \neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_k)$$

onde $C_1 \vee C_2 \vee \ldots \vee C_j$ são todos literais positivos e $\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_k$ são todos literais negativos, numa notação especial chamada de *Notação Clausal* será representada pela expressão

$$C_1, C_2, \ldots, C_j \leftarrow A_1, A_2, \ldots, A_k$$

O lado direito da expressão é chamado de *Antecedente* e representa uma conjunção de literais. O lado esquerdo é chamado de *Conseqüente* e representa uma disjunção de literais. Como uma cláusula é uma fórmula fechada, os quantificadores universais são omitidos.

Note que

$$\forall x_1 \ldots \forall x_m(C_1 \vee C_2 \vee \ldots \vee C_j \vee \neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_k)$$

é equivalente a

$$\forall x_1 \ldots \forall x_m(C_1 \vee C_2 \vee \ldots \vee C_j \vee \neg(A_1 \wedge A_2 \wedge \ldots \wedge A_k))$$

que por sua vez equivale a

$$\forall x_1 \ldots \forall x_m((A_1 \wedge A_2 \wedge \ldots \wedge A_k) \rightarrow (C_1 \vee C_2 \vee \ldots \vee C_j))$$

sendo portanto a notação intuitivamente correta. A partir dessa notação é possível definir formalmente a Linguagem das Cláusulas de Horn, que é constituída por cláusulas obedecendo a certas restrições de formação.

DEFINIÇÃO 4.5 : *Linguagem das Cláusulas de Horn*

*Uma* Cláusula Definida *é uma expressão da forma*

$$C \leftarrow A_1, A_2, \ldots, A_m \quad (m \geq 0)$$

*onde* $C, A_1, A_2, \ldots, A_m$ *são literais positivos.*

*Quando uma cláusula definida não possui nenhum literal no antecedente, isto é, uma cláusula do tipo* $C \leftarrow$ *a mesma é dita* Incondicional *e chamada de* Fato. *O significado pretendido para um fato* $C$ *é que para qualquer atribuição de todas as variáveis aparecendo em* $C$, $C$ *é verdadeiro. Quando o contexto não suscitar dúvida, um fato* $C \leftarrow$ *será representado simplesmente por* $C$.

*Quando uma cláusula definida possui um ou mais literais no antecedente, a mesma é dita* Condicional *e chamada de* Regra. *A cláusula*

$$C \leftarrow A_1, A_2, \ldots, A_n \tag{4.1}$$

*representa a noção de que para qualquer atribuição de todas as variáveis ocorrendo em (4.1), se* $A_1, A_2, \ldots, A_n$ *são todos verdadeiros, então* $C$ *é verdadeiro.*

*Uma* Cláusula Vazia, *denotada por* $\square$, *possui o antecedente e o conseqüente vazios e representa uma contradição.*

*Uma* Cláusula Objetivo *é uma cláusula que não possui nenhum literal no conseqüente. Segue que uma cláusula vazia é também uma cláusula objetivo.*

*Uma* Cláusula de Horn *é uma cláusula definida ou uma cláusula objetivo.*

*Um* Conjunto Definido *é um conjunto formado somente por cláusulas definidas.*

*Um* Conjunto quase-Definido *é um conjunto formado por cláusulas definidas e uma cláusula objetivo.*

*A* Definição de um Predicado $p$ *num Conjunto Definido é o conjunto formado por todas as cláusulas com* $p$ *no conseqüente.*

*A* Linguagem das Cláusulas de Horn *sobre um alfabeto de primeira ordem é o conjunto de todas as Cláusulas de Horn sobre esse alfabeto.*

A partir das definições dadas acima, é possível verificar que uma cláusula objetivo do tipo

$$\leftarrow M_1, M_2, \ldots, M_n$$

representa a expressão

$$\forall x_1, x_2, \ldots, x_m(\neg M_1 \vee \neg M_2 \vee \ldots \vee \neg M_n)$$

que por sua vez é equivalente a

$$\begin{gathered}
\forall x_1, x_2, \ldots, x_m \neg(M_1 \wedge M_2 \wedge \ldots \wedge M_n) \\
\vdots \\
\neg\exists x_1 \neg\neg\exists x_2 \neg \ldots \neg\exists x_m \neg\neg(M_1 \wedge M_2 \wedge \ldots \wedge M_n) \\
\vdots \\
\neg\exists x_1 \exists x_2 \ldots \exists x_m (M_1 \wedge M_2 \wedge \ldots \wedge M_n)
\end{gathered}$$

sendo portanto as variáveis de uma Cláusula Objetivo implicitamente quantificadas existencialmente.

As variáveis de uma cláusula serão denotadas por letras maiúsculas do final do alfabeto e as constantes por letras minúsculas do início do mesmo.

A definição da semântica das Cláusulas de Horn estabelece formalmente a correspondência entre a Linguagem destas e a Linguagem das Claúsulas:

DEFINIÇÃO 4.6 : *Satisfatibilidade das Cláusulas de Horn*

*Sejam A um alfabeto de primeira ordem e I uma estrutura para A.*

*i) I satisfaz uma Cláusula Definida* $C \leftarrow A_1, A_2, \ldots, A_n$ *sse I satisfaz a Cláusula* $\forall x_1, \forall x_2, \ldots, \forall x_m\ (C \vee \neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n)$, *onde* $x_1, x_2, \ldots, x_m$ *são todas as variáveis que ocorrem nesta.*

*ii) I satisfaz uma Cláusula Objetivo não vazia* $\leftarrow A_1, A_2, \ldots, A_n$ *sse I satisfaz a Cláusula* $\forall x_1, \forall x_2, \ldots, \forall x_m\ (\neg A_1 \vee \neg A_2 \vee \ldots \vee \neg A_n)$, *onde* $x_1, x_2, \ldots, x_m$ *são todas as variáveis que ocorrem nesta.*

*iii) I não satisfaz a Cláusula vazia.*

## 4.4 Resolução-LSD

O nome desse método de refutação deve-se ao fato de o mesmo ser baseado no método de **Resolução** **L**inear de Entrada, com função de **S**eleção, restrito a conjuntos quase-**D**efinidos.

A Resolução-LSD, apesar do nome, é efetivamente um método de refutação por eliminação de modelos. Aqui se trabalham com cadeias de literais, ao invés de cláusulas propriamente ditas e a seleção de literais a resolver é determinada por uma função de seleção.

Como os conjuntos de entrada são quase-definidos, não é necessário manter-se os literais resolvidos a fim de se atingir a completude, dispensando portanto as regras de Redução e Contração utilizadas no método de Eliminação de Modelos.

DEFINIÇÃO 4.7 : *Função de seleção para Cláusulas Objetivo*

*Uma* Função de Seleção $f$ *para cláusulas objetivo é aquela que mapeia cada cláusula objetivo* $G$ *em um dos literais de* $G$. *Se este for o literal mais à esquerda de* $G$, $f$ *é dita* Função de Seleção Padrão.

DEFINIÇÃO 4.8 :

*Uma* Renomeação para uma Cláusula $C$ em presença de uma Cláusula $C'$ *é uma renomeação de variáveis* $\beta$ *tal que* $C'$ *e* $C\beta$ *não possuem variáveis em comum.*

A única regra de inferência utilizada neste método é a $f$-extensão, que é a regra da Extensão do método da Eliminação de Modelos, adaptada para levar em consideração a função de seleção $f$.

DEFINIÇÃO 4.9 : *Regra da* $f$*-extensão*

*Sejam* $G$ *uma cláusula objetivo,* $D$ *uma cláusula definida e* $\beta$ *uma renomeação de* $D$ *em presença de* $G$.

*Se* $G$ *é da forma* $\leftarrow g_1,g_2,\ldots,g_n$, $D\beta$ *da forma* $q \leftarrow a_1,a_2,\ldots,a_k$ *e* $f(G) = g_i$, *então o resultado da aplicação de uma* $f$*-extensão de* $G$ *por* $D$ *é uma nova cláusula objetivo* $G'$ *sse existir um u.m.g.* $\theta$ *do conjunto de literais* $\{q, g_i\}$ *de forma que* $G'$ *é igual a* $(g_1,g_2,\ldots,g_{i-1}, a_1,a_2,\ldots,a_k, g_{i+1},\ldots,g_n)\theta$.

Através da regra de inferência definida acima, é possível então introduzir o conceito de LSD($f$)-dedução e de LSD($f$)-refutação.

DEFINIÇÃO 4.10 : *LSD(f)-dedução e LSD(f)-refutação*

***LSD(f)-dedução***

*Sejam $f$ uma função de seleção, $S$ um conjunto quase-definido e $G$ uma cláusula objetivo. Uma* LSD($f$)-dedução *de $G$ a partir de $S$ é uma seqüência $C = (C_1, C_2, \ldots, C_n)$ de cláusulas, terminando em $G$, isto é, $C_n = G$ e $\exists r \leq n$ tal que:*

*i) para todo $i < r$, $C_i$ é uma cláusula definida pertencente a $S$;*

*ii) $C_r$ é a cláusula objetivo de $S$;*

*iii) para todo $i > r$, $C_i$ foi obtido pela aplicação da regra da $f$-extensão a $C_{i-1}$ e $C_j$, para algum $j < r$.*

*A seqüência $C_{r+1}, C_{r+2}, \ldots, C_n$ é chamada de* Sufixo da Derivação *de $G$.*

***LSD($f$)-refutação***

*Uma* LSD($f$)-refutação *a partir de um conjunto quase-definido $S$ é uma LSD($f$)-dedução da cláusula vazia a partir de $S$. O sufixo da derivação nesse caso é chamdo de* Sufixo da Refutação.

DEFINIÇÃO 4.11 : *Árvore de Refutação por Resolução-LSD($f$)*

*Uma* Árvore de Refutação por Resolução-LSD($f$), *ou simplesmente árvore de LSD($f$)-refutação, é construída da seguinte forma:*

- *o rótulo da raiz é a cláusula objetivo do conjunto quase-definido.*
- *para cada nó $N$, o rótulo de $N$ é uma cláusula objetivo e os filhos obtidos pela regra da $f$-extensão usando como cláusula auxiliar cada uma das cláusulas definidas de entrada cuja cabeça seja unificável com o literal selecionado por $f$ da cláusula que rotula $N$.*

*Um* Nó de Sucesso *numa árvore de LSD($f$)-refutação é uma nó rotulado pela cláusula vazia.*

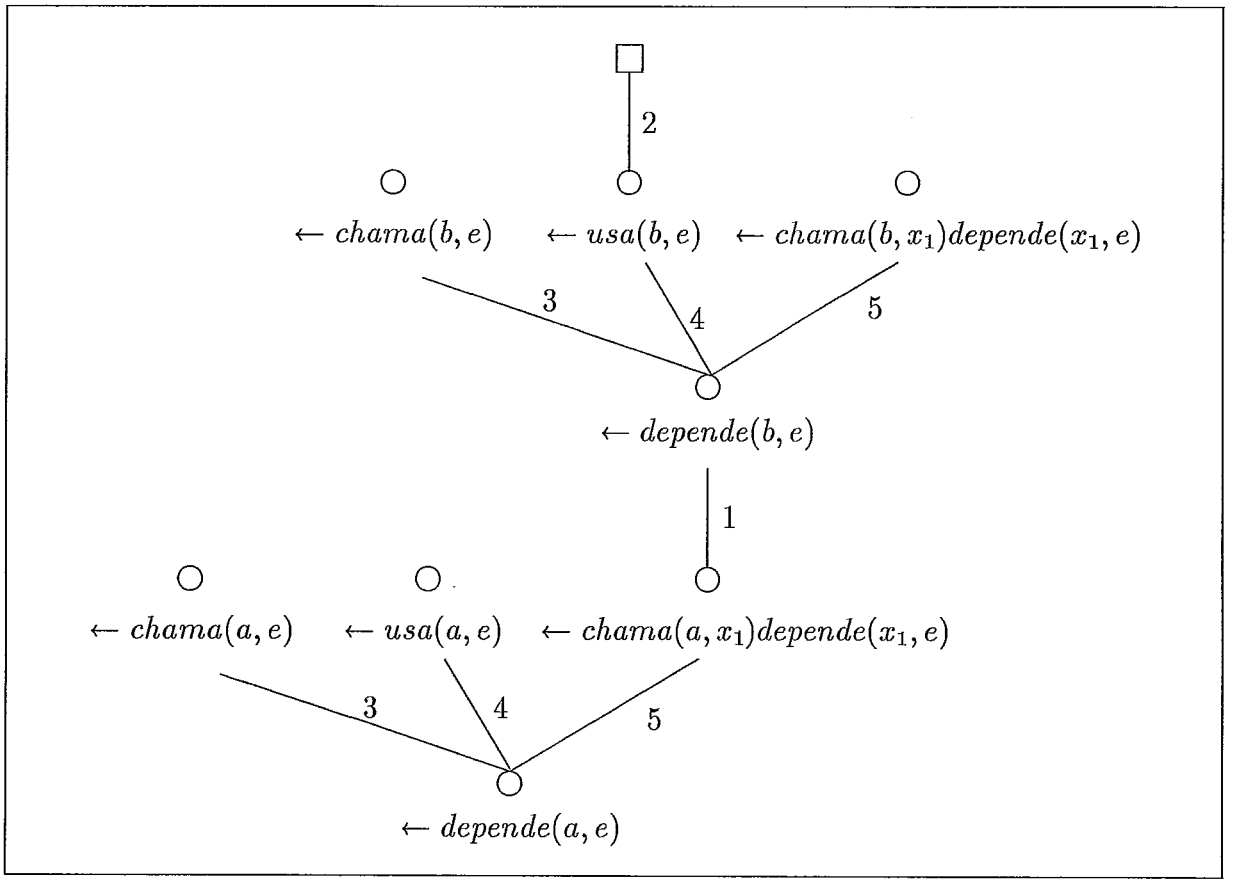
*Um* Ramo de Sucesso *numa árvore de LSD($f$)-refutação é um ramo que termina num nó de sucesso.*

EXEMPLO 4.1 : *Uma árvore de LSD($f$)-refutação ([CGF87])*

***Conjunto quase-definido***

1. $chama(a, b)$
2. $usa(b, e)$
3. $depende(X, Y) \leftarrow chama(X, Y)$
4. $depende(X, Y) \leftarrow usa(X, Y)$
5. $depende(X, Y) \leftarrow chama(X, Z), depende(Z, Y)$
6. $\leftarrow depende(a, e)$

***Árvore de LSD(f)-refutação***




Se uma árvore de LSD($f$)-refutação que começa na cláusula objetivo de um conjunto quase-definido $S$ possuir um nó de sucesso, então $S$ é insatisfatível.

DEFINIÇÃO 4.12 : *Árvore de prova por Resolução-LSD*

*Uma árvore $\mathcal{A}$ é uma* Árvore de Prova por Resolução-LSD *para um literal $l$ (ou simplesmente uma árvore de LSD-prova), a partir de um Conjunto Definido $S$, se:*

*i) a raiz de $\mathcal{A}$ for o literal $l$, e*

*ii) para cada nó $N$, rotulado por $l'$, existe uma instância $C'$ de uma cláusula $C$ de $S$ da forma*

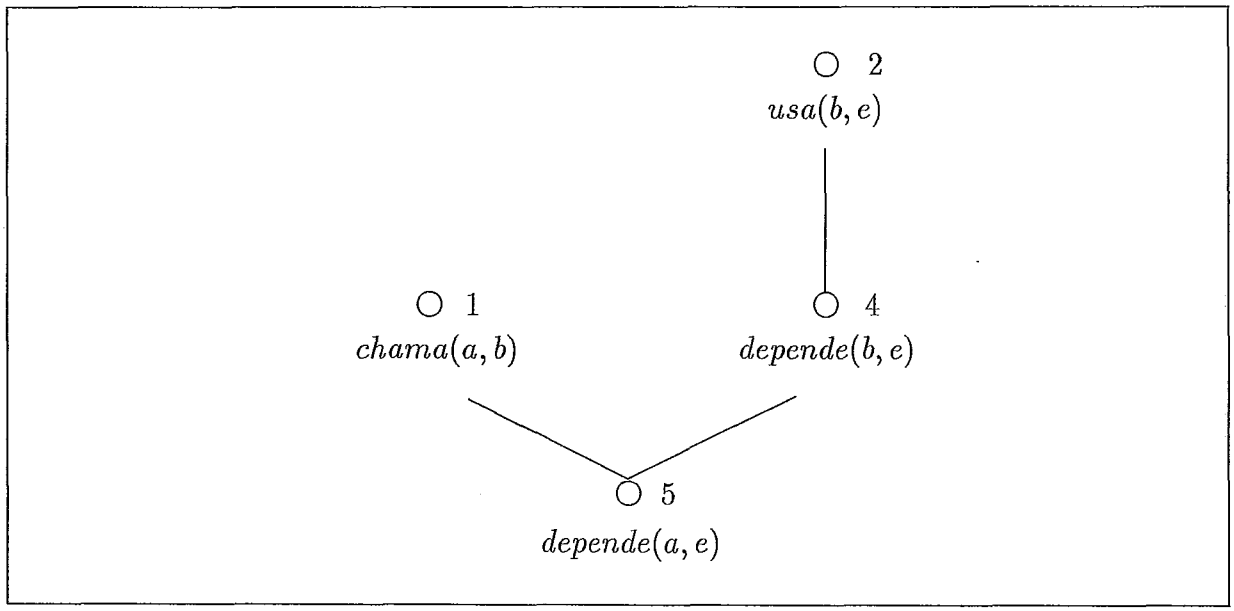
$$l' \leftarrow A_1, A_2, \ldots, A_n \quad (n \geq 0)$$

*e os filhos de $N$ são, respectivamente da esquerda para a direita, árvores de prova dos literais $A_1$, $A_2$, ..., $A_n$. Se a cláusula $C$ for um fato, o nó $N$ não terá filhos.*

*O conjunto de todas a árvores de LSD-prova para um literal a partir de um conjunto definido é chamado de* Floresta de Prova *desse literal.*

EXEMPLO 4.2 : *Árvore de LSD-prova*

*Considerando o mesmo conjunto definido do exemplo anterior, a árvore de LSD-prova para o literal depende$(a, e)$ é a seguinte:*




É possível mapear uma árvore $\mathcal{A}$ de LSD-prova de uma literal $l$ a partir de um Conjunto Definido $S$, no ramo de sucesso da árvore $\mathcal{A}'$ de LSD($f$)-refutação de $S \bigcup \{\leftarrow l\}$, da seguinte forma:

- a raiz de $\mathcal{A}'$ é a cláusula $\leftarrow l$;
- para cada nó $N'$ de $\mathcal{A}'$, se o rótulo de de $N'$ é a cláusula $C$ da forma

$\leftarrow A_1, A_2, \ldots, A_n$

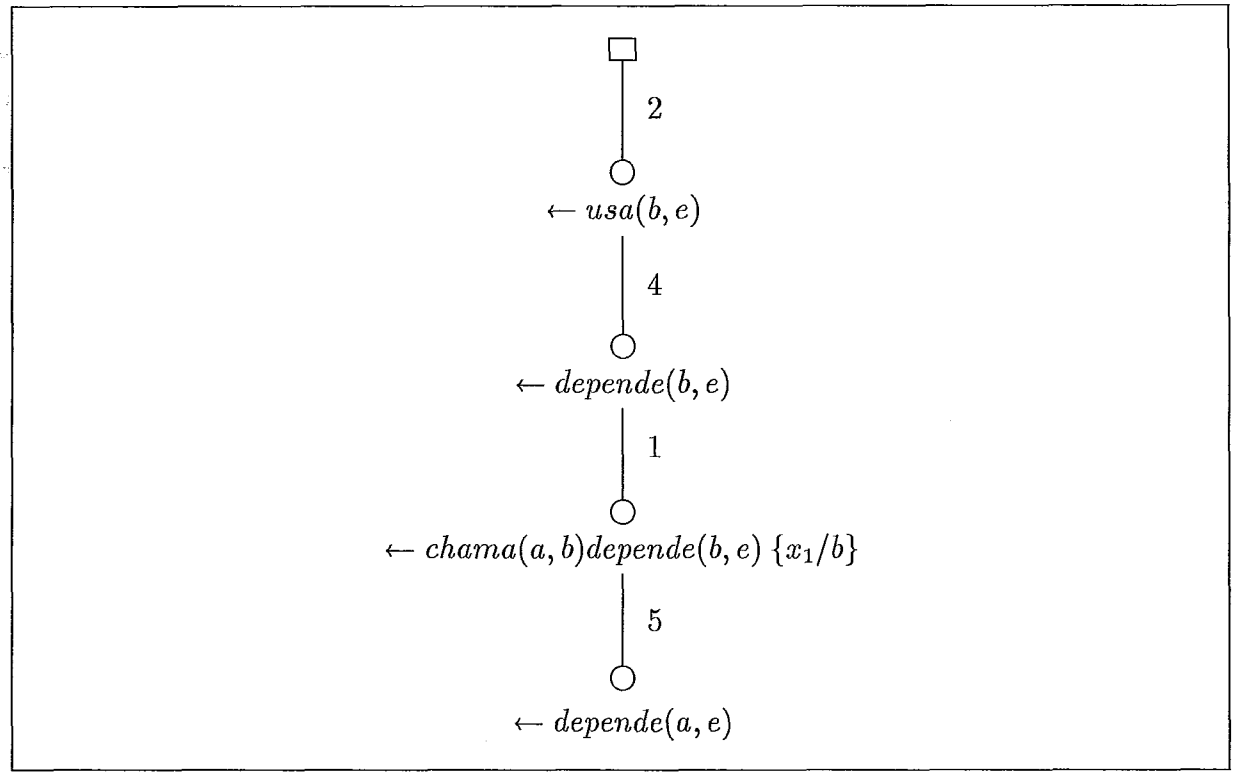
e $f(C) = A_i$, o filho de $N'$ no ramo de sucesso é a cláusula

$\leftarrow A_1, A_2, \ldots, A_{i-1}, B_1, B_2, \ldots, B_m, A_{i+1}, \ldots, A_n$

onde $B_1$, $B_2$, ..., $B_m$ são os filhos da raiz da sub-árvore de prova de $A_i$ em $\mathcal{A}$.

EXEMPLO 4.3 : *Mapeamento de uma árvore de LSD-prova num ramo de sucesso da árvore de LSD(f)-refutação correspondente*

*A árvore abaixo é um mapeamento da árvore de LSD-prova do Exemplo 4.4 no ramo de sucesso da árvore de LSD(f)-refutação do Exemplo 4.4.*




Como conseqüência imediata das definições acima, tem-se que uma floresta de prova de um literal $l$ a partir de um conjunto definido $S$ pode ser mapeada em todos os ramos de sucesso de uma árvore de LSD($f$)-refutação de $S \cup \{\leftarrow l\}$.

DEFINIÇÃO 4.13 : *Substituições resposta*

*Sejam S um conjunto definido e C uma cláusula objetivo.*

*a)* *Uma* Substituição Resposta *para* $S \bigcup \{C\}$ *é uma substituição para as variáveis de* $C$.

*b)* *Se* $C$ *é da forma*

$$\leftarrow A_1, A_2, \ldots, A_k,$$

$\theta$ *é uma* Substituição Resposta Correta *para* $S \bigcup \{C\}$ *se* $\forall((A_1 \wedge A_2 \wedge \ldots \wedge A_k)\theta)$ *é uma conseqüência lógica de* $S$.

*c)* *Se* $f$ *é uma função de seleção, uma* Substituição Resposta via $f$ *para* $S \bigcup \{C\}$ *é a substituição obtida restringindo a composição* $\theta_1, \theta_2, \ldots, \theta_n$ *às variáveis de* $C$, *onde* $\theta_1, \theta_2, \ldots, \theta_n$ *é a seqüência de umgs usados em uma LSD(*$f$*)-refutação de* $S \bigcup \{C\}$

### 4.4.1 Correção e Completude do Método de Resolução-LSD

A Correção da Resolução-LSD conforme demonstrada abaixo é encontrada em [Cla] e reproduzida em [Llo84]. Um tratamento alternativo pode ser encontrado em [CGF87] e visualiza a Resolução-LSD como um refinamento do método da Resolução. As LSD($f$)-deduções são mapeadas em R-deduções e a Correção é então derivada da Correção do método da Resolução.

**Teorema 4.1** *Seja* $P$ *um conjunto definido,* $G$ *uma cláusula objetivo e* $f$ *uma função de seleção. Toda substituição resposta via* $f$ *para* $P \bigcup \{G\}$ *é uma substituição resposta correta.*

**Prova:** Seja $G$ a cláusula $\leftarrow A_1, A_2, \ldots, A_k$ e $\theta_1, \theta_2, \ldots, \theta_n$, a seqüência de umgs usados na LSD($f$)-refutação de $P \bigcup \{G\}$. Vamos provar por indução no comprimento da refutação ($N$) que $\forall((A_1 \wedge A_2 \wedge \ldots \wedge A_k)\theta_1, \theta_2, \ldots, \theta_n)$ é uma substituição resposta correta.

**Base:** $N = 0$. Como a refutação tem tamanho 1, é porque $G$ é da forma $\leftarrow A_1$, $P$ tem um fato da forma $A \leftarrow$, e $A\theta_1 = A_1\theta_1$. Como $A_1\theta_1$ é uma instância de uma cláusula em $P$, segue que $\forall(A_1\theta_1)$ é uma conseqüência lógica de $P$.

Suponha que a proposição seja verdadeira para refutações de tamanho $N - 1$.

Sejam $A \leftarrow B_1, B_2, \ldots, B_q$ $(q \geq 0)$ a primeira cláusula de entrada usada nessa refutação e $f(G) = A_i$. Pela Hipótese de Indução,

$$\forall((A_1 \wedge \ldots \wedge A_{i-1} \wedge B_1 \wedge \ldots \wedge B_q \wedge A_{i+1} \wedge \ldots \wedge A_k)\theta_1,\theta_2,\ldots,\theta_n)$$

é uma conseqüência lógica de $P$ pois há uma refutação de

$$P \bigcup \{\leftarrow A_1,\ldots,A_{i-1},B_1,\ldots,B_q,A_{i+1},\ldots,A_k\}$$

de tamanho $N-1$. Dessa forma, $\forall((B_1 \wedge \ldots \wedge B_q)\theta_1,\theta_2,\ldots,\theta_n)$ é uma conseqüência lógica de $P$ e conseqüentemente $\forall(A_m\theta_1,\theta_2,\ldots,\theta_n)$, pois

$$\forall(A_m\theta_1,\theta_2,\ldots,\theta_n) = \forall(A\theta_1,\theta_2,\ldots,\theta_n)$$

é uma conseqüência lógica de $P$.

Logo, $\forall((A_1,A_2,\ldots,A_k)\theta_1,\theta_2,\ldots,\theta_n)$ é uma conseqüência lógica de $P$.

□

**Corolário 4.1** *Seja $P$ um conjunto definido, $G$ uma cláusula objetivo e $f$ uma função de seleção. Se existe uma LSD($f$)-refutação de $P \bigcup \{G\}$ então $P \bigcup \{G\}$ é insatisfatível.*

**Prova:** Seja $G$ da forma $\leftarrow A_1,A_2,\ldots,A_k$ e $\theta$ a substituição resposta via $f$ obtida na LSD($f$)-refutação de $P \bigcup \{G\}$. Pelo Teorema anterior, $\theta$ é correta e portanto $\forall((A_1 \wedge \ldots \wedge A_k)\theta)$ é uma conseqüência lógica de $P$. Logo $P \bigcup \{G\}$ é insatisfatível.

□

O Teorema abaixo estabelece a Completude do método de Resolução-LSD e foi inicialmente provado por Hill ([Hil74]). Posteriormente, foi demonstrado que a Completude do método independe da função de seleção. A prova dos dois Teoremas requer o estabelecimento de resultados intermediários e não será apresentada aqui. Para uma descrição completa dos resultados veja [Llo84].

**Teorema 4.2 (Teorema da Completude da Resolução-LSD)** *Sejam $S$ um Conjunto Definido e $G$ uma cláusula objetivo. Se $S \bigcup \{G\}$ for insatisfatível então existe uma função de seleção $f$ e uma LSD($f$)-refutação de $S \bigcup \{G\}$ via $f$.*

**Teorema 4.3 (Independência da função de seleção)** *Sejam $S$ um Conjunto Definido e $G$ uma cláusula objetivo e $f$ uma função de seleção. Suponha que existe uma LSD($f$)-refutação de $S \bigcup \{G\}$ via $f$. Seja $f'$ uma função de seleção qualquer. Então existe uma LSD($f$)-refutação de $S \bigcup \{G\}$ via $f'$. Além disso, se $\sigma$ e $\sigma'$ são as respectivas substituições respostas obtidas, então $G\sigma$ é uma variante de $G\sigma'$.*

## 4.5 Resolução-LSDNF

Com a restrição de que os conjuntos de entrada da Resolução-LSD sejam conjuntos quase-definidos ganhou-se grande simplicidade e eficiência para o método. Tal simplicidade contudo, causou severas limitações. Uma simples disjunção do tipo $p(a) \vee p(b)$ não pode pertencer a um conjunto de entrada, pois o mesmo é por definição um Conjunto Definido.

Para minimizar esses efeitos, o método da Resolução-LSD foi estendido através de um processo conhecido por Negação por Falha Finita, baseada na Hipótese de Mundo Fechado. A idéia por trás do novo método é inferir um literal negativo a partir de um conjunto de cláusulas, quando uma prova para o literal positivo correspondente não puder ser obtida.

As definições dadas anteriormente devem ser adaptadas para levar em consideração o fato de que agora é possível a presença de literais negativos no corpo das regras.

DEFINIÇÃO 4.14 : *Linguagem das Cláusulas pseudo-Horn*

*Uma* Cláusula pseudo-Definida *é uma expressão da forma*

$$C \leftarrow A_1, A_2, \ldots, A_n$$

*onde* $C$ *é um literal positivo, e* $A_1$, $A_2$, ..., $A_n$ *são literais.*

*Uma* Cláusula pseudo-Objetivo *é ou a Cláusula Vazia ou uma expressão da forma*

$$\leftarrow A_1, A_2, \ldots, A_n$$

*onde* $A_1$, $A_2$, ..., $A_n$ *são literais.*

*Uma* Cláusula pseudo-Horn *é ou uma Cláusula pseudo-Definida ou uma Cláusula pseudo-Objetivo.*

*Um* Conjunto pseudo-Definido *é um conjunto formado por apenas Cláusulas pseudo-Definidas.*

*Um* Conjunto quase-pseudo-Definido *é um Conjunto pseudo-Definido acrescentado de uma Cláusula pseudo-Objetivo.*

*A* Linguagem das Cláusulas pseudo-Horn *sobre um alfabeto de primeira ordem é o conjunto de todas as Cláusulas pseudo-Horn sobre esse alfabeto.*

Uma conseqüência imediata do fato de que o antecedente das cláusulas possam ter literais negativos é que agora é possível também tê-los nas cláusulas derivadas. O método da Resolução-LSD precisa então ser estendido para que a regra da $f$-extensão possa tratar o caso em que o literal escolhido pela função de seleção seja negativo. Isso é feito com a introdução de uma nova regra chamada de Negação por Falha Finita. Essa regra estabelece que se o literal $l$ selecionado a partir de uma cláusula objetivo $C$ for básico (sem a ocorrência de variáveis) e negativo e for possível mostrar de forma finitária que $\leftarrow l$ não poder ser refutado a partir do conjunto pseudo-Definido em questão, então o literal $l$ pode ser cancelado em $C$.

É possível mostrar de forma finitária que $\leftarrow l$ não pode ser refutado a partir de $S$ se a árvore de refutação por resolução LSDNF($f$) a partir de $S \bigcup \{\leftarrow l\}$, com raiz rotulada por $\leftarrow l$, for finita e não tiver nenhum nó de sucesso. A definição de árvore de refutação por Resolução-LSDNF($f$) será dada em seguida.

A noção de satisfatibilidade para Cláusulas pseudo-Horn segue imediatamente da Definição 4.3, com a diferença da possibilidade da presença de literais negativos no corpo das regras. As Cláusulas correspondentes possuem então o literal do conseqüente e os complementos dos literais do antecedente.

O método formal de Resolução com função de seleção $f$ para Conjuntos quase-pseudo-Definidos, $RSDNF(f)$, consiste de duas regras de inferência. A regra da $f$-extensão adaptada e uma nova regra, a $f$-contração por falha finita, que é definida a partir da Negação por Falha Finita introduzida de modo informal anteriormente.

DEFINIÇÃO 4.15 : *$f$-extensão e $f$-contração por falha finita*

***$f$-extensão***

*Sejam $f$ uma função seleção, $G$ uma cláusula pseudo-objetivo da forma $\leftarrow g_1, g_2, \ldots, g_n$, $D$ uma cláusula pseudo-definida e $\beta$ uma renomeação de $D$ em presença de $G$. Suponha que $f(G) = g_i$ e que $g_i$ é um literal positivo. Se $D\beta$ é da forma $C \leftarrow A_1, A_2, \ldots, A_k$, uma cláusula $G'$ é uma* $f$-extensão de $G$ por $D$ *sse existe um u.m.g. $\theta$ de $\{g_i, C\}$ tal que $G'$ é da forma $\leftarrow (g_1, g_2, \ldots, g_{i-1}, A_1, A_2, \ldots, A_k, g_{i+1}, g_{i+2}, \ldots, g_n)\theta$.*

*Uma cláusula pseudo-objetivo $G$ é uma* $f$-extensão canônica *de uma cláusula pseudo-objetivo $G'$ em presença de um conjunto de cláusulas pseudo-definidas $Q$ se e somente se $G$ é a variante canônica de uma $f$-extensão de $G'$ com uma cláusula pseudo-definida de $Q$.*

*f-contração por falha finita*

*Sejam f uma função seleção, G uma cláusula pseudo-objetivo da forma* $\leftarrow g_1,g_2,\ldots,g_n$ *e Q um conjunto de cláusulas pseudo-definidas. Suponha que* $f(G) = g_i$ *e que* $g_i$ *é um literal básico negativo da forma* $\neg L$*. A f-*contração por falha finita de *G* em presença de *Q é a cláusula* $\leftarrow g_1,g_2,\ldots,g_{i-1},g_{i+1},\ldots,g_m$*, sse existe uma árvore de LSDNF(f)-refutação A a partir de Q e* $\leftarrow L$ *tal que A é finita e não possui nó de sucesso.*

As restrições de que o literal selecionado na regra da $f$-contração por falha finita seja negativo e básico e a tentativa de se construir finitamente uma árvore de refutação SLDNF com raiz rotulada por $\leftarrow \neg L$, correspondem à definição de *Regra de Computação Segura* utilizada por [Llo84] na definição de LSDNF($f$)-dedução (chamada por ele de *SLDNF-derivation*).

DEFINIÇÃO 4.16 : *Árvore de LSDNF(f)-refutação*

*Sejam f uma função de seleção, Q um conjunto de cláusulas pseudo-definidas e C uma cláusula pseudo-objetivo. Uma árvore, com os nós rotulados por cláusulas pseudo-objetivo, é uma árvore de LSDNF(f)-refutação para Q e C se e somente se:*

*i) O rótulo da raiz é C;*

*ii) Para cada nó X, o conjunto dos filhos de X é definido da seguinte forma: Seja A' o rótulo de X e suponha que A' tenha a forma* $\leftarrow N_1,N_2,\ldots,N_k$ *e que* $f(A') = N_i$

- *Se* $N_i$ *é um literal positivo então para cada f-extensão canônica A de A' em presença de Q, existe exatamente um filho de X rotulado com A*
- *Se* $N_i$ *é um literal negativo básico da forma* $\neg L$ *e a árvore de LSDNF(f)-refutação para* $Q \cup \{\leftarrow L\}$ *for finita e não possuir nenhum nó de sucesso, então o nó X terá um único filho rotulado por* $\leftarrow N_1,N_2,\ldots,N_{i-1},N_{i+1},\ldots,N_k$*. Se* $N_i$ *contiver variáveis livres, ou a árvore for infinita ou possuir algum nó de sucesso, então o nó X não terá filhos.*

EXEMPLO 4.4 : *Uma árvore de LSDNF(f)-refutação ([CGF87])*

***Conjunto quase-pseudo-definido***

1. $chama(a,b)$

2. $usa(b, e)$

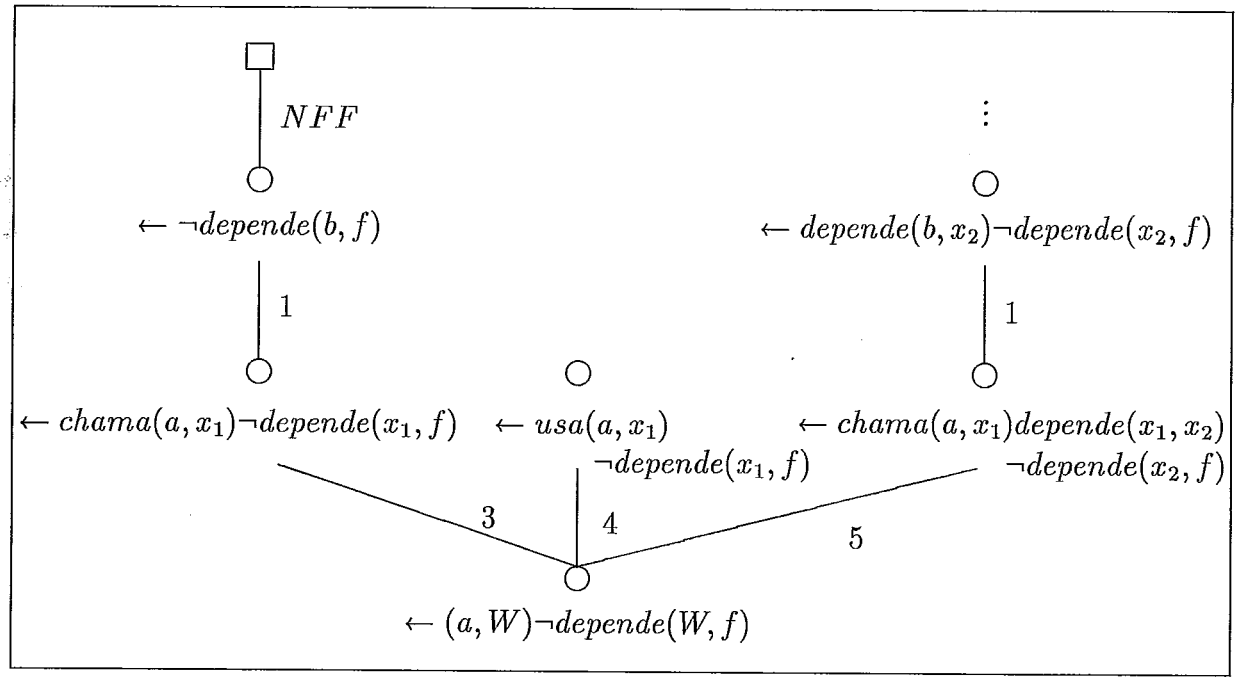
3. $depende(X, Y) \leftarrow chama(X, Y)$

4. $depende(X, Y) \leftarrow usa(X, Y)$

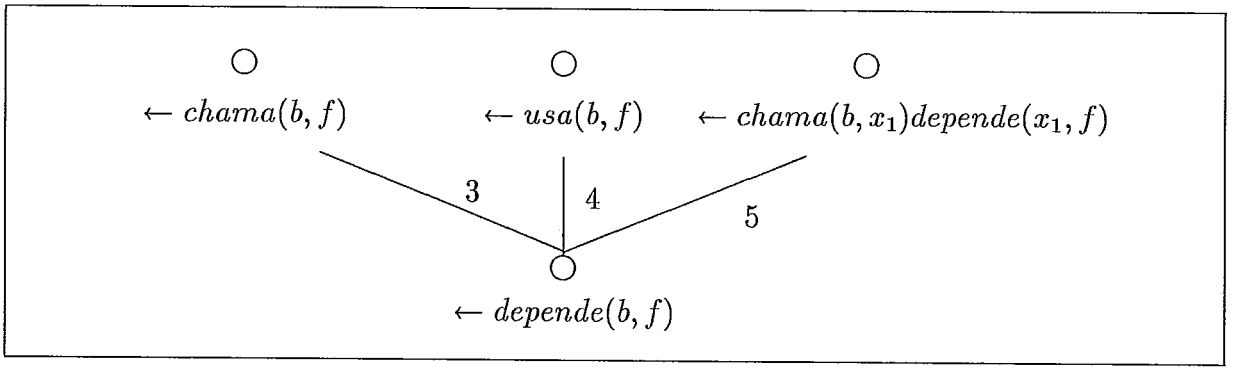
5. $depende(X, Y) \leftarrow chama(X, Z), depende(Z, Y)$

6. $\leftarrow depende(a, W), \neg depende(W, f)$

***Árvore de LSDNF(f)-refutação para*** $\leftarrow depende(a, W), depende(W, f)$




***Árvore de LSDNF(f)-refutação para*** $\leftarrow depende(b, f)$



DEFINIÇÃO 4.17 : *LSDNF(f)-dedução e LSDNF(f)-refutação*

***LSDNF(f)-dedução***

*Sejam f uma função de seleção, S um conjunto quase-pseudo-definido e G uma cláusula pseudo-objetivo. Uma* LSDNF($f$)-dedução *de uma cláusula pseudo-Horn $G'$ a partir de $S \bigcup \{G\}$ é uma seqüência $C = (C_1, C_2, \ldots, C_n)$ de cláusulas, terminando em $G'$, isto é, $C_n = G'$ e $\exists r \leq n$ tal que:*

*i) para todo $i < r$, $C_i$ é uma cláusula pseudo-definida pertencente a $S$;*

*ii) $C_r$ é a cláusula $G$;*

*iii) para todo $i > r$ se $f(C_{i-1})$*

- *Se $f(C_{i-1})$ for positivo, então $C_i$ é obtido pela aplicação da regra da $f$-extensão a $C_{i-1}$ e $C_j$, para algum $j < r$.*
- *Se $f(C_{i-1})$ for negativo e básico, então $C_i$ é a $f$-contração por falha finita de $C_{i-1}$ em presença de $D_1, D_2, \ldots, D_{r-1}$.*

***LSDNF(f)-refutação***

*Uma* LSDNF($f$)-refutação *a partir de um conjunto pseudo-definido $S$ e uma cláusula pseudo-objetivo $C$ é uma LSDNF(f)-dedução da cláusula vazia a partir do conjunto $S \bigcup \{C\}$.*

### 4.5.1 Correção e Completude da Resolução-LSDNF

A prova de Correção do método RSDNF($f$) depende antes da Hipótese de Mundo Fechado (HMF). Esse fato acontece porque a regra da NFF "completa" o conhecimento acerca do mundo descrito, assumindo que aquilo que não foi especificado é falso.

Por exemplo, suponha o seguinte conjunto quase-definido $S$:

$p(a)$
$\leftarrow p(a)$

Intuitivamente está tentando se provar $p(a)$ (que é a cláusula objetivo). Na realidade o que o sistema faz é tentar provar que o conjunto $S \bigcup \{\neg p(a)\}$ é insatisfatível, pois $\leftarrow p(a)$ representa a cláusula $\neg p(a)$. E essa prova é obtida quando se acha uma LSD-refutação no conjunto quase-definido.

Com a NFF é necessário um suporte formal que estabeleça que o que não foi descrito deve ser assumido como falso. Sem esse aparato não é possível se provar a Correção do método. Suponha o seguinte exemplo retirado de [CGF87]:

EXEMPLO 4.5 : *Uma LSDNF-refutação a partir de* $Q \bigcup \{C\}$

*Sejam* $Q = \emptyset$ *e* $C = \leftarrow \neg p(a)$. *Como a cláusula pseudo-objetivo* $C$ *só tem um literal,* $f(C) = \neg p(a)$ *para toda função de seleção* $f$.

1. $\leftarrow \neg p(a)$
2. $\square$ ***NFF***

*A seqüência acima é uma LSDNF(f)-refutação a partir de* $Q \bigcup \{C\}$, *logo o conjunto* $Q \bigcup \{p(a)\}$ *é insatisfatível, e portanto* $\neg p(a)$ *é verdadeiro. Contudo,* $Q \bigcup \{p(a)\}$ *é de fato satisfatível, pois em particular o conjunto* $\{p(a)\}$ *lhe é um H-modelo.*

Para que a resposta fosse correta, era necessário estabelecer que, assumindo a Hipótese de Mundo Fechado, $\forall x \neg P(x)$, uma vez que $Q$ é vazio. E é isso que é feito com a noção de Definição completa de um predicado:

DEFINIÇÃO 4.18 : *Definição Completa de um predicado*

*Sejam* $Q$ *um conjunto de claúsulas pseudo-definidas,* $p$ *um símbolo predicativo n-ário.*

- *caso 1: Se* $p$ *ocorre na cabeça de alguma cláusula de* $Q$, *então sejam* $C_1, C_2, \ldots, C_k$, *todas as cláusulas de* $Q$ *cuja cabeça seja um literal sobre* $p$ *e* $x_1, x_2, \ldots, x_n$ *variáveis que não ocorrem nessas cláusulas. A* Definição Completa *de* $p$ *em* $Q$ *é a fórmula*

  $$\forall x_1 \ldots \forall x_n (p(x_1, x_2, \ldots, x_n) \equiv (E_1 \vee \ldots \vee E_k))$$

  *onde* $E_i$ $(1 \leq i \leq k)$ *é a fórmula*

  $$\exists y_1, \ldots, \exists y_j (x_1 = t_1 \wedge \ldots \wedge x_n = t_n \wedge M_1 \wedge \ldots \wedge M_m)$$

  *se* $C_i$ *é da forma* $p(t_1, t_2, \ldots, t_n) \leftarrow M_1, M_2, \ldots, M_m$ *e as variáveis que ocorrem em* $C_i$ *são* $y_1, y_2, \ldots, y_j$.

- *caso 2: Se* $p$ *não ocorre na cabeça de nenhuma cláusula de* $Q$, *então a* Definição Completa *de* $p$ *em* $Q$ *é a fórmula*

$\forall x_1 \ldots \forall x_n(\neg p(x_1,x_2,\ldots,x_n))$

Para que a Correção da Resolução-LSDNF seja estabelecida, é necessário ainda estabelecer o conceito de fecho de um Conjunto de Cláusulas pseudo-Definidas que deve possuir uma teoria apropriada da igualdade e indicar explicitamente a distinção entre os indivíduos denotados por constantes ou símbolos funcionais. A Teoria de Igualdade descrita abaixo está em [Llo84] e será utilizada para esse fim. A expressão $\forall F$ é usada para denotar o fecho universal da fórmula $F$.

**Uma Teoria para a Igualdade**

1. $c \neq d$ para todos os pares de constantes distintas $c$ e $d$
2. $\forall(f(x_1,x_2,\ldots,x_n) \neq g(y_1,y_2,\ldots,y_m))$, para todos os pares de funções distintas $f$ e $g$
3. $\forall(f(x_1,x_2,\ldots,x_n) \neq c)$, para cada função $f$ e cada constante $c$
4. $\forall(t[x] \neq x)$, para cada termo não variável $t[x]$ contendo $x$
5. $\forall((x_1 \neq y_1) \vee \ldots \vee (x_n \neq y_n) \rightarrow f(x_1,x_2,\ldots,x_n) \neq f(y_1,y_2,\ldots,y_n))$, para cada função $f$
6. $\forall(x = x)$
7. $\forall((x_1 = y_1) \wedge \ldots \wedge (x_n = y_n) \rightarrow f(x_1,x_2,\ldots,x_n) = f(y_1,y_2,\ldots,y_n))$, para cada função $f$
8. $\forall((x_1 = y_1) \wedge \ldots \wedge (x_n = y_n) \rightarrow (p(x_1,x_2,\ldots,x_n) \rightarrow p(y_1,y_2,\ldots,y_n)))$, para cada predicado $p$, incluindo $=$

DEFINIÇÃO 4.19 : *Fecho de um Conjunto de Cláusulas pseudo-Definidas*

*O* Fecho *de um Conjunto de Cláusulas pseudo-Definidas $P$ sobre um alfabeto de primeira ordem $A$, denotado por $fecho(P)$, é o conjunto formado pelas definições completas de todos os predicados em $P$ junto com a Teoria para Igualdade.*

Na verdade o que se está tentando fazer é capturar formalmente a completude dos predicados fornecida pela Hipótese de Mundo fechado, de modo que o resultado de Correção possa então ser estabelecido. A insatisfatibilidade de um conjunto de Cláusulas pseudo-Definidas é então testada não a partir do Conjunto em si, mas do seu fecho, sendo a Resolução-LSDNF correta sob esse aspecto.

DEFINIÇÃO 4.20 :

*Sejam $Q$ um conjunto de Cláusulas pseudo-Definidas e $C$ uma cláusula pseudo-objetivo sobre um alfabeto $A$. O conjunto quase-pseudo-definido $Q \bigcup \{C\}$ é insatisfatível em presença da Hipótese de Mundo Fechado, sse $fecho(Q) \bigcup \{C\}$ for insatisfatível.*

O seguinte Teorema retirado de [Cla78] estabelece a Correção do Método:

**Teorema 4.4** *Seja $f$ uma função de seleção. Sejam $Q$ um Conjunto de Cláusulas pseudo-Definidas e $C$ uma Cláusula pseudo-Objetivo. Se existe uma LSDNF($f$)-refutação a partir de $Q \bigcup \{C\}$ então $Q \bigcup \{C\}$ é insatisfatível em presença da Hipótese de Mundo Fechado.*

Em outras palavras, o que o teorema acima estabelece é que se existe uma LSDNF($f$)-refutação a partir de um conjunto pseudo-Definido $S$ e uma cláusula pseudo-objetivo $C$, então o conjunto $fecho(S) \bigcup \{C\}$ é insatisfatível.

A Resolução-LSDNF não pode ser considerada completa. Se o fosse então para todo conjunto de cláusulas pseudo-definidas $Q$ e toda cláusula pseudo-objetivo $C$ se $Q \bigcup \{C\}$ fosse insatisfatível deveria haver uma LSDNF($f$)-refutação para $Q \bigcup \{C\}$.

O exemplo abaixo ilustra um caso onde o conjunto quase-pseudo-definido é insatisfatível mas não é possível encontrar uma LSDNF($f$)-refutação:

EXEMPLO 4.6 :

*Sejam $Q = \{q(a) \leftarrow p(a)\}$ e $C = \leftarrow \neg p(X)$. Por definição, $fecho(Q)$ contem a fórmula $\forall x \neg p(x)$. $C$ é satisfatível sse a fórmula $\forall x p(x)$ for satisfatível, também por definição. Logo o conjunto $fecho(Q) \bigcup \{C\}$ é insatisfatível. Contudo, independentemente da função de seleção, não é possível encontrar uma LSDNF($f$)-refutação a partir desse conjunto, pois para qualquer função de seleção $f$, $f(C) = \neg p(X)$. Como esse literal é negativo, mas não é básico, nenhuma das duas regras da Resolução-LSDNF se aplicam e dessa forma uma LSDNF($f$)-refutação não é encontrada.*

Para maiores detalhes sobre os problemas relacionados à Completude do Método da Resolução-LSDNF, veja [CGF87].

# Capítulo 5

# CONSIDERAÇÕES SOBRE O PROBLEMA DA GERAÇÃO DE PLANOS

## 5.1 Introdução

Esse capítulo faz uma breve introdução aos principais conceitos relacionados ao problema da geração de planos. Alguns dos problemas mais comumente enfrentados são analisados e ao final são apresentados e discutidos dois sistemas para geração de planos encontrados na literatura.

Para facilitar a definição do problema da geração de planos é interessante introduzir antes alguns conceitos correlatos.

Um **Mundo** é uma parte do universo que se quer modelar. Um **Estado** é uma descrição do mundo, num determinado instante, numa linguagem formal. O estado do mundo permance estático até que uma **Ação** seja executada, quando então a descrição do mundo muda, resultando num novo estado.

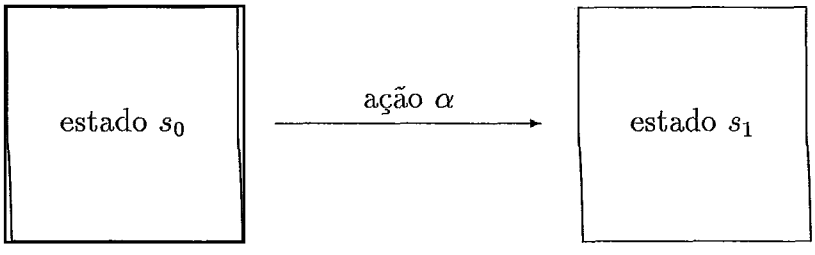



Figura 5.1: mudança de estado

A cada ação está associado um **Contexto habilitador**, que determina sob que condições a ação pode ser aplicada num estado. Esse contexto habilitador é caracterizado por um conjunto de **Pré-condições de aplicação da ação.**

Informalmente, o problema da geração de planos consiste em se determinar uma seqüência de ações que, quando executadas, mudem o mundo de um estado inicial dado, a um estado onde um conjunto de propriedades desejadas sejam verificadas. Esse conjunto de propriedades é comumente chamado de ***Goal*** ou **Objetivo**. Como em geral pode haver mais de um estado onde o *Goal* seja verificado, esses estados são denominados **Conjunto de Estados Objetivos Possíveis.**

Quando se definem os efeitos de uma ação sobre um mundo descrito numa linguagem formal como a Lógica de Primeira Ordem, é necessário se descrever não apenas as propriedades que mudam por sua execução, mas também aquelas que não são afetadas pela mesma. Dessa forma, se num estado temos a descrição de $n$ propriedades e existem $m$ ações no sistema, são necessárias $m \times n$ regras para descrever as propriedades que não são alteradas pela execução das ações. Esse problema ficou conhecido por *Problema de Frame* e a busca de métodos formais que possam evitá-lo, garantindo a consistência e precisão dos resultados obtidos, tem desempenhado um papel importante na área de Inteligência Artificial.

As principais características das regras de *Frame* são as seguintes:

- existem num sistema para relacionar as ações com as propriedades que não são afetadas por sua execução; e

- possuem restrições de aplicação de modo que as propriedades que não se espera mais valerem no estado resultante não são derivadas

Num sistema com regras de *Frame*, essas regras possibilitam no novo estado obtido pela execução de uma ação, a dedução apenas das propriedades que continuam válidas. Num sistema sem regras de *Frame*, o novo estado é uma modificação do anterior. Novas propriedades são adicionadas e as que não se espera mais valerem são removidas. A vantagem é que não é necessário se referir às propriedades que não mudam (em geral a maioria).

## 5.2 Tipos de Sistemas para Geração de Planos

Os principais tipos de sistemas utilizados no processo de geração de planos se classificam segundo a forma pela qual a busca pela solução do problema é conduzida.

### 5.2.1 Sistemas Regressivos

Os sistemas regressivos partem do *Goal*, visualizando-o como uma sub-especificação de um estado final e procuram analisar os estados anteriores possíveis em função das ações que teriam adicionado as propriedades do *Goal* não verificadas no Cenário Inicial.

É como se a cada iteração o sistema procurasse uma ação que aplicada ao cenário anterior originasse um cenário com as propriedades do cenário atual e tal que esse cenário anterior tivesse o máximo de características do cenário inicial.

Quando uma ação $\alpha$ desse tipo é encontrada, o componente do *Goal* por ela adicionada é retirado do mesmo e se o sistema está num estado $i$, ele deve recuar para um estado $i-1$ tal que satisfaça as pré-condições de aplicação de $\alpha$, pois se $\alpha$ foi aplicada no cenário $i-1$ é porque o cenário $i-1$ possibilitava sua aplicação. Essas pré-condições passam então a fazer parte do *Goal*. Nessa busca, uma solução é obtida quando o *Goal* pode ser satisfeito no Cenário Inicial.

Por partirem das propriedades desejadas em direção ao estado inicial, tem-se de um lado um certo direcionamento na busca da solução, mas por outro, parte-se de uma definição incompleta de estado. Tudo o que se sabe é que um dos estados finais (de onde a busca começa) possui as propriedades presentes no Goal. Dependendo da aplicação esses estados finais podem ser muitos e torna-se difícil se determinar qual está mais perto do estado inicial.

### 5.2.2 Sistemas Progressivos

Os sistemas progressivos partem da descrição do Cenário Inicial dada e procuram aplicar sucessivamente ações a partir dele até que o cenário atingido satisfaça as propriedades presentes no *Goal*. As principais vantagens e desvantagens desses dois tipos de sistemas serão discutidas a seguir.

Alguns dos problemas verificados nos sistemas regressivos não aparecem nos progressivos. Isso decorre do fato de que o processo de busca inicia-se de um estado

bem descrito e as aplicações das ações por sua vez também geram novos estados bem definidos. Como conseqüência, é possível se determinar com precisão quais ações são realmente aplicáveis a um determinado cenário.

Conforme dito anteriormente, os mecanismos progressivos carecem de um procedimento que os direcione à solução do problema para reduzir o espaço de busca. Em geral, nesse tipo de sistema, é usada uma análise *means-end* que, baseada em algum critério, estima um custo de aplicação das diversas ações disponíveis escolhendo aquela de menor valor.

### 5.2.3 Sistemas Lineares

Os sistemas lineares tentam resolver um *Goal* assumindo que cada componente do mesmo pode ser satisfeito individualmente. Quando um dos componentes do *Goal* é satisfeito, o sistema passa a trabalhar no seguinte, atingindo uma solução quando todos os componentes forem satisfeitos.

Alguns problemas de geração de planos não podem ser resolvidos de forma ótima por esse tipo de sistema. O mais conhecido destes é o chamado *Problema dos Três Blocos* e foi apontado em [Sus73] como uma "situação anômala" e posterioremente referenciado por [War74].

#### Problema dos Três Blocos

Esse problema é um problema do mundo dos blocos no qual os estados inicial e objetivo estão conforme a figura abaixo, ou seja o estado objetivo é tal que o bloco $a$ está sobre o bloco $b$ e o bloco $b$ sobre o bloco $c$ ($sobre(a, b) \wedge sobre(b, c)$).

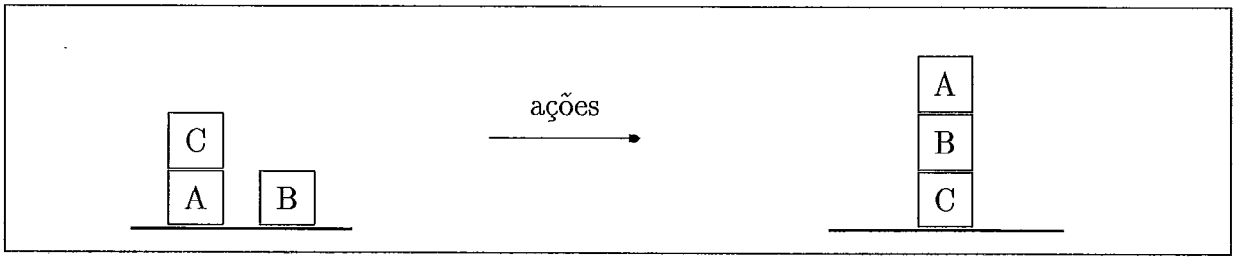



Figura 5.2: Cenário Inicial e Cenário Objetivo do problema

Nos sistemas lineares, esse problema ocorre porque a resolução de um dos componentes do *Goal*, desfaz parte da solução obtida para outro dos componentes. Nilsson

([Nil80]) se refere ao problema dos três blocos afirmando que os componentes do *Goal* $sobre(b, c) \wedge sobre(a, b)$ *interagem.*

Suponha que a ordem de solução num sistema linear seja primeiro atingir $sobre(a, b)$ e depois $sobre(b, c)$. Para atingir $sobre(a, b)$ o sistema executaria a sequinte seqüência de ações (onde $move(c, a, chao)$ representa o ato de mover o bloco $c$ de cima do bloco $a$ para o chão; e $move(a, chao, b)$ o ato de mover o bloco $a$ do chão para sobre o bloco $b$):

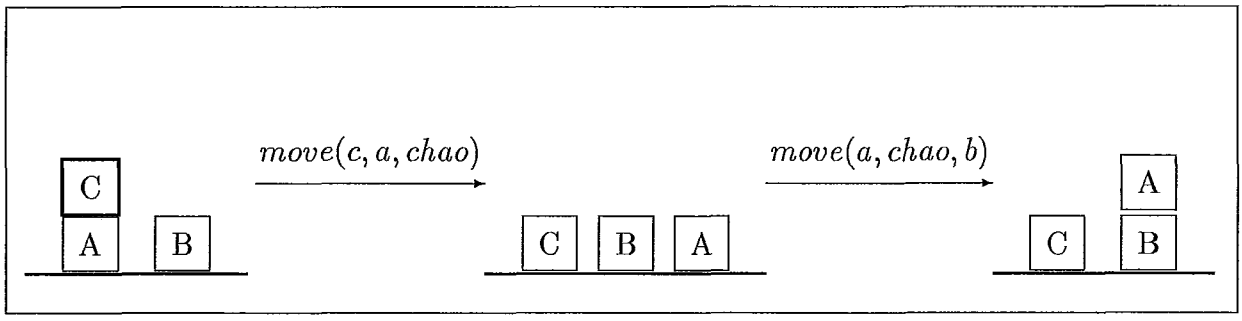



Figura 5.3: Solução intermediária de $sobre(a, b)$

O sistema passaria então a trabalhar no próximo componente do *Goal*, o literal $sobre(b, c)$. Contudo, para empilhar o bloco $b$ sobre o bloco $c$, o bloco $b$ deveria estar livre e isso só pode ser obtido se parte da solução obtida para o componente $sobre(a, b)$ for desfeita. O mesmo aconteceria se a ordem de solução do *Goal* tivesse sido $sobre(b, c)$ e depois $sobre(a, b)$:

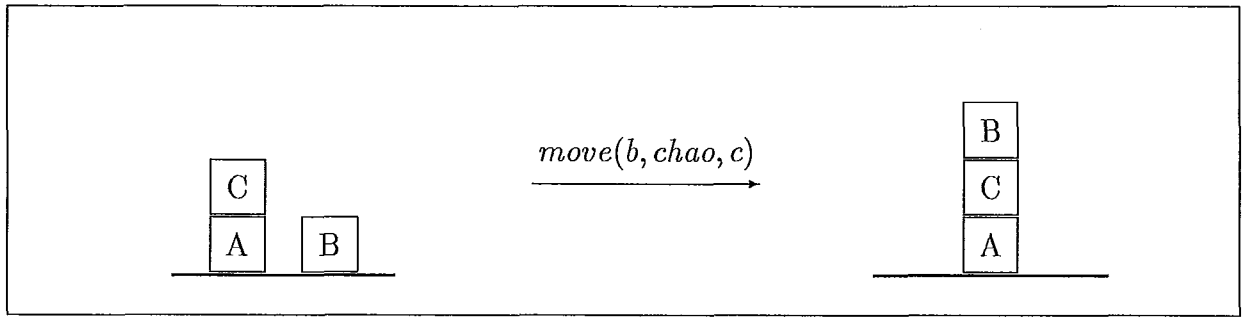



Figura 5.4: Solução intermediária de $sobre(b, c)$

Nesse caso, para atingir $sobre(a, b)$ todos os passos executados para atingir $sobre(b, c)$ devem ser desfeitos. Um plano ótimo para o problema acima seria o descrito em 5.1.

$$[move(c, a, chao)][move(b, chao, c)][move(a, chao, b)] \quad (5.1)$$

## 5.3 Warplan

O sistema Warplan ([War74]) pode ser considerado como um sistema misto, pois utiliza as características dos sistemas regressivos no sentido de que parte do *Goal* para o Cenário Inicial, mas no momento de testar uma propriedade de um cenário específico funciona como um sistema progressivo pois esse cenário é obtido atualizando-se o cenário inicial através das ações no plano até então gerado. Um ponto forte a favor do Warplan é que é fornecida uma implementação do provador tornando possível uma análise mais detalhada das suas características.

Nesse sistema, os estados intermediários não são gerados durante a solução de um plano. Quando uma propriedade precisa ser verificada num cenário, um mecanismo especial atualiza o cenário inicial em relação a essa propriedade efetuando a verificação. Isso reduz o espaço de armazenamento e sobretudo a computação necessária para atualizar os cenários e tem como conseqüência imediata que cenários e planos podem ser vistos como a mesma entidade. Contudo, conforme mencionado pelo próprio autor, há mais computação envolvida para acessar um fato em particular.

O Warplan não enfrenta os problemas dos chamados sistemas lineares. No momento de agregar uma nova ação ao plano, a mesma não precisa ser adicionada ao final do mesmo, mas em qualquer posição que preserve a integridade do plano. Os critérios básicos para isso são:

i) Uma ação deve preservar propriedades que sejam verificadas no estado inicial e que sejam pre-condições de aplicação de ações localizadas em posições posteriores no plano.

ii) Uma ação deve preservar propriedades adicionadas pelas ações localizadas em posições anteriores no plano que sejam necessárias à solução do problema.

iii) As pré-condições de uma ação devem:

   a) ser verificadas no estado inicial do sistema e serem preservadas pelas ações localizadas em posições anteriores no plano ou;

   b) ser adicionadas por ações localizadas em posições anteriores no plano e preservadas pelas ações a partir destas até o ponto onde a nova ação será inserida.

| Ação | pré-condição | acrescenta | retira |
|---|---|---|---|
| $[move(c, a, chao)]$ | $sobre(c, a), a \neq chao,$ $limpo(c)$ | $sobre(c, chao),$ $limpo(a)$ | $sobre(c, a),$ $limpo(chao)$ |
| $[move(b, chao, c)]$ | $limpo(c), sobre(b, chao),$ $b \neq c, limpo(b)$ | $sobre(b, c),$ $limpo(chao)$ | $sobre(b, chao),$ $limpo(c)$ |
| $[move(a, chao, b)]$ | $limpo(b), sobre(a, chao),$ $a \neq b, limpo(a)$ | $sobre(a, b),$ $limpo(chao)$ | $sobre(a, chao),$ $limpo(b)$ |

Tabela 5.1: Descrição das ações no plano ótimo

Essa facilidade permite que as ações possam ser intercaladas, fazendo com que o sistema não funcione de modo linear. As propriedades do *Goal* já provadas são armazenadas numa lista de fatos ditos *protegidos*. A descrição das ações no plano ótimo (5.1), segundo [War74] pode ser vista na Tabela 5.1.

Nesse exemplo, a descrição do estado inicial seria feita da seguinte forma:

**Estado Inicial**

$sobre(a, chao)$
$sobre(b, chao)$
$sobre(c, a)$
$limpo(b)$
$limpo(c)$

Supondo a ordem $sobre(a, b)$, $sobre(b, c)$ para os componentes do *Goal*, a execução seguiria conforme descrito abaixo (para simplificar, parte das instanciações já foi efetuada):

A primeira ação escolhida é $[move(a, X, b)]$, pois esta acrescenta o componente $sobre(a, b)$ do *Goal*. As pré-condições desta ação passam a fazer parte do *Goal*, que agora fica igual a:

$$limpo(b), sobre(a, X), X \neq b, limpo(a), sobre(b, c)$$

Fazendo $X = chao$, os três primeiros componentes são satisfeitos no estado inicial, e a primeira ação do plano passa a ser $[move(a, chao, b)]$. Essa ação ainda tem uma pré-condição que precisa ser satisfeita, dessa forma, uma outra ação que acrescente o literal $limpo(a)$ deve ser inserida antes dela. A ação adequada a isto é $[move(c, a, chao)]$,

cujas pré-condições, conforme pode ser visto na Tabela 5.1, são satisfeitas no estado inicial. Dessa forma, o primeiro componente do *Goal* original, $sobre(a, b)$ foi totalmente satisfeito e o plano para isso é o seguinte:

$$[move(c, a, chao)][move(a, chao, b)] \tag{5.2}$$

Num sistema linear, o outro componente $sobre(b, c)$ não poderia ser obtido sem desfazer parte dessa solução. O que o Warplan faz é procurar um lugar adequado onde possa inserir a nova ação necessária ao plano.

Para solucionar $sobre(b, c)$ a ação $[move(b, chao, c)]$ deve ser usada. Contudo, a mesma não pode ser adicionada nem ao final de 5.2, nem antes do mesmo, pois os estados correspondentes não satisfazem as suas pre-condições. O Warplan então percorre o plano de trás para a frente tentando satisfazê-las, observando que as pré-condições das ações seguintes e propriedades por elas adicionadas continuem satisfeitas.

O primeiro ponto é entre as duas ações de 5.2. Todas as pré-condições de $move(b, chao, c)$ são verificadas no estado inicial e preservadas por $[move(c, a, chao)]$. Quanto às pré-condições de $[move(a, chao, b)]$, a saber, $limpo(a)$, $mesa(a)$ e $limpo(b)$:

$limpo(a)$ É acrescentada por $[move(c, a, chao)]$ e preservada por $[move(b, chao, c)]$

$mesa(a)$ É verificada no estado inicial, preservada por $[move(c, a, chao)]$ e também preservada por $[move(b, chao, c)]$

$limpo(b)$ É verificada no estado inicial, preservada por $[move(c, a, chao)]$ e também preservada por $[move(b, chao, c)]$

Resta então verificar as propriedades acrescentadas por $[move(c, a, chao)]$, que são necessárias para $[move(a, chao, b)]$, nesse caso, apenas $limpo(a)$. Como esta propriedade também é preservada por $[move(b, chao, c)]$, este ponto é um local satisfatório para inserção da ação e a solução obtida para o problema é mostrada em 5.3.

$$[move(c, a, chao)][move(b, chao, c)][move(a, chao, b)] \tag{5.3}$$

## 5.4 STRIPS

O STRIPS (**ST**anford **R**easerach **I**nstitute **P**roblem **S**olver) é um sistema progressivo voltado para o problema da geração de planos. Apesar de ter surgido em 1971 ([FN71])),

ainda é muito discutido e tomado como referência na maioria dos sistemas desenvolvidos. Apesar de muita discussão sobre o que o STRIPS faz, há pouca literatura disponível sobre como ele realmente funciona.

O STRIPS trabalha com *Modelos de Mundo*. Esses modelos descrevem estados do sistema e são constituídos por fórmulas bem-formadas da Lógica de Primeira Ordem (doravante chamadas simplesmente de fórmulas). Um conjunto de *operadores* são usados para descrever as mudanças de estado. A cada operador está associada uma pré-condição, uma lista de adição e uma lista de remoção. Um operador só pode ser aplicado a um modelo de mundo se o mesmo satisfizer sua pré-condição, representada por uma fórmula. A aplicação de um operador a um modelo de mundo resulta num novo modelo onde as fórmulas presentes na lista de adição se verificam e as fórmulas presentes na lista de remoção deixam de ser verificadas.

O Goal a ser solucionado é representado por uma fórmula. Uma solução é obtida quando o sistema encontra uma seqüência de operadores que transforme o modelo de mundo inicial num modelo onde o *Goal* seja satisfeito.

Uma estratégia de busca similar à usada no [GN69], procura reduzir o espaço de busca. Um provador de teoremas é incorporado para tentar provar os *Goals* a partir do modelo de mundo corrente. Quando uma prova não é encontrada, a parte incompleta da prova é assumida como a *diferença* entre o *Goal* e o modelo atual. O STRIPS procura então por operadores que se aplicados permitiriam a continuação da prova, reduzindo a diferença inicial calculada.

### 5.4.1 Funcionamento

O STRIPS funciona de modo linear. Uma pilha é mantida para controlar os *Goals* a serem satisfeitos. Cada elemento do topo da pilha é analisado e passa a ser o objetivo atual do sistema. Quando o modelo de mundo atual satisfaz esse elemento, ele é retirado da pilha e o sistema passa a trabalhar no próximo se este existir. Uma solução é obtida quando a pilha se torna vazia.

Se o elemento do topo da pilha for uma conjunção não satisfeita no modelo de mundo atual, a mesma é decomposta e os componente empilhados individualmente na ordem inversa em que aparecem na conjunção, que permanece na pilha. A idéia é que a busca de uma solução para cada componente isolado da conjunção forneça uma solução para toda a conjunção. Como isso não é suficiente para resolver o problema dos três blocos

visto anteriormente, a conjunção é depois reavaliada.

Quando o elemento do topo da pilha é um literal não satisfeito no modelo de mundo atual, um operador que o adicione é usado. O elemento é retirado, o operador empilhado e em seguida suas pré-condições na forma usual, tornando-se agora parte do *Goal*. Se o elemento do topo é um operador é porque suas pré-condições foram satisfeitas e ele pode ser aplicado ao modelo atual. O operador é retirado da pilha e aplicado, gerando um novo modelo onde as fórmulas restantes na pilha são avaliadas.

O exemplo abaixo procura ilustrar melhor as idéias discutidas aqui. Nesse exemplo, o operador $pegue(X)$ representa o ato de erguer o bloco $X$; $solte(X)$ o ato de por o bloco $X$ sobre a mesa; $empilhe(X,Y)$ o ato de empilhar um bloco $X$ que esteja erguido, sobre um bloco $Y$ (e abreviado para $emp(X,Y)$) e $desempilhe(X,Y)$ o ato de erguer o bloco $X$ de cima do bloco $Y$ (abreviado para $des(X,Y)$).

As propriedades do sistema são $mesa(X)$ que indica que o bloco $X$ está sobre a mesa; $sobre(X,Y)$, que indica que o bloco $X$ está sobre o bloco $Y$; $limpo(X)$ que o bloco $X$ não tem nenhum bloco sobre ele; $erg(X)$ que o robô está erguendo o bloco $X$ e finalmente *livre* que o braço do robô está livre para pegar um bloco.

As pré-condições e as listas de adição e remoção dos operadores estão descritos na Tabela 5.2. O objetivo é atingir um estado onde a propriedade $sobre(b,c) \wedge sobre(a,b)$ seja satisfeita. A descrição do modelo de mundo inicial está dada abaixo:

### Modelo de Mundo Inicial

$mesa(a)$
$mesa(b)$
$sobre(c,a)$
$limpo(c)$
$limpo(b)$
$livre$

### *Goal* Inicial

$sobre(b,c) \wedge sobre(a,b)$

Um esquema de execução detalhado está descrito na Tabela 5.3.

A seqüência de operadores completa para a solução do problema é obtida através das ações que são executadas pelo STRIPS durante a busca:

| Operador | pré-condição | lista de adição | lista de remoção |
|---|---|---|---|
| $pegue(X)$ | $mesa(X) \wedge livre \wedge limpo(X)$ | $erg(X)$ | $mesa(X) \wedge livre \wedge limpo(X)$ |
| $solte(X)$ | $erg(X)$ | $mesa(X) \wedge limpo(X) \wedge livre$ | $erg(X)$ |
| $emp(X,Y)$ | $erg(X) \wedge limpo(Y)$ | $livre \wedge sobre(X,Y)$ | $erg(X) \wedge limpo(Y)$ |
| $des(X,Y)$ | $livre \wedge limpo(X) \wedge sobre(X,Y)$ | $erg(X) \wedge limpo(Y)$ | $livre \wedge limpo(X) \wedge sobre(X,Y)$ |

Tabela 5.2: Descrição dos operadores

$$[des(c,a)][solte(c)][pegue(a)][emp(a,b)][des(a,b)]$$
$$[solte(a)][pegue(b)][emp(b,c)][pegue(a)][emp(a,b)] \tag{5.4}$$

Apesar de ser um sistema linear, a estratégia de reavaliação do *Goal* após a verificação dos componentes individuais faz com que uma solução seja encontrada, embora esta não seja um plano ótimo.

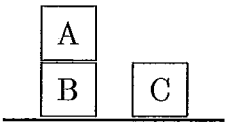
Num sistema puramente linear, ações de *sub-goals* diferentes não poderiam ser intercaladas. O que acontece no STRIPS é que seguindo uma ordem para solucionar o *Goal*, quando a conjunção é reavaliada se uma solução total não foi encontrada, uma nova solução para cada componente é procurada a partir do ponto onde a primeira tentativa falhou.

Observe a seguinte seqüência de operadores do plano 5.4:

$$[des(c,a)][solte(c)][pegue(a)][emp(a,b)]$$

Estes operadores são os necessários para solucionar o componente do *Goal* $sobre(a,b)$. Essa solução, num sistema linear, não poderia ser alterada na busca de uma solução para um outro componente do *Goal*, pois causaria uma *violação de proteção* (veja [War74]). O sistema tentaria então uma busca com outra ordenação do *Goal* a partir do Cenário Inicial. A motivação para essa restrição de proteção é unicamente garantir que todo o *Goal* esteja realmente satisfeito no final da solução.

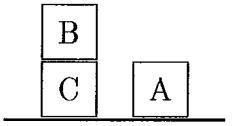
O STRIPS pode ignorar essa restrição porque a conjunção permanece na pilha enquanto os componentes individuais são trabalhados. Isso garante que *goals* previamente atingidos sejam posteriormente reavaliados. A busca de uma solução para o próximo componente prossegue então do ponto atual. Note que o problema agora se transforma em solucionar o *Goal* $sobre(b,c) \wedge sobre(a,b)$ a partir do cenário:




como $sobre(a, b)$ é verdadeiro nesse cenário, o objetivo passa a ser solucionar $sobre(b, c)$. Para solucioná-lo, a seguinte seqüência de operadores é necessária:

$[des(a, b)][solte(a)][pegue(b)][emp(b, c)]$

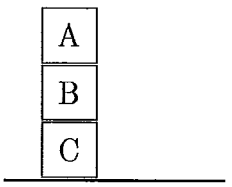
Nesse ponto, o estado do sistema está da seguinte forma:




E o *Goal* original $sobre(b, c) \wedge sobre(a, b)$ é então reavaliado. $sobre(a, b)$ ainda não é verificado, mas a seqüência

$[pegue(a)][emp(a, b)]$

resolve esse componente levando o sistema ao cenário




onde toda a conjunção é satisfeita.

Detalhes sobre como a reavaliação do *Goal* é efetivamente executada não são descritos na literatura. Não se sabe por exemplo, se há um mecanismo para controle de "loop", quando uma solução de fato não existe. Apesar de ser permitida a descrição das propriedades através de quaisquer fórmulas da linguagem da Lógica de Primeira Ordem, apenas sob restrições específicas é possível se garantir a Correção dos resultados fornecidos pelo STRIPS. Uma excelente análise dessas restrições é encontrada em [Lif87].

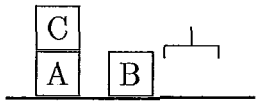
| It. | Estado da pilha | Op. relevante ou observações | Pré-condições do operador ou Estado do Sistema |
|---|---|---|---|
| 0 | $sobre(b,c) \wedge sobre(a,b)$ | |   |
| 1 | $sobre(a,b)$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $emp(a,b)$ | $erg(a) \wedge limpo(b)$ |
| 2 | $limpo(b)$<br>$erg(a)$<br>$erg(a) \wedge limpo(b)$<br>(1) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | | |
| 3 | $erg(a)$<br>$erg(a) \wedge limpo(b)$<br>(1) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $pegue(a)$ | $mesa(a) \wedge livre \wedge limpo(a)$ |
| 4 | $limpo(a)$<br>$livre$<br>$mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(2) $[pegue(a)]$<br>$erg(a) \wedge limpo(b)$<br>(1) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $des(X,a)$ | $livre \wedge limpo(X) \wedge sobre(X,a)$ |

Tabela 5.3: Esquema de Execução do STRIPS para o problema dos três blocos (cont.)

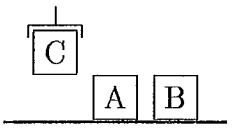
| It. | Estado da pilha | Op. relevante ou observações | Pré-condições do operador ou Estado do Sistema |
|---|---|---|---|
| 5 | $sobre(X,a)$<br>$limpo(X)$<br>$livre$<br>$livre \wedge limpo(X) \wedge sobre(X,a)$<br>(3) $[des(X,a)]$<br>$livre$<br>$mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(2) $[pegue(a)]$<br>$erg(a) \wedge limpo(b)$<br>(1) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | faz $\{X/c\}$, satisfaz os 4 primeiros elementos da pilha e executa (3) |  |
| 6 | $livre$<br>$mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(2) $[pegue(a)]$<br>$erg(a) \wedge limpo(b)$<br>(1) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | | |
| 7 | $livre$<br>$mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(2) $[pegue(a)]$<br>$erg(a) \wedge limpo(b)$<br>(1) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $solte(c)$ | $erg(c)$ |

Tabela 5.3: Esquema de Execução do STRIPS para o problema dos três blocos (cont.)

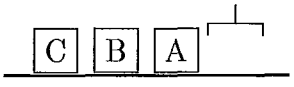
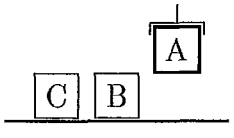
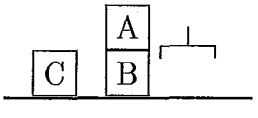
| It. | Estado da pilha | Op. relevante ou observações | Pré-condições do operador ou Estado do Sistema |
|---|---|---|---|
| 8 | $erg(c)$<br>(4) [solte(c)]<br>$mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(2) [$pegue(a)$]<br>$erg(a) \wedge limpo(b)$<br>(1) [$emp(a,b)$]<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz $erg(c)$ e executa (4) |  |
| 9 | $mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(2) [$pegue(a)$]<br>$erg(a) \wedge limpo(b)$<br>(1) [$emp(a,b)$]<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz os dois primeiros elementos e executa (2) |  |
| 10 | $erg(a) \wedge limpo(b)$<br>(1) [$emp(a,b)$]<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz o elemento do topo e executa (1) |  |
| 11 | $sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $emp(b,c)$ | $erg(b) \wedge limpo(c)$ |
| 12 | $limpo(c)$<br>$erg(b)$<br>$erg(b) \wedge limpo(c)$<br>(5) [$emp(b,c)$]<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz o elemento do topo | |
| 13 | $erg(b)$<br>$erg(b) \wedge limpo(c)$<br>(5) [$emp(b,c)$]<br>$sobre(b,c) \wedge sobre(a,b)$ | $pegue(b)$ | $mesa(b) \wedge livre \wedge limpo(b)$ |

Tabela 5.3: Esquema de Execução do STRIPS para o problema dos três blocos (cont.)

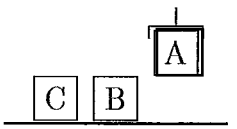
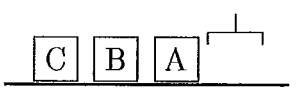
| It. | Estado da pilha | Op. relevante ou observações | Pré-condições do operador ou Estado do Sistema |
|---|---|---|---|
| 14 | $limpo(b)$<br>$livre$<br>$mesa(b)$<br>$mesa(b) \wedge livre \wedge limpo(b)$<br>(6) $[pegue(b)]$<br>$erg(b) \wedge limpo(c)$<br>(5) $[emp(b,c)]$<br>$sobre(b,c) \wedge sobre(a,b)$ | $des(a,b)$ | $livre \wedge limpo(a) \wedge sobre(a,b)$ |
| 15 | $sobre(a,b)$<br>$limpo(a)$<br>$livre$<br>$livre \wedge limpo(a) \wedge sobre(a,b)$<br>(7) $[des(a,b)]$<br>$livre$<br>$mesa(b)$<br>$mesa(b) \wedge livre \wedge limpo(b)$<br>(6) $[pegue(b)]$<br>$erg(b) \wedge limpo(c)$<br>(5) $[emp(b,c)]$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz os 4 primeiros elementos e executa (7) |  |
| 16 | $mesa(b)$<br>$mesa(b) \wedge livre \wedge limpo(b)$<br>(6) $[pegue(b)]$<br>$erg(b) \wedge limpo(c)$<br>(5) $[emp(b,c)]$<br>$sobre(b,c) \wedge sobre(a,b)$ | $solte(a)$ | $erg(a)$ |
| 17 | $erg(a)$<br>(8) $[solte(a)]$<br>$mesa(b)$<br>$mesa(b) \wedge livre \wedge limpo(b)$<br>(6) $[pegue(b)]$<br>$erg(b) \wedge limpo(c)$<br>(5) $[emp(b,c)]$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz o elemento do topo e executa (8) |  |

Tabela 5.3: Esquema de Execução do STRIPS para o problema dos três blocos (cont.)

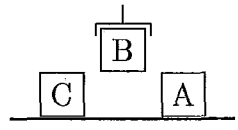
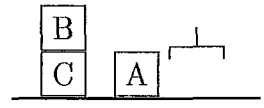
| It. | Estado da pilha | Op. relevante ou observações | Pré-condições do operador ou Estado do Sistema |
|---|---|---|---|
| 18 | $mesa(b)$<br>$mesa(b) \wedge livre \wedge limpo(b)$<br>(6) $[pegue(b)]$<br>$erg(b) \wedge limpo(c)$<br>(5) $[emp(b,c)]$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz os 2 primeiros elementos e executa (6) |  |
| 19 | $erg(b) \wedge limpo(c)$<br>(5) $[emp(b,c)]$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz o elemento do topo e executa (5) |  |
| 20 | $sobre(b,c) \wedge sobre(a,b)$ | não é satisfeito | |
| 21 | $sobre(a,b)$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $emp(a,b)$ | $erg(a) \wedge limpo(b)$ |
| 22 | 23 $limpo(b)$<br>$erg(a)$<br>$erg(a) \wedge limpo(b)$<br>(9) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz o elemento do topo | |
| 23 | $erg(a)$<br>$erg(a) \wedge limpo(b)$<br>(9) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | $pegue(a)$ | $mesa(a) \wedge livre \wedge limpo(a)$ |

Tabela 5.3: Esquema de Execução do STRIPS para o problema dos três blocos (cont.)

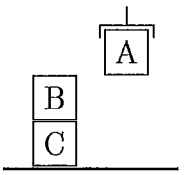
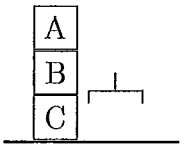
| It. | Estado da pilha | Op. relevante ou observações | Pré-condições do operador ou Estado do Sistema |
|---|---|---|---|
| 24 | $limpo(a)$<br>$livre$<br>$mesa(a)$<br>$mesa(a) \wedge livre \wedge limpo(a)$<br>(10) $[pegue(a)]$<br>$erg(a) \wedge limpo(b)$<br>(9) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz os 4 primeiros elementos da pilha e executa (10) |  |
| 25 | 26 $erg(a) \wedge limpo(b)$<br>(9) $[emp(a,b)]$<br>$sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz o elemento do topo e executa (9) |  |
| 26 | $sobre(b,c)$<br>$sobre(b,c) \wedge sobre(a,b)$ | satisfaz os dois elementos | |
| 27 | $\emptyset$ | uma solução foi obtida | |

Tabela 5.3: Esquema de Execução do STRIPS para o problema dos três blocos (cont.)

# Capítulo 6

# PROLOG MODAL DE AÇÃO

## 6.1 Introdução

O Prolog Modal de Ação tem por objetivo incorporar os conceitos de mudanças de cenário da Lógica Modal de Ação num mecanismo com semântica procedural similar à do Prolog. Nesse formalismo, o conjunto de informações acerca de um estado do sistema é denominado Cenário. O estado inicial do sistema será referido por Cenário Inicial.

A representação de um problema é efetuada através de um conjunto de fatos da Linguagem das Cláusulas de Horn, usado para descrever parte da informação do Cenário Inicial; e um conjunto de regras, denominado Especificação, que diz respeito a todos os cenários.

Algumas regras da Especificação, ditas não-Modais, têm por objetivo descrever informação condicional sobre os estados do sistema. Funcionam como as regras das Cláusulas de Horn, exceto por poderem ser aplicadas também a fatos de outros cenários além do Inicial (chamados de átomos modais).

O conjunto de fatos do Cenário Inicial junto com as regras não-modais da Especificação formam um conjunto de Cláusulas Definidas que descrevem inteiramente a informação relevante sobre aquele cenário.

Para estabelecer as mudanças de cenário, existem regras na Especificação que descrevem as relações entre cenários diferentes. Essas regras estão intimamente relacionadas às ações e são chamadas de Cláusulas Definidas Modais. Intuitivamente, a aplicação de uma regra modal em qualquer cenário que satisfaça seu antecedente (a pré-condição de aplicação da ação), leva o sistema a um novo cenário onde o conseqüente é verificado.

Uma abordagem semelhante é encontrada em [dC86]. Nesse trabalho é proposto o Molog, um formalismo semelhante ao Prolog que suporta modalidades nas cláusulas. O sistema de dedução varia de acordo com o sistema modal subjacente que se pretende implementar, no trabalho citado, esse é o Sistema Modal S5. A nossa abordagem é diferente porque é voltada para o problema da geração de planos e também porque em virtude de possuir características modais específicas possui também um mecanismo de dedução mais simples e próximo ao da linguagem PROLOG.

Para facilitar a compreensão do mecanismo de funcionamento será apresentada a princípio uma versão proposicional que posteriormente será estendida para o caso de Primeira Ordem.

## 6.2 Linguagem Proposicional das Cláusulas de Horn Modais

DEFINIÇÃO 6.1 : *Alfabeto*

- *Um conjunto infinito de constantes sentenciais;*
- *Um conjunto finito de nomes de ações, denominado Ac;*
- *Símbolos Lógicos:* $\leftarrow$, [, ] *e* ,.

DEFINIÇÃO 6.2 : *Símbolos da linguagem*

- *Literal Positivo: constante sentencial*
- *Átomo Puro: literal positivo*
- *Ação: Se* $\alpha \in Ac$ *então* $[\alpha]$ *é uma ação (e corresponde ao cenário atingido a partir do cenário inicial pela execução de* $\alpha$*)*
- *Átomo Modal:* $[\alpha_1][\alpha_2]...[\alpha_n]q$*, onde* $q$ *é um átomo puro,* $n > 0$ *e cada* $\alpha_i \in Ac$
- *Átomo: átomo puro ou um átomo modal*

Antes de prosseguirmos com as definições dos outros elementos da linguagem, é importante efetuarmos algumas considerações sobre os conceitos apresentados na seção

anterior, a fim de possibilitar uma melhor compreensão de como os mesmos se relacionam com o sistema que estamos apresentando.

Conforme mencionado anteriormente, existem algumas propriedades que são observadas apenas no instante inicial do sistema. Outras características, no entanto, têm por objetivo descrever o funcionamento do sistema e são observadas em todos os instantes (Especificação).

Para simplificar as futuras definições, a seqüência $[\alpha_1]\,[\alpha_2]\,\ldots\,[\alpha_n]$ será representada por $[\![\alpha]\!]^n$. A letra grega $\epsilon$ representará a seqüência vazia de ações. Diz-se que o *tamanho do prefixo* do átomo modal $[\![\alpha]\!]^n q$ é igual a $n$. A composição das seqüências $[\![\alpha]\!]^n$ e $[\![\beta]\!]^k$, $[\![\alpha]\!]^n[\![\beta]\!]^k$, é a seqüência $[\alpha_1]\,[\alpha_2]\,\ldots\,[\alpha_n]\,[\beta_1]\,[\beta_2]\,\ldots\,[\beta_k]$.

Uma cláusula definida do tipo

$$q \leftarrow a_1, a_2, \ldots, a_m$$

procura estabelecer a informação de que a propriedade $q$ é verificada em qualquer cenário que possua as propriedades $a_1$ e $a_2$ e ... e $a_m$.

Uma cláusula definida modal do tipo

$$[\![\alpha]\!]^n q \leftarrow a_1, a_2, \ldots, a_m$$

intuitivamente estabelece a informação de que se o cenário $C$ possui as propriedades $a_1$ e $a_2$ e ... e $a_m$, então o cenário resultante da execução de $[\![\alpha]\!]^n$ em $C$ possui a propriedade $q$.

Formalmente, a Linguagem das Cláusulas de Horn Modais para o caso proposicional é definida a partir dos símbolos introduzidos na Definição 6.2.

DEFINIÇÃO 6.3 :

*Uma* Cláusula Definida não-Modal *é uma regra do tipo*

$$q \leftarrow a_1, a_2, \ldots, a_m \quad (m \geq 0)$$

*onde cada* $a_i$ $(1 \leq i \leq m)$ *é um átomo e* $q$ *é um átomo puro.*

*Um* fato *é uma Cláusula Definida não-Modal cujo antecedente é vazio, isto é,* $m = 0$.

*Uma* Cláusula Vazia, *denotada por* $\Box$, *possui o antecedente e o conseqüente vazios e representa uma contradição.*

*Uma* Cláusula Definida Modal *é uma regra do tipo*

$$[\![\alpha]\!]^n q \leftarrow a_1, a_2, \ldots, a_m \quad (m \geq 0)$$

*onde cada* $a_i$ $(1 \leq i \leq m)$ *é um átomo e* $[\![\alpha]\!]^n q$ *é um átomo modal. Diz-se que a seqüência* $[\![\alpha]\!]^n$ *é a* parte modal do conseqüente *ou* prefixo.

*Uma* Cláusula Objetivo *é uma cláusula da forma*

$$\leftarrow a_1, a_2, \ldots, a_m \quad (m \geq 0)$$

*onde cada* $a_i$ $(1 \leq i \leq m)$ *é um átomo.*

*Um* Conjunto Definido Modal *é um conjunto formado por Cláusulas Definidas Modais e não-Modais.*

*Um* Conjunto quase-Definido Modal *é um Conjunto Definido Modal acrescido de exatamente uma cláusula objetivo.*

*Uma* Cláusula de Horn Modal *é uma cláusula definida modal, uma cláusula definida não-modal ou uma cláusula objetivo.*

*A* Linguagem Proposicional das Cláusulas de Horn Modais *é o conjunto formado por todas as Cláusulas de Horn Modais proposicionais.*

### 6.2.1 Prova de Expressões da Linguagem

Essa seção tem por objetivo introduzir o conceito de prova de Expressões da Linguagem. Ou seja, como é possível deduzir novas propriedades a partir dos fatos e regras descritos inicialmente.

A representação de um problema no Prolog Modal de Ação pode ser visualizada na Figura 6.1.

### 6.2.2 Prova de um átomo $[\![\alpha]\!]^n q$

A noção de prova de um átomo descrita abaixo é baseada no mecanismo de obtenção de uma LSD($f$)-refutação visto no Capítulo 4, procurando dar um tratamento adequado às modalidades.

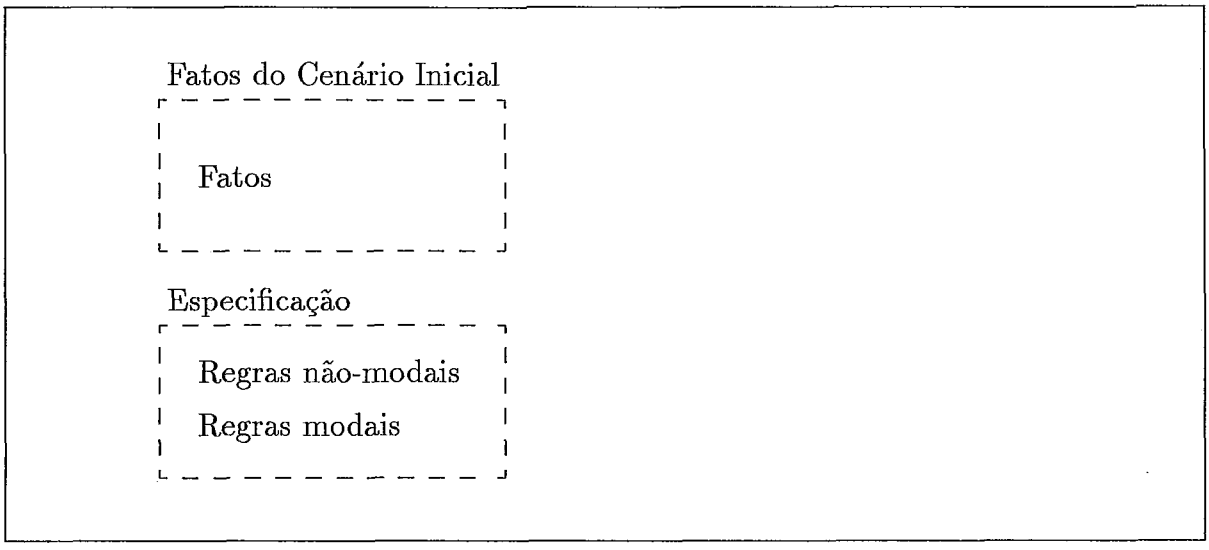



Figura 6.1: Estrutura da Representação de um Problema

DEFINIÇÃO 6.4 : *Função de seleção*

*Uma* função de seleção *f para cláusulas objetivo é aquela que mapeia cada cláusula objetivo G em um dos átomos de G. Se este for o átomo mais à esquerda de G, f é dita* função de seleção padrão.

Os exemplos utilizados para ilustrar o mecanismo de prova utilizarão a função de seleção padrão, em virtude de esta ser a utilizada na maioria dos sistemas PROLOG. Contudo, as definições se aplicam a qualquer função de seleção determinada, desde que ela satisfaça à Definição 6.2.2.

Apesar do nome, a regra da $f$-redução Modal não tem nenhuma relação com a regra da Redução do método de refutação por Eliminação de Modelos. O termo redução foi empregado numa alusão ao fato de que a aplicação da regra tende a reduzir o tamanho do prefixo dos átomos das cláusulas objetivo derivadas. Essa regra pode ser vista na realidade como a regra da Extensão do método da Resolução-LSD modificada para trabalhar com modalidades.

DEFINIÇÃO 6.5 : *Regra da f-redução Modal*

*Sejam G uma cláusula objetivo e D uma cláusula definida modal ou uma cláusula definida não-modal.*
*Se G é da forma*

$\leftarrow g_1, g_2, \ldots, g_n,$

$f(G) = g_i = [\![\beta]\!]^j [\![\alpha]\!]^k q \quad (j \geq 0, \quad k \geq 0) \quad e$

*D é da forma*

$$[\![\alpha]\!]^k q \leftarrow a_1, a_2, \ldots, a_m \quad (k \geq 0, \quad m \geq j),$$

*então o resultado de uma f-redução modal de G por D é uma nova cláusula objetivo G' da forma*

$$\leftarrow g_1, \ldots, g_{i-1}, [\![\beta]\!]^j a_1, [\![\beta]\!]^j a_2, \ldots, [\![\beta]\!]^j a_m, g_{i+1}, \ldots, g_n$$

Note que foi requerido que quando o prefixo do literal escolhido da cláusula objetivo for maior que o prefixo da cláusula de entrada, o antecedente desta seja não-vazio ($m \geq j$). Se isso não fosse exigido não haveria como impedir que as $f$-reduções modais de literais de cenários futuros se realizassem com os fatos do Cenário Inicial. Sejam, por exemplo

$g_i = [\alpha]q$ e

$D = q \leftarrow$

O antecedente de $D$ deveria ser provado no cenário $[\alpha]$, contudo o mesmo é vazio. Essa situação deve ser evitada porque $q$ é uma característica específica do Cenário Inicial (pois é um fato) e nesse caso não é possível provar $[\alpha]q$ a partir dessa cláusula.

Através da regra da $f$-redução modal introduzida anteriormente é possível efetuar a definição dos conceitos de LSD($f$)-refutação e LSD($f$)-dedução para Cláusulas de Horn Modais:

DEFINIÇÃO 6.6 : *LSD(f)-dedução modal e LSD(f)-refutação modal*

***LSD(f)-dedução modal***

*Sejam f uma função de seleção, S um conjunto quase-definido modal e G uma cláusula objetivo. Uma* LSD($f$)-dedução modal *de G a partir de S é uma seqüência* $C = (C_1, C_2, \ldots, C_n)$ *de cláusulas, terminando em G, isto é,* $C_n = G$ *e* $\exists r \leq n$ *tal que:*

*i) para todo* $i < r$, $C_i$ *é uma cláusula definida modal ou não-modal pertencente a S;*

*ii)* $C_r$ *é a cláusula objetivo de* $S$*;*

*iii) para todo* $i > r$, $C_i$ *é obtido pela aplicação da regra da* $f$*-redução modal a* $C_{i-1}$ *e* $C_j$*, para algum* $j < r$.

***LSD(f)-refutação modal***

*Uma* LSD($f$)-refutação modal *a partir de um conjunto quase-definido* $S$ *é uma LSD(f)-dedução modal da cláusula vazia a partir de* $S$.

Uma *prova* para um átomo $[\![\alpha]\!]^n q$ a partir de um conjunto definido modal $S$ é obtida se existir uma LSD($f$)-refutação modal a partir do conjunto $S \cup \{\leftarrow [\![\alpha]\!]^n q\}$. O Exemplo 6.2.2 procura ilustrar melhor a obtenção de uma LSD($f$)-refutação modal a partir de um conjunto:

EXEMPLO 6.1 : *Prova do átomo* $[\beta][\alpha]q$

***Conjunto Definido Modal*** $S$

***Fatos do Cenário Inicial***

1. $p$
2. $r$
3. $t$

***Especificação***

4. $[\alpha]q \leftarrow p,\ s.$
5. $[\beta]s \leftarrow r.$
6. $[\beta]p \leftarrow t.$

***LSD(f)-refutação modal a partir de*** $S \cup \{\leftarrow [\![\alpha]\!]^n q\}$

1. $r$
2. $t$

3. $[\alpha]q \leftarrow p,\ s.$

4. $[\beta]s \leftarrow r.$

5. $[\beta]p \leftarrow t.$

6. $\leftarrow [\beta][\alpha]q$

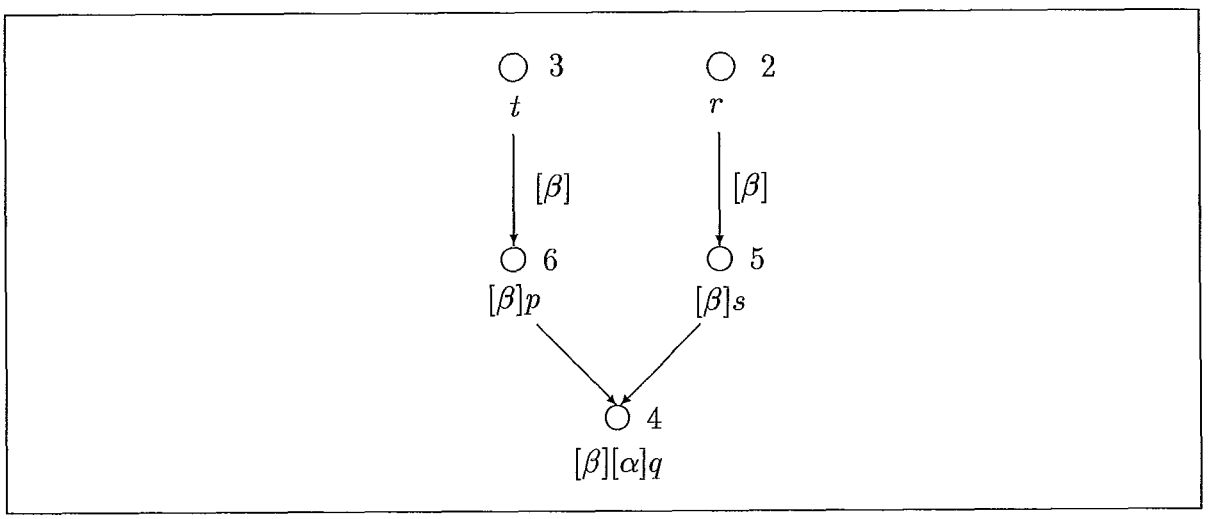
7. $\leftarrow [\beta]p[\beta]s$ *(4)*

8. $\leftarrow t[\beta]s$ *(6)*

9. $\leftarrow [\beta]s$ *(3)*

10. $\leftarrow r$ *(5)*

11. $\square$ *(2)*




No exemplo acima, vemos que a única forma de obtermos $q$ é executando a ação $[\alpha]$. Contudo, $[\alpha]$ só pode ser executada num cenário onde $p$ e $s$ sejam observados. $p$ é verificado no Cenário Inicial, mas $s$ não. A outra alternativa para $p$ é no cenário atingido pela execução de $[\beta]$. $[\beta]$ pode ser executada no cenário inicial pois $t$, sua única pré-condição é verificada nesse cenário. A prova agora prossegue para $s$, que é obtido também pela execução de $[\beta]$, num cenário onde $r$ seja observado. Como $r$ está no Cenário Inicial, a prova é bem-sucedida e a evolução do sistema pode ser observada na Figura 6.2.

Nas considerações efetuadas a seguir, o átomo selecionado da cláusula objetivo pela função de seleção será chamado de $g_i$ e o átomo do conseqüente da cláusula de entrada utilizada na $f$-redução modal daquela será chamado de $c$.

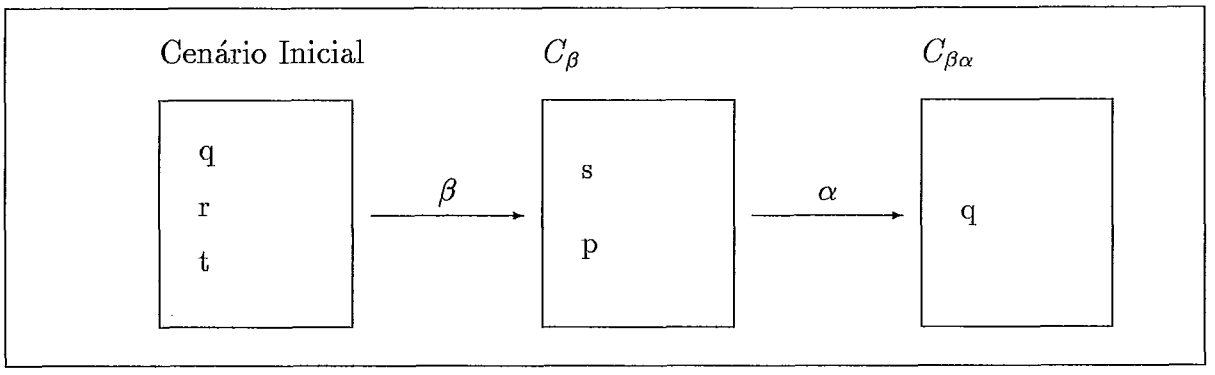



Figura 6.2: Mudanças de Cenário

O conceito de LSD($f$)-refutação modal recém-definido é empregado para obter provas de átomos modais específicos. Contudo, sob o ponto de vista do problema da geração de planos, ele deixa de fora as utilizações mais interessantes para o Prolog Modal de Ação, uma vez que o mesmo fica restrito a provar ou não propriedades, ao invés de procurar um cenário que possua as características desejadas.

Isso acontece porque na definição da regra da $f$-redução modal, foi requerido que o prefixo de $c$ fosse uma sub-cadeia final do prefixo de $g_i$. Seria interessante apenas informar um átomo e obter uma prova do mesmo no seu cenário ou em um cenário futuro, quando esta existisse. A adição dessa característica tornaria o Prolog Modal de Ação um gerador de planos, pois ao se descobrir um cenário onde o átomo seja verificado se está automaticamente obtendo um plano que leve o sistema a esse cenário.

Como o cenário da prova deve ser um cenário obtido a partir do cenário representado pelo prefixo do átomo, a idéia é definir uma nova regra para permitir também uma Extensão[1] da cláusula objetivo com uma cláusula de entrada quando o prefixo de $g_i$ for uma sub-cadeia final do prefixo de $c$.

O prefixo de $g_i$ deve ser ou uma sub-cadeia final do prefixo de $c$, quando então uma prova pra $g_i$ está sendo procurada num cenário atingido a partir do cenário atual; ou então o prefixo de $c$ deve ser uma sub-cadeia final do prefixo de $g_i$, quando então uma prova para $g_i$ está sendo procurada no cenário associado ao seu prefixo. Caso contrário, $g_i$ e $c$ se referem a propriedades de cenários atingidos por sub-planos de planos diferentes.

Se o prefixo de $c$ for uma sub-cadeia final do prefixo de $g_i$ teremos a seguinte situação:
Sejam $g_i$ da forma

[1]no sentido da Resolução-LSD

$[\![\alpha]\!]^k [\![\beta]\!]^l q$ e

$c$ da forma

$[\![\beta]\!]^l q$.

Para provar $g_i$, deve-se provar o antecedente da regra de $c$ no cenário $[\![\alpha]\!]^k$, porque se esse antecedente for provado nesse cenário, então $[\![\beta]\!]^l q$ também o é. Sendo então uma prova de $[\![\alpha]\!]^k [\![\beta]\!]^l q$. Esse caso é tratado pela regra da $f$-redução modal definida anteriormente.

Por outro lado, se o prefixo de $g_i$ for uma sub-cadeia final do prefixo de $c$, teremos a seguinte situação:
Sejam $g_i$ da forma

$[\![\beta]\!]^l q$ e

$c$ da forma

$[\![\alpha]\!]^k [\![\beta]\!]^l q$.

Nesse caso, uma prova para $g_i$ estará sendo procurada no cenário $[\![\alpha]\!]^k$. Como todos os átomos de um antecedente devem ser provados num mesmo cenário, os outros átomos de $G$ também devem ser provados no cenário $[\![\alpha]\!]^k$. Esse caso será tratado por uma nova regra de inferência denominada *regra da $f$-extensão modal.* Mais uma vez vale salientar que o termo extensão foi empregado em virtude de a aplicação da regra ocasionar um aumento do tamanho dos prefixos dos átomos de $G$.

DEFINIÇÃO 6.7 : *Regra da $f$-extensão Modal*

*Sejam $G$ uma cláusula objetivo e $D$ uma cláusula definida modal ou uma cláusula definida não-modal.*
*Se $G$ é da forma*

$\leftarrow g_1, g_2, \ldots, g_n,$

$f(G) = g_i = [\![\alpha]\!]^k q \quad (k \geq 0)$ *e*

*$D$ é da forma*

$$[\![\beta]\!]^j [\![\alpha]\!]^k q \leftarrow a_1, a_2, \ldots, a_m \quad (j > 0, \ k \geq 0)$$

*então o resultado de uma f-extensão modal de G por D é uma nova cláusula objetivo G' da forma*

$$\leftarrow [\![\beta]\!]^j g_1, \ldots, [\![\beta]\!]^j g_{i-1}, a_1, a_2, \ldots, a_m, [\![\beta]\!]^j g_{i+1}, \ldots, [\![\beta]\!]^j g_n$$

*A seqüência $[\![\beta]\!]^j$ é denominada de* seqüência de extensão de $G'$.

As definições de LSD($f$)-dedução modal e de LSD($f$)-refutação modal devem ser adaptadas a fim de possibilitarem a utilização da regra da $f$-extensão modal introduzida. Dessa forma,

DEFINIÇÃO 6.8 : *LSD(f)-dedução modal genérica e LSD(f)-refutação modal genérica*

***LSD(f)-dedução modal genérica***

*Sejam f uma função de seleção, S um conjunto quase-definido modal e G uma cláusula objetivo. Uma* LSD($f$)-dedução modal genérica *de G a partir de S é uma seqüência $C = (C_1, C_2, \ldots, C_n)$ de cláusulas, terminando em G, isto é, $C_n = G$, uma seqüência $A = ([\![s]\!]_0, [\![s]\!]_1, \ldots, [\![s]\!]_{n-r})$ de seqüências de ações e $\exists r \leq n$ tais que:*

*i)* $[\![s]\!]_0 = \epsilon$;

*ii) para todo $i < r$, $C_i$ é uma cláusula definida modal ou não-modal pertencente a S;*

*iii) $C_r$ é a cláusula objetivo de S;*

*iv) para todo $i > r$, $C_i$ é obtido pela aplicação da regra da f-redução modal ou da f-extensão modal a $C_{i-1}$ e $C_j$, para algum $j < r$ e*

1. *Se $C_i$ é obtido por f-redução modal, então $[\![s]\!]_{i-r} = [\![s]\!]_{i-r-1}$*
2. *Se $C_i$ é obtido por f-extensão modal, então $[\![s]\!]_{i-r} = [\![\beta]\!]^j [\![s]\!]_{i-r-1}$, onde $[\![\beta]\!]^j$ é a seqüência de extensão de $C_i$.*
3. *Se existe e $(r < e < n)$ tal que $C_e$ é obtido por f-redução modal a $C_{e-1}$ e $C_j$ para algum $j < r$ onde $C_j$ é um fato, então nenhum $C_i$ $(i > e)$ foi obtido por f-extensão modal*

***LSD(f)-refutação modal genérica***

*Uma* LSD($f$)-refutação modal genérica *a partir de um conjunto quase-definido $S$ é uma LSD($f$)-dedução modal genérica da cláusula vazia a partir de $S$.*

A seqüência $[\![s]\!]_{n-r}$ obtida numa LSD($f$)-dedução modal genérica corresponde ao plano que leva o sistema do Cenário Inicial a um cenário onde o átomo da prova é verificado. Note que em *iv.2* cada seqüência de extensão é adicionada ao início do plano até então obtido. Isso ocorre porque o sistema funciona de modo regressivo, em direção ao Cenário Inicial. Assim, a seqüência de extensão obtida na última aplicação de uma $f$-extensão modal da derivação corresponde à primeira seqüência de ações aplicada ao Cenário Inicial (porque é a $f$-extensão modal que realiza a transição para um cenário futuro). O exemplo a seguir utiliza o mesmo conjunto definido modal do Exemplo 6.2.2, nele o mecanismo de obtenção de uma LSD($f$)-refutação modal genérica pode ser observado (a função $f$ é a função de seleção padrão).

EXEMPLO 6.2 : *Prova genérica do átomo $q$*

***Conjunto Definido Modal $S$***

***Fatos do Cenário Inicial***

1. $p$
2. $r$
3. $t$

***Especificação***

4. $[\alpha]q \leftarrow p,\ s.$
5. $[\beta]s \leftarrow r.$
6. $[\beta]p \leftarrow t.$

***LSD(f)-refutação modal genérica a partir de $S \cup \{\leftarrow q\}$***

1. $r$

2. $t$

3. $[\alpha]q \leftarrow p, s.$

4. $[\beta]s \leftarrow r.$

5. $[\beta]p \leftarrow t.$

6. $\leftarrow q$

| | | |
|---|---|---|
| 7. $\leftarrow p, s$ | *EXT (4)* | $[\![s]\!]_1 = [\alpha]$ |
| 8. $\leftarrow t, [\beta]s$ | *EXT (6)* | $[\![s]\!]_2 = [\beta][\alpha]$ |
| 9. $\leftarrow [\beta]s$ | *RED (3)* | $[\![s]\!]_3 = [\beta][\alpha]$ |
| 10. $\leftarrow r$ | *RED (5)* | $[\![s]\!]_4 = [\beta][\alpha]$ |
| 11. $\square$ | *RED (2)* | $[\![s]\!]_5 = [\beta][\alpha]$ |

### 6.2.3 Uma Interpretação Modal para o Sistema Apresentado

O objetivo dessa seção é apresentar uma intuição, sob o ponto de vista das Lógicas Modais, das regras da $f$-extensão e $f$-redução apresentadas anteriormente. Por estarmos trabalhando com uma Lógica Modal de Ação, ao invés de uma modalidade do tipo $\square$, temos uma família de modalidades uma para cada ação.

Como se pretende que as regras da Especificação sejam verificadas em todos os estados do sistema, temos implicitamente a modalidade por sobre todas as suas regras. Isso decorre da regra da Necessitação, que é uma forma encontrada de propagar as informações acerca do funcionamento do sistema a todos os estados do mesmo. Por se referirem exclusivamente a propriedades do estado inicial do sistema, a regra da Necessitação não é aplicável aos fatos do Cenário Inicial.

A regra da $f$-redução modal está fundamentada no axioma $K$ dos Sistemas Modais:

$$K: \quad \square(p \rightarrow q) \rightarrow (\square p \rightarrow \square q)$$

Olhando mais de perto os elementos dessa regra:

$$g_i = [\![\beta]\!]^j [\![\alpha]\!]^k q$$

$$D = [\![\alpha]\!]^k q \leftarrow a_1,a_2,\ldots,a_m$$

$D$ é uma regra da Especificação, por conseguinte vale também no Cenário $[\![\beta]\!]^j$ e em particular temos

$$[\![\beta]\!]^j([\![\alpha]\!]^k q \leftarrow a_1,a_2,\ldots,a_m)$$

Agora pelo axioma $K$:

$$[\![\beta]\!]^j([\![\alpha]\!]^k q) \leftarrow [\![\beta]\!]^j(a_1,a_2,\ldots,a_m)$$

Ou seja, uma dedução para $[\![\beta]\!]^j[\![\alpha]\!]^k q$ (que é o próprio *sub-goal* $g_i$) é possível se for possível encontrar uma dedução para $[\![\beta]\!]^j(a_1,a_2,\ldots,a_m)$, que são exatamente os átomos adicionados à claúsula objetivo resultante da aplicação da regra.

A regra da $f$-extensão introduzida pode ser vista como uma forma de representar uma meta-pergunta ao provador. Não se pretende perguntar por um átomo específico, pois isso restringiria a busca ao próprio cenário do átomo. O que se pretende na realidade é submeter uma prova de uma família de átomos relacionados. Por exemplo, uma pergunta do tipo $[\alpha_1]q$, com a regra de $f$-extensão, corresponde a uma pergunta do tipo

$$[\alpha_1]q \cong \begin{cases} [\alpha_1]q \\ [\![\beta]\!]^1[\alpha_1]q \\ [\![\beta]\!]^2[\alpha_1]q \\ \vdots \end{cases}$$

onde $[\![\beta]\!]^1$ é uma seqüência de ações da Linguagem, de tamanho 1; $[\![\beta]\!]^2$ é uma seqüência de ações da Linguagem, de tamanho 2; e assim sucessivamente.

É possível estabelecer uma ordem de preferência de aplicação das regras. Supondo que uma aplicação da regra da $f$-extensão só fosse permitida quando não fosse possível obter uma refutação através de $f$-reduções, as provas seriam investigadas de cenário em cenário de forma que cenários com seqüências de ações menores seriam investigados antes. Isso decorre porque aplicações da regra da $f$-redução tentam efetuar a prova das propriedades no cenário relativo à seqüência de ações da parte modal do átomo correspondente, enquanto aplicações da regra da $f$-extensão buscam uma prova num cenário imediatamente seguinte ao atual.

## 6.3 Linguagem de Primeira Ordem das Cláusulas de Horn Modais

A seguir fazemos uma extensão da linguagem proposicional apresentada na Seção 6.2 para o caso de primeira ordem.

DEFINIÇÃO 6.9 : *Alfabeto*

- *Um conjunto de símbolos predicativos n-ários;*
- *Um conjunto infinito de variáveis;*
- *Um conjunto de símbolos funcionais n-ários;*
- *Um conjunto de constantes;*
- *Um conjunto finito de nomes de ações, denominado Ac;*
- *Símbolos Lógicos:* $\leftarrow$, [, ] *e* ,.

DEFINIÇÃO 6.10 : *Símbolos da linguagem*

- *Termo: é uma variável, uma constante do alfabeto, ou uma expressão da forma*

  $$f(t_1,t_2,\ldots,t_n)$$

  *onde* $f$ *é um símbolo funcional n-ário e* $t_1$, $t_2$, ..., $t_n$ *são termos do alfabeto.*

- *Literal Positivo: é uma expressão da forma*

  $$p(t_1,t_2,\ldots,t_n)$$

  *onde* $p$ *é um símbolo predicativo n-ário e* $t_1$, $t_2$, ..., $t_n$ *são termos*

- *Átomo Puro: é um literal positivo*

- *Ação: Se* $\alpha^n$ *é um nome de ação n-ário e* $t_1$, $t_2$, ..., $t_n$ *são termos, então* $[\alpha(t_1,t_2,\ldots,t_n)]$ *é uma ação.*

- *Átomo Modal:* $[\![\alpha]\!]^n q$, *onde* $q$ *é um literal positivo e* $n > 0$

- *Fatos: átomos puros da linguagem*

A definição das Cláusulas de Horn Modais para o caso de Primeira Ordem é a mesma dada anteriormente, exceto que seus constituintes são agora expressões de Primeira Ordem, conforme acima. As definição de prova para o caso de Primeira Ordem será dada diretamente para o caso genérico. Assim,

### 6.3.1 Prova de um átomo de Primeira Ordem $[\![\alpha]\!]^n q$

DEFINIÇÃO 6.11 : *Regra da f-redução Modal*

*Sejam G uma cláusula objetivo, D uma cláusula definida modal ou uma cláusula definida não-modal e $\psi$ uma renomeação de D em presença de G.*
*Se G é da forma*

$$\leftarrow g_1, g_2, \ldots, g_n,$$

$$f(G) = g_i = [\![\beta]\!]^j [\![\gamma]\!]^k q \quad (j \geq 0, \quad k \geq 0) \quad e$$

*$D\psi$ é da forma,*

$$[\![\alpha]\!]^k q' \leftarrow a_1, a_2, \ldots, a_m \quad (k \geq 0)$$

*e existe um umg $\theta$ de $\{[\![\gamma]\!]^k q, [\![\alpha]\!]^k q'\}$ então o resultado de uma f-redução modal de G por D é uma nova cláusula objetivo G' da forma*

$$\leftarrow (g_1, \ldots, g_{i-1}, [\![\beta]\!]^j a_1, [\![\beta]\!]^j a_2, \ldots, [\![\beta]\!]^j a_m, g_{i+1}, \ldots, g_n)\theta$$

DEFINIÇÃO 6.12 : *Regra da f-extensão Modal*

*Sejam G uma cláusula objetivo, D uma cláusula definida modal ou uma cláusula definida não-modal e $\psi$ uma renomeação de D em presença de G.*
*Se G é da forma*

$$\leftarrow g_1, g_2, \ldots, g_n,$$

$$f(G) = g_i = [\![\alpha]\!]^k q \quad (k \geq 0),$$

*D é da forma,*

$$[\![\beta]\!]^j[\![\gamma]\!]^k q' \leftarrow a_1,a_2,\ldots,a_m \quad (j > 0,\ k \geq 0)$$

*e existe um umg* $\theta$ *de* $\{[\![\alpha]\!]^k q, [\![\gamma]\!]^k q'\}$ *então o resultado de uma f-extensão modal de G por D é uma nova cláusula objetivo G' da forma*

$$\leftarrow ([\![\beta]\!]^j g_1,\ldots,[\![\beta]\!]^j g_{i-1}, a_1,a_2,\ldots,a_m, [\![\beta]\!]^j g_{i+1},\ldots,[\![\beta]\!]^j g_n)\theta$$

*A seqüência* $[\![\beta]\!]^j$ *é denominada de* seqüência de extensão de $G'$.

As definições de LSD($f$)-dedução modal genérica e de LSD$f$-refutação modal genérica são exatamente as mesmas dadas para o caso proposicional, exceto que agora os umgs intermediários obtidos devem ser levados em consideração nas derivações intermediárias:

DEFINIÇÃO 6.13 : *LSD(f)-dedução modal genérica e LSD(f)-refutação modal genérica*

***LSD(f)-dedução modal genérica***

*Sejam f uma função de seleção, S um conjunto quase-definido modal e G uma cláusula objetivo. Uma* LSD($f$)-dedução modal genérica *de G a partir de S é uma seqüência* $C = (C_1,C_2,\ldots,C_n)$ *de cláusulas, terminando em G, isto é,* $C_n = G$, *uma seqüência* $A = ([\![s]\!]_0, [\![s]\!]_1, \ldots, [\![s]\!]_{n-r})$ *de seqüências de ações, uma seqüência de umgs* $\theta_1, \theta_2, \ldots, \theta_{n-r}$ *e* $\exists r \leq n$ *tais que:*

*i)* $[\![s]\!]_0 = \epsilon$ *e* $\theta_0 = \varepsilon$;

*ii) para todo* $i < r$, $C_i$ *é uma cláusula definida modal ou não-modal pertencente a S;*

*iii)* $C_r$ *é a cláusula objetivo de S;*

*iv) para todo* $i > r$, $C_i$ *é obtido pela aplicação da regra da f-redução modal ou da f-extensão modal a* $C_{i-1}$ *e* $C_j$, *para algum* $j < r$ *e*

1. *Se* $C_i$ *é obtido por f-redução modal, então* $[\![s]\!]_{i-r} = ([\![s]\!]_{i-r-1})\theta_{i-r}$.
2. *Se* $C_i$ *é obtido por f-extensão modal, então* $[\![s]\!]_{i-r} = ([\![\beta]\!]^j[\![s]\!]_{i-r-1})\theta_{i-r}$, *onde* $[\![\beta]\!]^j$ *é a seqüência de extensão de* $C_i$.

***LSD(f)-refutação modal genérica***

*Uma* LSD($f$)-refutação modal genérica *a partir de um conjunto quase-definido $S$ é uma LSD($f$)-dedução modal genérica da cláusula vazia a partir de $S$.*

Através da regra de inferência acima é possível efetuar a definição dos conceitos de LSD($f$)-refutação e LSD($f$)-dedução para Cláusulas de Horn Modais:

### 6.3.2 Um Exemplo de Representação

Para ilustrar a representação de um problema e o funcionamento do mecanismo de prova consideremos o exemplo do *Mundo dos Blocos.* Nesse exemplo, o mundo consiste de uma superfície (uma mesa, por exemplo) sobre a qual estão alguns blocos de madeira. A posição horizontal dos mesmos não é relevante. Um bloco pode estar sobre a mesa, ou sobre exatamente um dos outros blocos. O problema consiste em, dada uma configuração inicial dos blocos, achar uma seqüência de ações que, quando executadas mude a disposição dos blocos até atingirem uma configuração desejada.

No nosso exemplo, utilizaremos três predicados para representar as possíveis posições dos blocos: $mesa(X)$, para indicar que o bloco $X$ está sobre a mesa; $limpo(X)$, quando não houver nenhum bloco sobre o bloco $X$; e $sobre(X,Y)$ para indicar que o bloco $X$ está sobre o bloco $Y$. Além disso, sempre que o bloco $X$ estiver sobre a mesa e limpo dizemos que ele está *livre* ($livre(X)$). O predicado $dif$ é utilizado para descrever a desigualdade entre blocos. Assim, $dif(X,Y)$ indica que o bloco $X$ é diferente do bloco $Y$.

Na configuração inicial, o bloco $c$ está sobre a mesa, o bloco $b$ está sobre o bloco $c$ e o bloco $a$ está sobre o bloco $b$. Desejamos encontrar uma seqüência de ações que leve o mundo a um estado onde o bloco $b$ esteja *livre.* Note pela Figura 6.3 que mais de um estado satisfazem esta exigência.

Definimos ainda duas ações para efetuar as modificações de cenários:

- $[emp(X,Y)]$ para empilhar o bloco $X$ sobre o bloco $Y$; e
- $[des(X,Y)]$ para desempilhar o bloco $X$ de cima do bloco $Y$.

As condições necessárias para que essas operações possam ser executadas podem ser vistas na representação do problema abaixo, onde foram incluídos apenas os aspectos relevantes e também regras de *frame*, para descrever as características que não

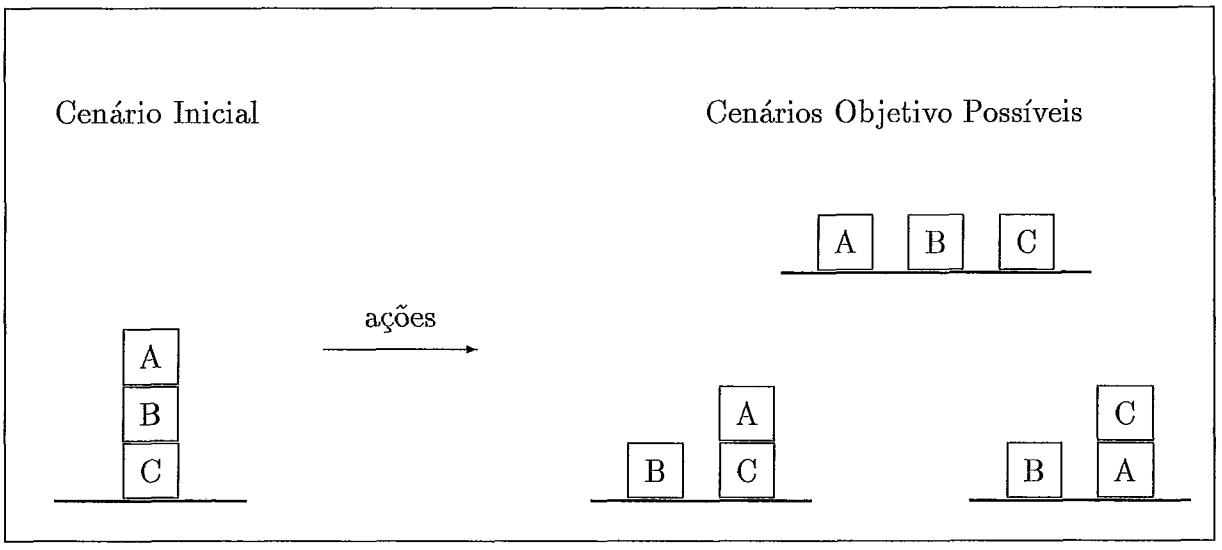



Figura 6.3: Visualização do Problema

são alteradas pela execução das ações. $W$, $X$, $Y$ e $Z$ são variáveis e $f$ é a função de seleção padrão. No sufixo da refutação são indicados para cada nova cláusula objetivo derivada:

(1) Por que regra a cláusula derivada foi obtida

(2) Qual cláusula auxiliar da refutação foi utilizada

(3) A renomeação $\psi$ da cláusula auxiliar em presença da cláusula objetivo anterior

(4) O umg utilizado na unificação da cláusula auxiliar e da cláusula objetivo anterior

(5) A seqüência de ações do plano obtida nesse passo da derivação

**Representação do Problema: Conjunto Definido Modal $S$**

**Fatos do Cenário Inicial**

1. $dif(a, b)$
2. $dif(a, c)$
3. $dif(b, a)$
4. $dif(b, c)$
5. $dif(c, a)$

6. $dif(c, b)$

7. $limpo(a)$

8. $mesa(c)$

9. $sobre(a, b)$

10. $sobre(b, c)$

**Especificação**

11. $livre(X) \leftarrow mesa(X), limpo(X)$

12. $[des(X,Y)]mesa(X) \leftarrow limpo(X), sobre(X,Y)$

13. $[des(X,Y)]limpo(Y) \leftarrow limpo(X), sobre(X,Y)$

14. $[des(X,Y)]limpo(Z) \leftarrow limpo(Z)$

15. $[des(X,Y)]mesa(Z) \leftarrow mesa(Z)$

16. $[des(X,Y)]sobre(W,Z) \leftarrow dif(W,X), sobre(W,Z)$

17. $[emp(X,Y)]sobre(X,Y) \leftarrow dif(X,Y), mesa(X), limpo(X), limpo(Y)$

18. $[emp(X,Y)]limpo(Z) \leftarrow dif(Z,Y), limpo(Z)$

19. $[emp(X,Y)]mesa(Z) \leftarrow dif(Z,Y), mesa(Z)$

20. $[emp(X,Y)]sobre(W,Z) \leftarrow sobre(W,Z)$

EXEMPLO 6.3 : *Uma LSD(f)-refutação modal genérica a partir de* $S \bigcup \{\leftarrow livre(b)\}$

1. $dif(b, a)$

2. $limpo(a)$

3. $sobre(a, b)$

4. $sobre(b, c)$

5. $livre(X) \leftarrow mesa(X), limpo(X)$

6. $[des(X,Y)]mesa(X) \leftarrow limpo(X), sobre(X,Y)$

7. $[des(X,Y)]limpo(Y) \leftarrow limpo(X), sobre(X,Y)$

8. $[des(X,Y)]limpo(Z) \leftarrow limpo(Z)$

9. $[des(X,Y)]sobre(W,Z) \leftarrow dif(W,X), sobre(W,Z)$

10. $\leftarrow livre(b)$

11. $\leftarrow mesa(b), limpo(b)$

    *RED 5* $\psi = \varepsilon$ $\theta = \{X/b\}$ $[\![s]\!]_1 = \epsilon$

12. $\leftarrow limpo(b), sobre(b,Y)[des(b,Y)]limpo(b)$

    *EXT 6* $\psi = \varepsilon$ $\theta = \{X/b\}$ $[\![s]\!]_2 = [des(b,Y)]$

13. $\leftarrow limpo(X), sobre(X,b), [des(X,b)]sobre(b,Y), [des(X,b)][des(b,Y)]limpo(b)$

    *EXT 6* $\psi = \{Y/y_1\}$ $\theta = \{y_1/b\}$ $[\![s]\!]_3 = [des(X,b)][des(b,Y)]$

14. $\leftarrow sobre(a,b), [des(a,b)]sobre(b,Y), [des(a,b)][des(b,Y)]limpo(b)$

    *RED 2* $\psi = \varepsilon$ $\theta = \{X/a\}$ $[\![s]\!]_4 = [des(a,b)][des(b,Y)]$

15. $\leftarrow [des(a,b)]sobre(b,Y), [des(a,b)][des(b,Y)]limpo(b)$

    *RED 3* $\psi = \varepsilon$ $\theta = \varepsilon$ $[\![s]\!]_5 = [des(a,b)][des(b,Y)]$

16. $\leftarrow dif(b,a), sobre(b,Y), [des(a,b)][des(b,Y)]limpo(b)$

    *RED 9* $\psi = \{Y/y_1\}$ $\theta = \{X/a, y_1/b, W/b, Z/Y\}$ $[\![s]\!]_6 = [des(a,b)][des(b,Y)]$

17. $\leftarrow sobre(b,Y), [des(a,b)][des(b,Y)]limpo(b)$

    *RED 1* $\psi = \varepsilon$ $\theta = \varepsilon$ $[\![s]\!]_7 = [des(a,b)][des(b,Y)]$

18. $\leftarrow [des(a,b)][des(b,c)]limpo(b)$

    *RED 4* $\psi = \varepsilon$ $\theta = \{Y/c\}$ $[\![s]\!]_8 = [des(a,b)][des(b,c)]$

19. $\leftarrow [des(a,b)]limpo(b)$

    *RED 8* $\psi = \varepsilon$ $\theta = \{X/b, Y/c, Z/b\}$ $[\![s]\!]_9 = [des(a,b)][des(b,c)]$

20. $\leftarrow limpo(a), sobre(a,b)$

    *RED 7* $\psi = \varepsilon$ $\theta = \{X/a, Y/b\}$ $[\![s]\!]_{10} = [des(a,b)][des(b,c)]$

*21.* $\leftarrow$ $sobre(a, b)$

*RED 2* $\psi = \varepsilon$ $\theta = \varepsilon$ $[\![s]\!]_{11} = [des(a, b)][des(b, c)]$

*22.* $\square$

*RED 3* $\psi = \varepsilon$ $\theta = \varepsilon$ $[\![s]\!]_{12} = [des(a, b)][des(b, c)]$

Como $livre(b)$ não está no Cenário Inicial, começamos pela Especificação, utilizando a cláusula (4) da refutação e fazendo $X = b$:

$$livre(b) \leftarrow mesa(b),\ limpo(b).$$

Então $mesa(b)$, $limpo(b)$ devem ser verificados num mesmo cenário. $mesa(b)$ não está no Cenário Inicial, então utilizando a cláusula (5) e fazendo $X = b$:

$$[des(b, Y)]mesa(b) \leftarrow limpo(b),\ sobre(b, Y)$$

O objetivo agora é provar $limpo(b)$, $sobre(b, Y)$ num mesmo cenário, o restante do *Goal* anterior deve ser provado no cenário $[des(b, Y)]$, ou seja, $[des(b, Y)]limpo(b)$ uma vez que uma prova para $mesa(b)$ está sendo procurada no cenário $[des(b, Y)]$. Como $limpo(b)$ não está no Cenário Inicial, utilizamos a cláusula (6), desta vez com $Y = b$. Note que a variável $Y$ de (6) deve ser antes renomeada para uma variável $y_1$ pois $Y$ já aparece em (11). A instância da cláusula (6) é a seguinte:

$$[des(X, b)]limpo(b) \leftarrow limpo(X),\ sobre(X, b)$$

Mais uma vez uma extensão foi utilizada e portanto os literais restantes de (11) devem ser provados no cenário relativo à seqüência de extensão de (12), nesse caso $[des(X, b)]$. É isso que pode ser observado na cláusula (12) resultante.

É fácil verificar que fazendo $X = a$, uma prova de $limpo(a)$, $sobre(a, b)$ é encontrada no Cenário Inicial, sendo então as cláusulas (13) e (14) obtidas por redução modal. O restante do *Goal* é agora $[des(a, b)]sobre(b, Y), [des(a, b)][des(b, Y)]limpo(b)$. Pela cláusula (8), com $X' = a$, $Y' = b$, $W' = b$ e $Z' = Y$ (renomeações adequadas são necessárias e podem ser vistas na substituição $\psi$ de (15), o *Goal* resultante é:

$$dif(b, a), sobre(b, Y)[des(a, b)][des(b, Y)]limpo(b)$$

Note que, supondo domínio constante, até a desigualdade $dif(X, Y)$, teria que ser levada aos outros cenários via regras de *frame*. Nesse caso, a desigualdade é procurada no Cenário Inicial e não houve necessidade de utilização dessas regras. Para uma melhor ilustração desse problema, veja o exemplo da solução do Problema dos Três Blocos no final do Capítulo.

Passa-se à prova de $sobre(b, Y)$. Com $Y = c$, a mesma é obtida por redução no Cenário Inicial, restando provar apenas $[des(a, b)][des(b, c)]limpo(b)$. Novamente por redução, utilizando a cláusula (7), fazendo $X = b$, $Y = c$ e $Z = b$:

$$[des(a, b)][des(b, c)]limpo(b) \leftarrow [des(a, b)]limpo(b)$$

E o novo objetivo é agora $[des(a, b)]limpo(b)$.

Utilizando a regra (6), com $X = a$ e $Y = b$:

$$[des(a, b)]limpo(b) \leftarrow limpo(a),\ sobre(a, b)$$

Mas $limpo(a)$ e $sobre(a, b)$ estão no Cenário Inicial. Como a cláusula (21) é a cláusula vazia, uma refutação foi obtida e o Plano gerado corresponde à seqüência $[des(a, b)][des(b, c)]$.

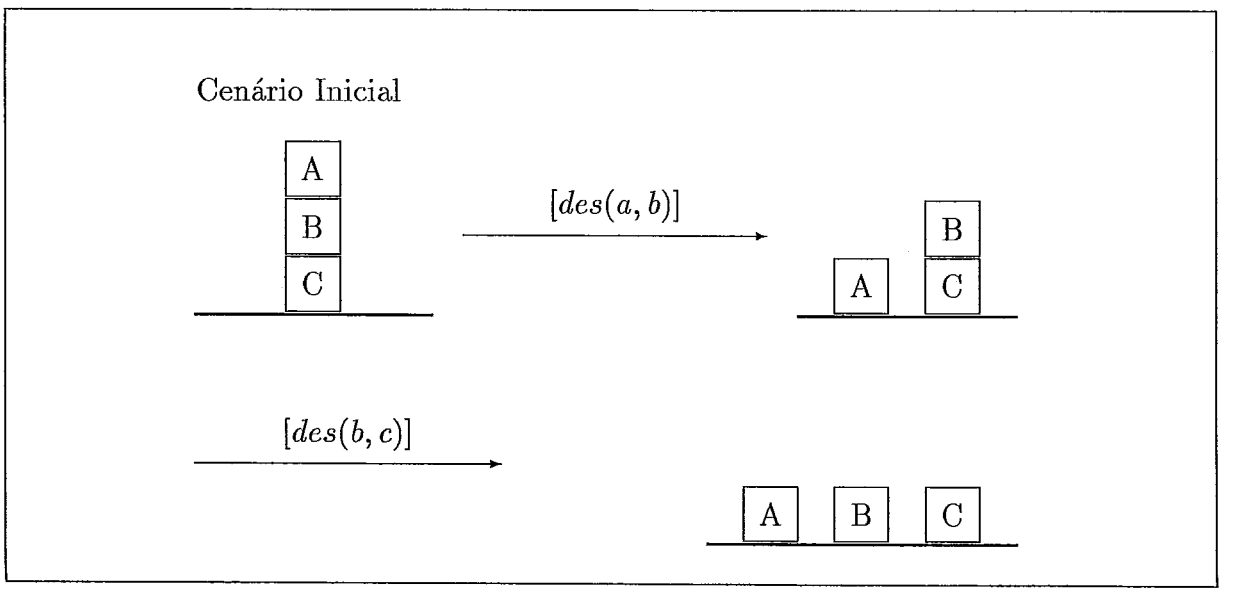



Figura 6.4: Uma solução para o problema

Conforme mencionado anteriormente, algumas regras ditas de *frame* tiveram que ser adicionadas para possibilitar uma solução para o exemplo. Isso decorre do chamado

*problema de frame*. Ao se definir os efeitos de uma ação sobre um mundo é necessário se definir não apenas as propriedades que mudam por sua execução mas também aquelas que não são afetadas pela mesma.

## 6.4 Solução para o Problema dos Três blocos

Essa seção apresenta uma representação e uma solução para o Problema dos três blocos visto no Capítulo 5. Os significados pretendidos para os predicados são os mesmos vistos nas seções anteriores. Esse problema, por ser um pouco mais complexo, exigiu a utilização de regras de *Frame* também para as desiguldades, conforme pode ser visto nas Cláusulas 13 e 19 da Especificação. A função de seleção utilizada é novamente a função de seleção padrão. Para relembrar, o problema pode ser visualizado na Figura abaixo:

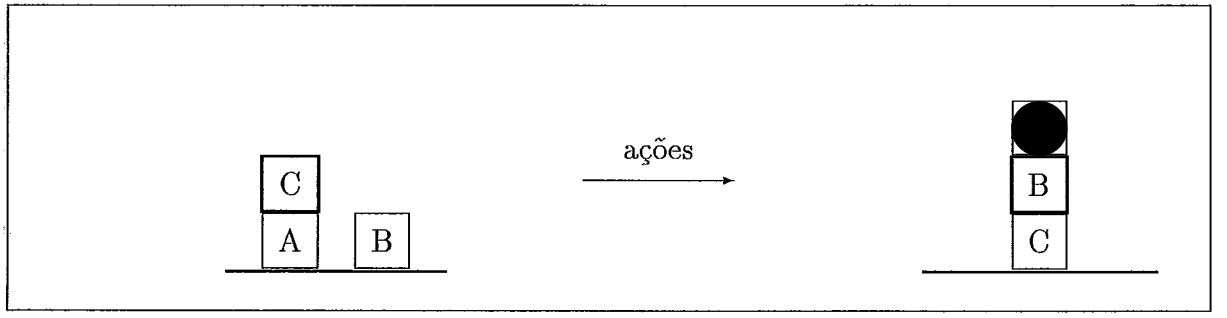



Figura 6.5: Cenário Inicial e Cenário Objetivo do problema

**Representação do Problema: Conjunto Definido Modal $S$**

**Fatos do Cenário Inicial**

1. $dif(a, b)$
2. $dif(a, c)$
3. $dif(b, a)$
4. $dif(b, c)$
5. $dif(c, a)$
6. $dif(c, b)$

7. $limpo(c)$
8. $mesa(a)$
9. $sobre(c, a)$
10. $limpo(b)$
11. $mesa(b)$

**Especificação**

12. $livre(X) \leftarrow mesa(X), limpo(X)$
13. $[des(X,Y)]dif(W,Z) \leftarrow dif(W,Z)$
14. $[des(X,Y)]mesa(X) \leftarrow limpo(X), sobre(X,Y)$
15. $[des(X,Y)]limpo(Y) \leftarrow limpo(X), sobre(X,Y)$
16. $[des(X,Y)]limpo(Z) \leftarrow limpo(Z)$
17. $[des(X,Y)]mesa(Z) \leftarrow mesa(Z)$
18. $[des(X,Y)]sobre(W,Z) \leftarrow dif(W,X), sobre(W,Z)$
19. $[emp(X,Y)]dif(W,Z) \leftarrow dif(W,Z)$
20. $[emp(X,Y)]sobre(X,Y) \leftarrow dif(X,Y), mesa(X), limpo(X), limpo(Y)$
21. $[emp(X,Y)]limpo(Z) \leftarrow dif(Z,Y), limpo(Z)$
22. $[emp(X,Y)]mesa(Z) \leftarrow dif(Z,Y), mesa(Z)$
23. $[emp(X,Y)]sobre(W,Z) \leftarrow sobre(W,Z)$

A seguir temos uma LSD($f$)-refutação modal genérica a partir do conjunto $S \cup \{\leftarrow sobre(a,b), sobre(b,c)\}$. Pois o objetivo é encontrar um cenário onde o bloco $a$ esteja sobre o bloco $b$ e o bloco $b$ sobre o bloco c. Esse cenário corresponderá ao Cenário atingido a partir do Cenário Inicial pela execução da seqüência de ações da dedução da cláusula vazia.

EXEMPLO 6.4 : *Uma LSD(f)-refutação modal genérica*

1. $dif(a, b)$
2. $dif(a, c)$
3. $dif(b, c)$
4. $limpo(c)$
5. $mesa(a)$
6. $sobre(c, a)$
7. $limpo(b)$
8. $mesa(b)$
9. $[des(X, Y)]dif(W, Z) \leftarrow dif(W, Z)$
10. $[des(X, Y)]limpo(Y) \leftarrow limpo(X), sobre(X, Y)$
11. $[des(X, Y)]limpo(Z) \leftarrow limpo(Z)$
12. $[des(X, Y)]mesa(Z) \leftarrow mesa(Z)$
13. $[emp(X, Y)]dif(W, Z) \leftarrow dif(W, Z)$
14. $[emp(X, Y)]sobre(X, Y) \leftarrow dif(X, Y), mesa(X), limpo(X), limpo(Y)$
15. $[emp(X, Y)]limpo(Z) \leftarrow dif(Z, Y), limpo(Z)$
16. $[emp(X, Y)]mesa(Z) \leftarrow dif(Z, Y), mesa(Z)$
17. $[emp(X, Y)]sobre(W, Z) \leftarrow sobre(W, Z)$
18. $\leftarrow sobre(a, b), sobre(b, c)$
19. $\leftarrow dif(a, b), mesa(a), limpo(a), limpo(b), [emp(a, b)]sobre(b, c)$

    $EXT\ 14 \quad \psi = \varepsilon \qquad \theta = \{X/a, Y/b\}$

    $[\![s]\!]_1 = [emp(a, b)]$
20. $\leftarrow dif(a, b), [emp(X, Y)]mesa(a), [emp(X, Y)]limpo(a), [emp(X, Y)]limpo(b),$
    $[emp(X, Y)][emp(a, b)]sobre(b, c)$

    $EXT\ 13 \quad \psi = \varepsilon \qquad \theta = \{W/a, Z/a\}$

    $[\![s]\!]_2 = [emp(X, Y)][emp(a, b)]$

*21.* $\leftarrow dif(a,b),[des(X_1,Y_1)][emp(X,Y)]mesa(a),[des(X_1,Y_1)][emp(X,Y)]limpo(a),$
$[des(X_1,Y_1)][emp(X,Y)]limpo(b),[des(X_1,Y_1)][emp(X,Y)][emp(a,b)]sobre(b,c)$

*EXT 9* $\psi = \{X/X_1, Y/Y_1\}$ $\theta = \{W/a, Z/a\}$
$[\![s]\!]_3 = [des(X_1,Y_1)][emp(X,Y)][emp(a,b)]$

*22.* $\leftarrow [des(X_1,Y_1)][emp(X,Y)]mesa(a),[des(X_1,Y_1)][emp(X,Y)]limpo(a),$
$[des(X_1,Y_1)][emp(X,Y)]limpo(b),[des(X_1,Y_1)][emp(X,Y)][emp(a,b)]sobre(b,c)$

*RED 1* $\psi = \varepsilon$ $\theta = \varepsilon$
$[\![s]\!]_4 = [des(X_1,Y_1)][emp(X,Y)][emp(a,b)]$

*23.* $\leftarrow [des(X_1,Y_1)]dif(a,Y),[des(X_1,Y_1)]mesa(a),[des(X_1,Y_1)][emp(X,Y)]limpo(a),$
$[des(X_1,Y_1)][emp(X,Y)]limpo(b),[des(X_1,Y_1)][emp(X,Y)][emp(a,b)]sobre(b,c)$

*RED 16* $\psi = \{X/X_2, Y/Y_2\}$ $\theta = \{X_2/X, Y_2/Y, Z/a\}$
$[\![s]\!]_5 = [des(X_1,Y_1)][emp(X,Y)][emp(a,b)]$

*24.* $\leftarrow dif(a,Y),[des(X_1,Y_1)]mesa(a),[des(X_1,Y_1)][emp(X,Y)]limpo(a),$
$[des(X_1,Y_1)][emp(X,Y)]limpo(b),[des(X_1,Y_1)][emp(X,Y)][emp(a,b)]sobre(b,c)$

*RED 9* $\psi = \{X/X_2, Y/Y_2\}$ $\theta = \{X_2/X_1, Y_2/Y_1, W/a, Z/Y\}$
$[\![s]\!]_6 = [des(X_1,Y_1)][emp(X,Y)][emp(a,b)]$

*25.* $\leftarrow [des(X_1,Y_1)]mesa(a),[des(X_1,Y_1)][emp(X,c)]limpo(a),$
$[des(X_1,Y_1)][emp(X,c)]limpo(b),[des(X_1,Y_1)][emp(X,c)][emp(a,b)]sobre(b,c)$

*RED 2* $\psi = \varepsilon$ $\theta = \{Y/c\}$
$[\![s]\!]_7 = [des(X_1,Y_1)][emp(X,c)][emp(a,b)]$

*26.* $\leftarrow mesa(a),[des(X_1,Y_1)][emp(X,c)]limpo(a),[des(X_1,Y_1)][emp(X,c)]limpo(b),$
$[des(X_1,Y_1)][emp(X,c)][emp(a,b)]sobre(b,c)$

*RED 12* $\psi = \{X/X_2\}$ $\theta = \{X_2/X_1, Y/Y_1, Z/a\}$
$[\![s]\!]_8 = [des(X_1,Y_1)][emp(X,c)][emp(a,b)]$

*27.* $\leftarrow [des(X_1,Y_1)][emp(X,c)]limpo(a),[des(X_1,Y_1)][emp(X,c)]limpo(b),$
$[des(X_1,Y_1)][emp(X,c)][emp(a,b)]sobre(b,c)$

*RED 5* $\psi = \varepsilon$ $\theta = \varepsilon$
$[\![s]\!]_9 = [des(X_1,Y_1)][emp(X,c)][emp(a,b)]$

*28.* $\leftarrow [des(X_1,Y_1)]dif(a,c),[des(X_1,Y_1)]limpo(a),[des(X_1,Y_1)][emp(X,c)]limpo(b),$
$[des(X_1,Y_1)][emp(X,c)][emp(a,b)]sobre(b,c)$

*RED 15* $\psi = \{X/X_2\}$ $\theta = \{X_2/X, Z/a, Y/c\}$

$[\![s]\!]_{10} = [des(X_1, Y_1)][emp(X, c)][emp(a, b)]$

*29.* $\leftarrow dif(a, c), [des(X_1, Y_1)]limpo(a), [des(X_1, Y_1)][emp(X, c)]limpo(b),$
$[des(X_1, Y_1)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 9* $\psi = \{X/X_2\}$ $\theta = \{X_2/X_1, Y/Y_1, W/a, Z/c\}$

$[\![s]\!]_{11} = [des(X_1, Y_1)][emp(X, c)][emp(a, b)]$

*30.* $\leftarrow [des(X_1, Y_1)]limpo(a), [des(X_1, Y_1)][emp(X, c)]limpo(b),$
$[des(X_1, Y_1)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 2* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{12} = [des(X_1, Y)][emp(X, c)][emp(a, b)]$

*31.* $\leftarrow limpo(X_1), sobre(X_1, a), [des(X_1, a)][emp(X, c)]limpo(b),$
$[des(X_1, a)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 10* $\psi = \{X/X_2\}$ $\theta = \{X_2/X_1, Y/a, Y_1/a\}$

$[\![s]\!]_{13} = [des(X_1, a)][emp(X, c)][emp(a, b)]$

*32.* $\leftarrow sobre(c, a), [des(c, a)][emp(X, c)]limpo(b),$
$[des(c, a)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 4* $\psi = \varepsilon$ $\theta = \{X_1/c\}$

$[\![s]\!]_{14} = [des(c, a)][emp(X, c)][emp(a, b)]$

*33.* $\leftarrow [des(c, a)][emp(X, c)]limpo(b), [des(c, a)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 6* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{15} = [des(c, a)][emp(X, c)][emp(a, b)]$

*34.* $\leftarrow [des(c, a)]dif(b, c), [des(c, a)]limpo(b), [des(c, a)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 15* $\psi = \{X/X_1\}$ $\theta = \{X_1/X, Y/c, Z/b\}$

$[\![s]\!]_{16} = [des(c, a)][emp(X, c)][emp(a, b)]$

*35.* $\leftarrow dif(b, c), [des(c, a)]limpo(b), [des(c, a)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 9* $\psi = X/X_1\}$ $\theta = \{X_1/c, Y/a, W/b, Z/c\}$

$[\![s]\!]_{17} = [des(c, a)][emp(X, c)][emp(a, b)]$

*36.* $\leftarrow [des(c, a)]limpo(b), [des(c, a)][emp(X, c)][emp(a, b)]sobre(b, c)$

*RED 3* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{18} = [des(c, a)][emp(X, c)][emp(a, b)]$

*37.* $\leftarrow$ $limpo(b), [des(c,a)][emp(X,c)][emp(a,b)]sobre(b,c)$

*RED 11* $\psi = \{X/X_1\}$ $\theta = \{X_1/c, Y/a, Z/b\}$

$[\![s]\!]_{19} = [des(c,a)][emp(X,c)][emp(a,b)]$

*38.* $\leftarrow$ $[des(c,a)][emp(X,c)][emp(a,b)]sobre(b,c)$

*RED 7* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{20} = [des(c,a)][emp(X,c)][emp(a,b)]$

*39.* $\leftarrow$ $[des(c,a)][emp(X,c)]sobre(b,c)$

*RED 17* $\psi = \{X/X_1\}$ $\theta = \{X_1/a, Y/b, W/b, Z/c\}$

$[\![s]\!]_{21} = [des(c,a)][emp(X,c)][emp(a,b)]$

*40.* $\leftarrow$ $[des(c,a)]dif(b,c), [des(c,a)]mesa(b), [des(c,a)]limpo(b), [des(c,a)]limpo(c)$

*RED 14* $\psi = \{X/X_1\}$ $\theta = \{X/b, Y/c, X_1/b\}$

$[\![s]\!]_{22} = [des(c,a)][emp(b,c)][emp(a,b)]$

*41.* $\leftarrow$ $dif(b,c), [des(c,a)]mesa(b), [des(c,a)]limpo(b), [des(c,a)]limpo(c)$

*RED 9* $\psi = \varepsilon$ $\theta = \{X/c, Y/a, W/b, Z/c\}$

$[\![s]\!]_{23} = [des(c,a)][emp(b,c)][emp(a,b)]$

*42.* $\leftarrow$ $[des(c,a)]mesa(b), [des(c,a)]limpo(b), [des(c,a)]limpo(c)$

*RED 3* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{24} = [des(c,a)][emp(b,c)][emp(a,b)]$

*43.* $\leftarrow$ $mesa(b), [des(c,a)]limpo(b), [des(c,a)]limpo(c)$

*RED 12* $\psi = \varepsilon$ $\theta = \{X/c, Y/a, Z/b\}$

$[\![s]\!]_{25} = [des(c,a)][emp(b,c)][emp(a,b)]$

*44.* $\leftarrow$ $[des(c,a)]limpo(b), [des(c,a)]limpo(c)$

*RED 8* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{26} = [des(c,a)][emp(b,c)][emp(a,b)]$

*45.* $\leftarrow$ $limpo(b), [des(c,a)]limpo(c)$

*RED 11* $\psi = \varepsilon$ $\theta = \{X/c, Y/a, Z/b\}$

$[\![s]\!]_{27} = [des(c,a)][emp(b,c)][emp(a,b)]$

*46.* $\leftarrow [des(c,a)]limpo(c)$

*RED 7* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{28} = [des(c,a)][emp(b,c)][emp(a,b)]$

*47.* $\leftarrow limpo(c)$

*RED 11* $\psi = \varepsilon$ $\theta = \{X/c, Y/a, Z/c\}$

$[\![s]\!]_{29} = [des(c,a)][emp(b,c)][emp(a,b)]$

*48.* $\square$

*RED 4* $\psi = \varepsilon$ $\theta = \varepsilon$

$[\![s]\!]_{30} = [des(c,a)][emp(b,c)][emp(a,b)]$

A solução para o problema dos três blocos acima demonstra que o método descrito não sofre as limitações dos sistemas lineares. Apesar disso, usando a função de seleção padrão cada componente do *Goal* é trabalhado isoladamente.

A refutação acima não necessariamente seria obtida numa implementenção em PROLOG do Prolog Modal de Ação (ainda que essa refutação se utilize da função de seleção padrão). O que ocorre é que o PROLOG utiliza uma estratégia de busca em profundidade, acarretando perda da Completude em alguns casos (por exemplo, entrando num ramo de derivação infinito). Uma alternativa seria mudar a estratégia para uma busca em Largura, ou estipular um limite máximo para a profundidade da mesma.

## 6.5 Conclusões

Neste Capítulo, apresentamos o Prolog Modal de Ação, um formalismo que pode ser utilizado no problema da geração de planos.

Por sua semântica procedural semelhante à do PROLOG, o mesmo se torna de fácil assimilação por parte de pessoas familiarizadas com aquela linguagem de programação.

O formalismo é baseado em conceitos de uma Lógica Modal de Ação aplicado ao ambiente da Programação em Lógica. Sua linguagem permite a utilização de modalidades acompanhando os literais. As cláusulas compostas por esses literais seguem as mesmas restrições de formação que as da Linguagem das Cláusulas de Horn.

Através da adaptação da regra da Extensão para tratar os literais com modalidade, foi desenvolvido um mecanismo para obter refutações a partir de um (assim chamado) Conjunto quase-Definido Modal. A regra resultante é baseada na Regra da Necessitação e do Axioma $K$ dos sistemas Modais.

Para tornar o mecanismo capaz de encontrar cenários onde propriedades desejadas fossem verificadas, uma nova regra denominada $f$-extensão modal foi introduzida.

Uma das principais deficiências desse formalismo é enfrentar o problema de *frame* o que força que ao se especificar a dinâmica de uma ação sejam explicitamente declaradas todas as propriedades que não são afetadas por sua execução. Tal deficiência motivou a análise dos mecanismos de Mudanças Epistemológicas em Conjuntos Definidos.

# Capítulo 7

# MUDANÇAS EPISTEMOLÓGICAS EM CONJUNTOS DE CLÁUSULAS DE HORN

## 7.1 Introdução

Ao propor a modelagem dos estados epistêmicos de um indivíduo através de conjuntos de sentenças, [Gär88] sugeriu dois critérios de racionalidade básicos a que tais conjuntos deveriam obedecer.

O primeiro diz respeito à consistência do conjunto e o segundo indica que as conseqüências lógicas das propriedades aceitas também devem ser aceitas. Esses dois critérios de racionalidade forneceriam então um parâmetro para guiar a forma pela qual o processo de inferência se realizaria.

Partindo-se desses critérios, a visualização de um Conjunto de Crenças modelando estados epistêmicos, governado por uma lógica parece a alternativa mais natural, pois uma lógica forneceria o aparato dedutivo formal satisfatório. O segundo critério de racionalidade associado à lógica faz com que os Conjuntos de Crenças sejam Teorias, e isso acarreta alguns inconvenientes, em especial na definição de funções de Contração para modelar mudanças epistemológicas.

O problema parece residir no fato de que ao se fechar a Teoria sobre um conjunto de sentenças nas quais se acredita, as sentenças que são derivadas passam a ter importância igual às do conjunto básico.

Tomemos como exemplo um Conjunto de Crenças $K$ tal que $A \in K$. Conforme

visto na Seção 3.5.1, para um elemento $B$ tal que $B \notin K$ e $\neg B \notin K$ então teremos que $B \rightarrow A \in K$ e $\neg B \rightarrow A \in K$, pois como $K$ é um Conjunto de Crenças, todas as conseqüências lógicas derivadas a partir de $A$ estarão em $K$. Ao se fazer a Contração de $K$ por $A$ ($K^-_A$), visando a uma mudança mínima no Conjunto de Crenças as duas implicações serão mantidas, pois nem $B$ nem $\neg B$ estão em $K$ e portanto não acarretam na conseqüência de $A$.

Um problema ainda maior decorre da utilização de funções de Contração *Maxi-choice* em Conjuntos de Crenças para obter Revisões. Uma Revisão do tipo $K^*_{\neg A}$, pela Identidade de Levi, equivale a $(K^-_A)^+_{\neg A}$. Como para qualquer $B$ da Linguagem, $B \rightarrow A \in K^-_A$ ou $\neg B \rightarrow A \in K^-_A$ (ou ambos), o Conjunto $K^*_{\neg A}$ definido dessa forma será maximal (ou seja, para qualquer sentença $B$ da Linguagem, ou $B \in K^*_{\neg A}$ ou $\neg B \in K^*_{\neg A}$), o que evidentemente não corresponde a nenhuma representação intuitiva de estados epistêmicos, perdendo a motivação inicial.

A questão é que não era o objetivo primordial a entrada das implicações no Conjunto de Crenças. A presença delas era suportada apenas pela existência de $A$, o que levaria a supor que uma vez retirado $A$, elas e todas as sentenças dessa forma introduzidas deveriam também ser descartadas.

Isso sugere que as mudanças epistemológicas seriam melhor representadas se executadas na Apresentação da Teoria, sendo as conseqüências lógicas desse conjunto usadas apenas para realizar inferências.

Suponhamos o caso em que temos um conjunto de sentenças $S$, tal que $B \in Cn(S)$ mas $B \notin S$. Se $B$ não é uma tautologia, a Contração de $Cn(S)$ por $B$, $Cn(S)^-_B$ vai implicar necessariamente na mudança de elementos de $S$, que são detectáveis analisando-se as deduções de $B$. Não parece haver uma razão suficientemente forte para justificar a aplicação das mudanças epistemológicas na Teoria em si.

Além do mais, a utilização do conceito de Apresentação de Teoria nos processos de Revisão fornece elementos estruturais extras que não estão disponíveis usando-se Teorias. Essa abordagem possibilita a definição de um conjunto de sentenças central a partir do qual é possível se deduzir tudo o que um indivíduo suporta como conclusão válida, além de estabelecer um limite hierárquico entre sentenças.

A idéia de se trabalhar com um conjunto central de sentenças não é nova. [Lak70] introduziu o conceito de *core*, que consistia de um conjunto de sentenças que não são passíveis de mudança quando alguma se torna necessária. Ao invés disso, as alterações

são efetuadas na estrutura por ele chamada de *protective belt.*

[BM91] propõe uma metodologia voltada para o problema da geração de planos, na qual estados do sistema são chamados julgamentos. Existe um conjunto de regras extra-lógicas que descrevem a interação entre os diversos estados do sistema. Essas regras efetuam as modificações na Apresentação da Teoria.

Na nossa abordagem, a descrição do estado inicial é feita através de um conjunto de fatos em Prolog. Existe um conjunto de regras ditas não-modais que podem ser aplicadas sobre esses fatos ou outras propriedades derivadas, fornecendo a dedução de novas propriedades, mas não existe um conjunto que contenha todas as propriedades de um cenário. Tais propriedades são verificadas a partir da aplicação das regras não-modais ao conjunto básico de fatos. É portanto sobre esses fatos e regras que as mudanças devem ser aplicadas.

## 7.2 Mudanças Epistemológicas em Conjuntos Definidos

### 7.2.1 Considerações Iniciais

Os Estados Epistêmicos nesse trabalho correspondem ao conjunto de propriedades verificadas num estado de um sistema. Esse conjunto de propriedades é denominado de Cenário e representado por um Conjunto Definido. Em função disso, algumas observações são importantes sobre a aplicação dos processos de Revisão de Crenças:

i) as mudanças epistemológicas devem ser realizadas na Apresentação da Teoria, que origina o conjunto de propriedades de um estado do sistema

ii) como estamos trabalhando com um subconjunto *menor* da Lógica de Primeira Ordem, o poder expressivo é mais reduzido. Em particular não é possível representar-se disjunções legítimas de propriedades, p.ex. $A \vee B$, ou literais negativos. O caso da Negação por Falha Finita traz algumas conseqüências interessantes ao processo de Revisão, que serão consideradas no final da seção.

### 7.2.2 Contrações em Conjuntos de Cláusulas de Horn

A definição de Contração com relação a uma proposição exige que a mesma não seja uma conseqüência lógica do Conjunto de Crenças resultante. No caso das Cláusulas de Horn, uma proposição só pode ser obtida como conseqüência se for um fato ou se for o conseqüente de uma regra cujo antecedente puder ser provado. Há dois casos a se considerar:

i) A proposição a ser contraída não é conseqüência do Cenário

Intuitivamente, a noção que se tem é que o Cenário deve permanecer inalterado, uma vez que nada há a se retirar. Agindo dessa forma, a função de Contração respeita o Princípio da Economia Informativa estabelecido por $(K^-3)$. As Contrações consideradas aqui assumem a verificação desse postulado.

ii) A proposição a ser contraída é conseqüência do Cenário

Nesse caso, a realização da Contração pode exigir duas fases: a retirada do literal propriamente dita, quando o literal a ser contraído é um fato; e a execução da Contração quando o literal é conseqüência da aplicação de uma ou mais regras. A primeira fase é trivial, sendo analisadas aqui apenas as alternativas de realização da segunda.

A retirada das próprias regras em si acarreta, em geral, maior perda de informação o que é desaconselhável pelo critério de economia informativa. Caso a mesma não seja adotada, todas as alternativas para a segunda fase recaem na primeira, pois em última instância serão retirados os fatos que suportam a conseqüência da regra.

Além disso, não parece razoável adotar esse procedimento se considerarmos que o objetivo primordial das regras era modelar o funcionamento do sistema, cuja mecânica se supunha manter estável durante as mudanças de Cenário. A retirada de Regras acarreta perda de informação não só no Cenário onde a Contração é realizada, mas também em todos os outros a partir deste.

Considere o exemplo abaixo:

**Exemplo 7.1** *Contração de livre(b)*

***Cenário***

*1. mesa(a)*

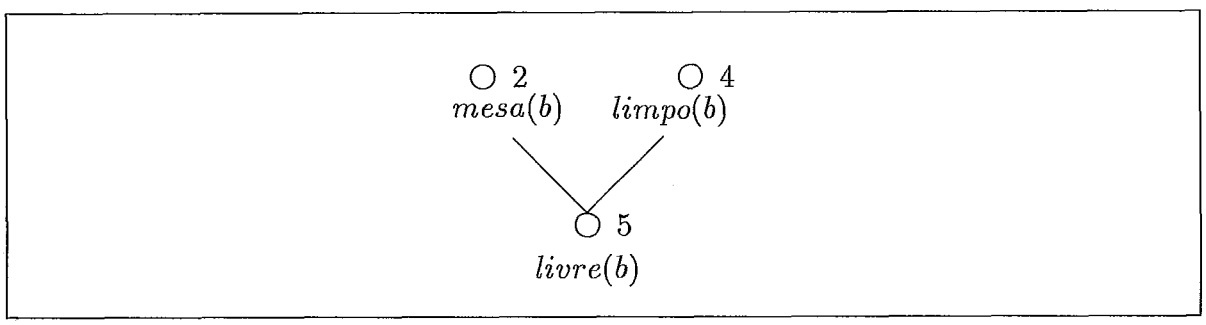
2. $mesa(b)$

3. $limpo(a)$

4. $limpo(b)$

5. $livre(X) \leftarrow mesa(X), limpo(X)$

***Árvore de Prova***




*Existem duas formas de se obter a Contração requerida no Cenário acima:*

*1) Retirar* $mesa(b)$ *e/ou* $limpo(b)$

*2) Retirar* $livre(X) \leftarrow mesa(X), limpo(X)$

No primeiro caso, dizemos que a Contração é *centralizada na instância* $b$, ocasionando a perda de $mesa(b)$ e/ou $limpo(b)$ que suportavam a conseqüência $livre(b)$.

No segundo caso, dizemos que a Contração é *centralizada no predicado*, pois para evitar $livre(b)$, a regra que o derivava foi retirada, acarretando a perda de outras instâncias de $livre(X)$, particularmente $livre(a)$. Mais formalmente, a especificação da propriedade $\forall x[mesa(x) \wedge limpo(x) \rightarrow livre(x)]$, também se perdeu. Nesse caso, não somente $livre(b)$ será contraído mas todas as instâncias representando blocos que pudessem estar livres nesse ou em cenários futuros.

O tipo de Contrações analisado aqui, portanto, será o baseado na centralização nas instâncias, pois é o que acarreta menor perda de informação. Considere o exemplo abaixo, que será utilizado para ilustrar melhor as alternativas que se seguirão. Note que um mesmo literal pode ter diversas árvores de prova.

**Exemplo 7.2** *Árvores de Prova do literal suspeito(mordomo)*

***Significados pretendidos para os predicados***

| | |
|---|---|
| *nao_gosta(X, Y)* | *X nao_gosta de Y* |
| *amigo(X, Y)* | *X é amigo de Y* |
| *morto(X)* | *X está morto* |
| *trabalha(X, Y)* | *X trabalha para Y* |
| *fichado(X)* | *X tem registro na polícia* |
| *suspeito(X)* | *X é suspeito do crime* |
| *conhece(X, Y)* | *X conhece Y* |
| *tem_antecedentes(X)* | *X tem antecedentes criminais* |

***Cenário Inicial*** - $C_0$

1. $amigo(mordomo, copeira)$
2. $morto(patrao)$
3. $nao\_gosta(copeira, patrao)$
4. $nao\_gosta(mordomo, patrao)$
5. $fichado(mordomo)$
6. $trabalha(mordomo, patrao)$
7. $suspeito(X) \leftarrow conhece(X, Y), morto(Y), nao\_gosta(X, Y).$
8. $suspeito(X) \leftarrow conhece(X, Y), morto(Y), tem\_antecedentes(X).$
9. $nao\_gosta(X, Y) \leftarrow amigo(X, Z), nao\_gosta(Z, Y).$
10. $conhece(X, Y) \leftarrow trabalha(X, Y).$
11. $tem\_antecedentes(X) \leftarrow fichado(X).$

*A Figura a seguir mostra as árvores de prova para o literal suspeito(mordomo) no Cenário Inicial.*

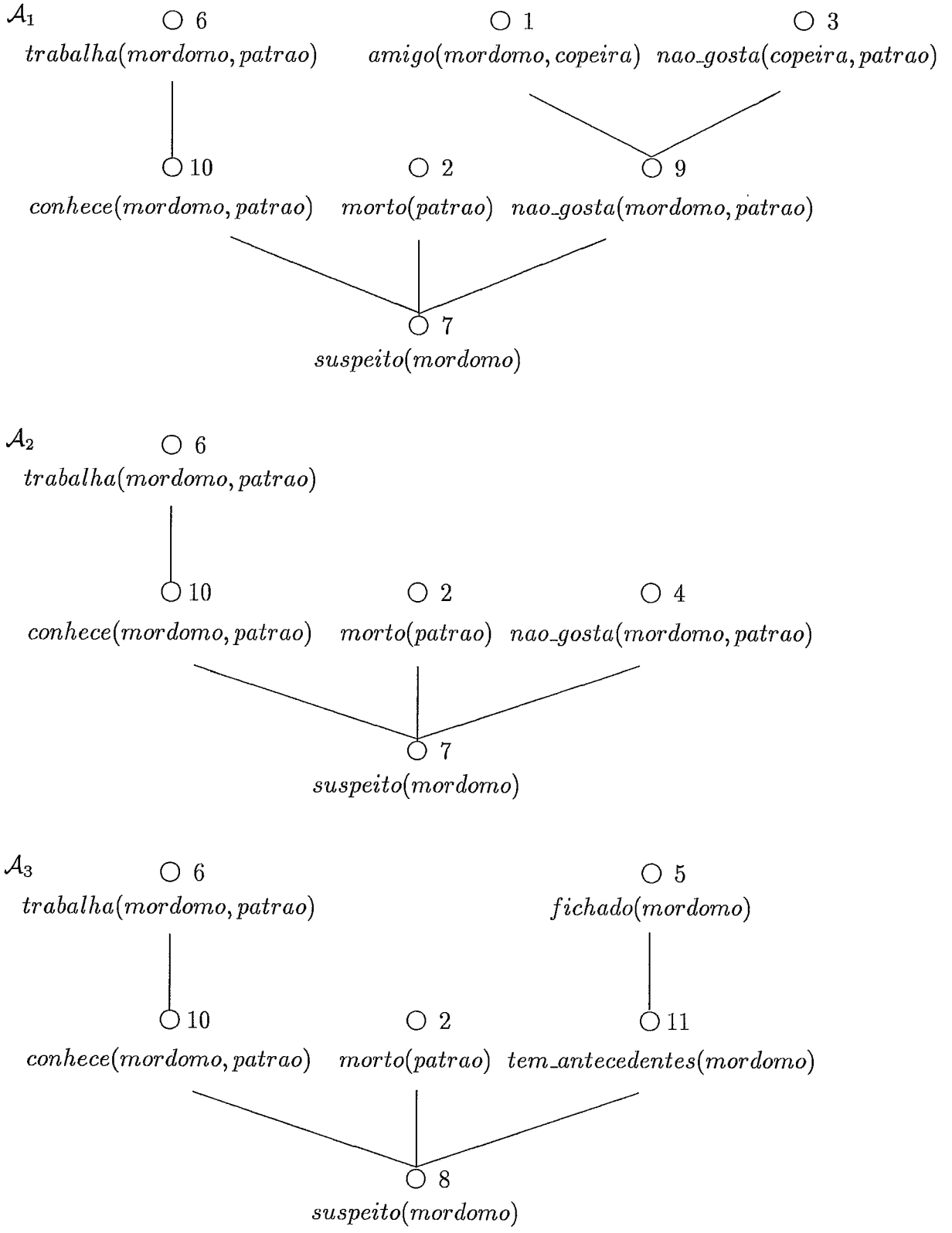


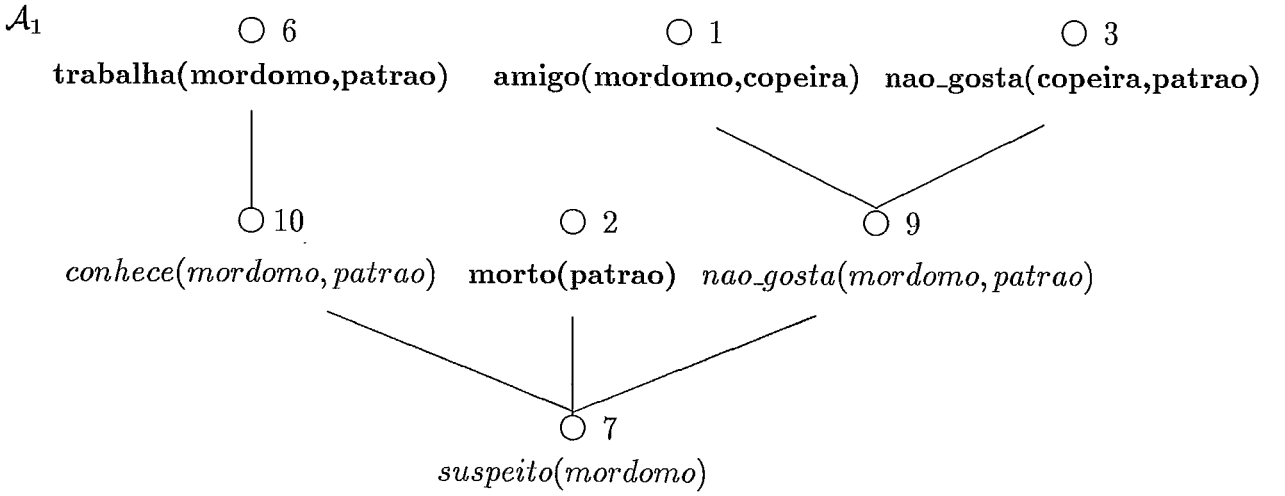

*Figura 7.1: Árvores de prova para o literal suspeito(mordomo) no Cenário Inicial*

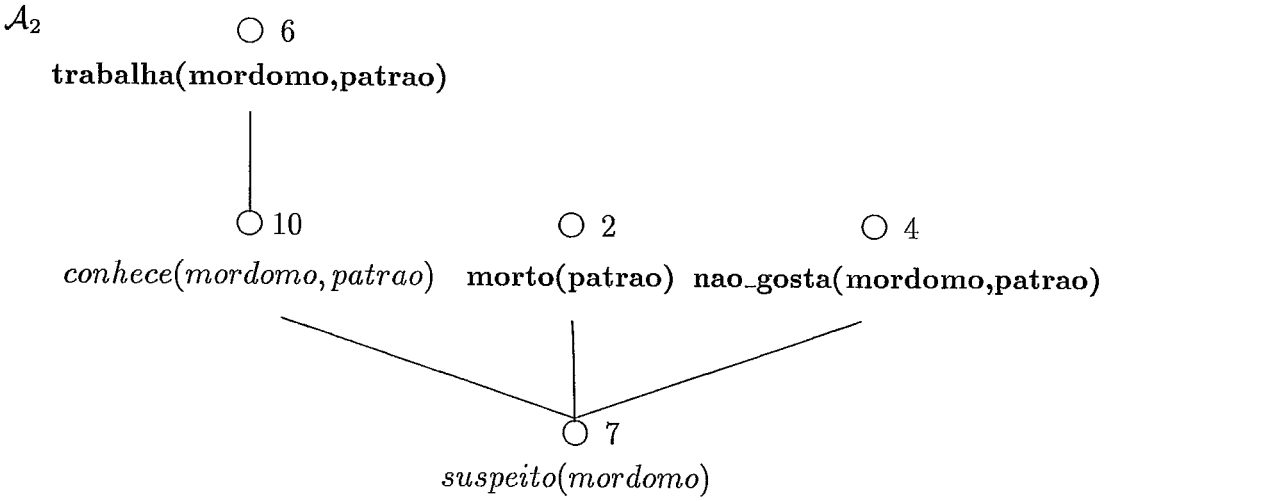
A primeira alternativa a analisar é o caso em que todos os fatos que suportam a conseqüência do literal são retirados. Para que isso seja atingido é necessário cortar todas as folhas da floresta de prova do literal.

### $C_{max}$ - Retirada máxima de fatos

Essa alternativa ocasiona a maior perda de informação dentre as Contrações centralizadas nas instâncias. Seu uso contudo pode ser utilizado quando se desejar obter uma Contração mais efetiva de propriedades, uma vez que futuras Expansões que indiretamente ocasionem a recuperação da propriedade recém-contraída são mais improváveis (veja seção 7.2.3).

Os literais a retirar para obter a Contração de *suspeito*(*mordomo*) no Cenário Inicial, segundo esse critério, estão em negrito na Figura abaixo:



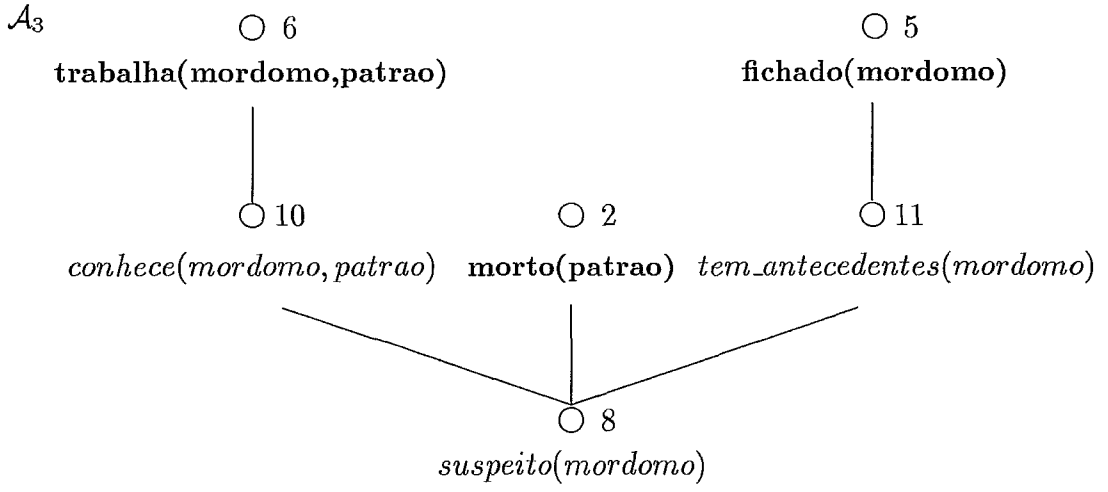



Figura 7.2: Literais a retirar utilizando o critério $C_{max}$

O conjunto de fatos a retirar de um conjunto definido $S$, a fim de se obter a Contração de um literal $L$, segundo o critério $C_{max}$ será denotado por $C_{max}(S : L)$. Dessa forma, $C_{max}(C_0 : suspeito(mordomo)) =$

$$\{fichado(mordomo), morto(patrao), amigo(mordomo, copeira),$$
$$trabalha(mordomo, patrao), nao\_gosta(copeira, patrao), nao\_gosta(mordomo, patrao)\}.$$

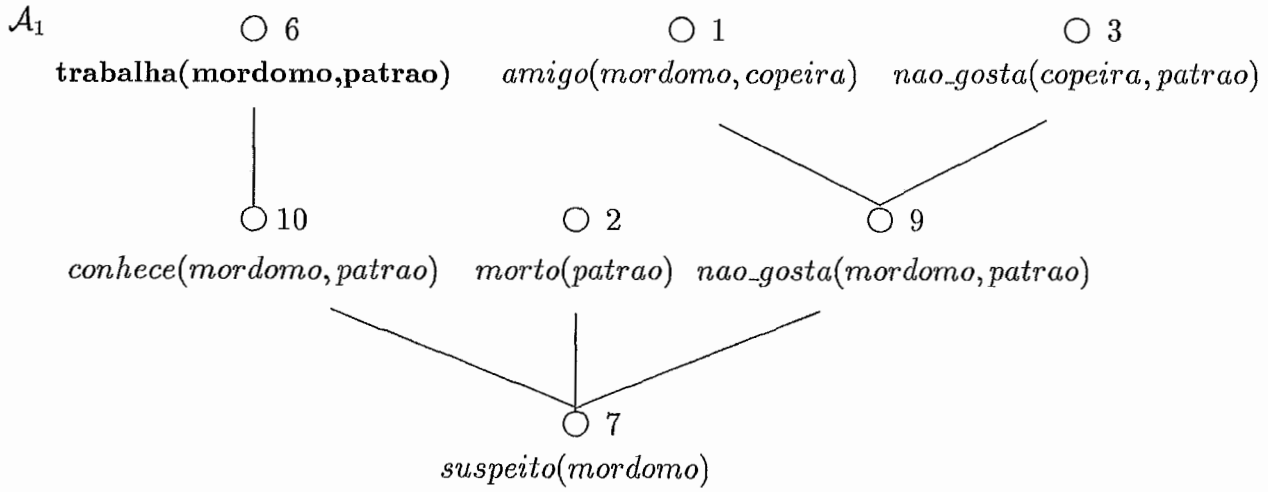
Esse tipo de Contração contraria o Princípio da Conservatividade que estabelece o conceito de Mudança Mínima nos processos de Revisão (veja seção 3.3).

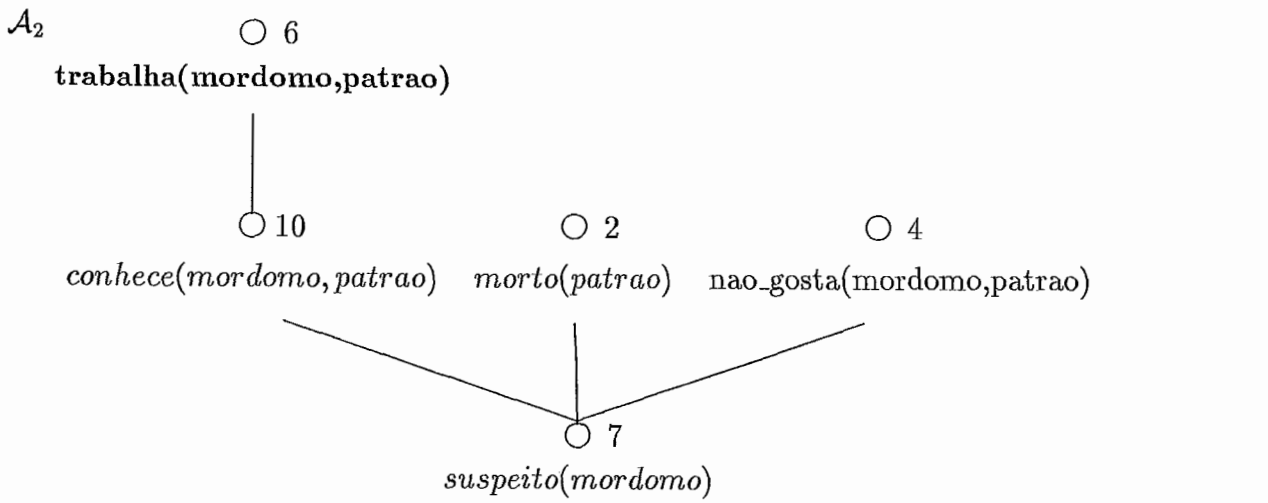
### $C_{esq}$ - Retirada dos fatos localizados mais à esquerda nas árvores de prova

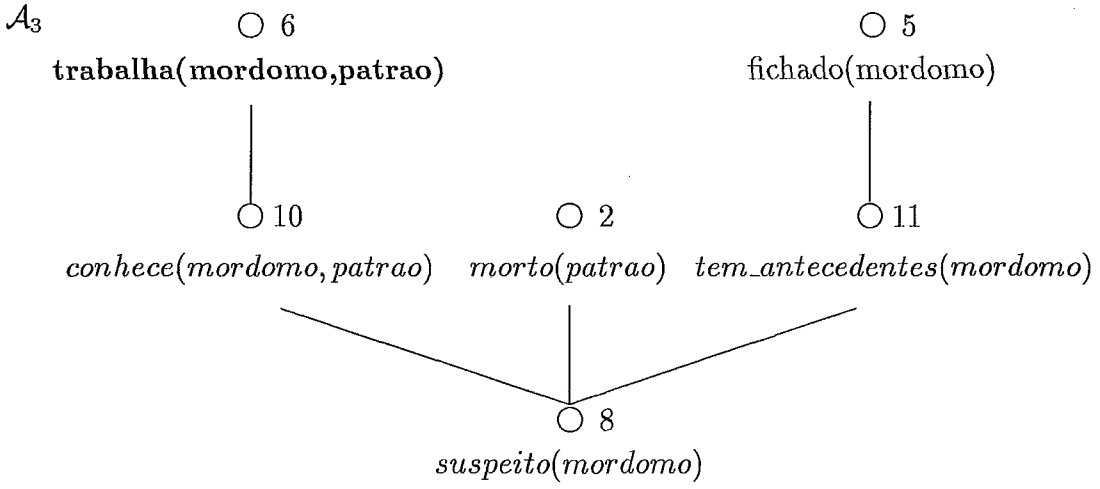
Como a conseqüência de uma regra só pode ser obtida se todos os literais do antecedente puderem ser provados, existem outras maneiras de obter a Contração, objetivando uma menor mudança no conjunto.

Vamos analisar o caso em que a Contração é realizada sempre em relação apenas ao literal mais à esquerda das árvores de prova. Uma motivação para isso é que a Linguagem de Programação PROLOG utiliza a função de seleção padrão, o que induz a um processo de busca em profundidade pela prova dos literais. Dessa forma, o literal localizado mais à esquerda nas árvores de prova é o primeiro fato a ser acessado. $C_{esq}$ é um critério particularmente adequado para implementação nesse tipo de sistema.

Mais uma vez é necessário percorrer toda a floresta de prova em busca agora apenas das folhas que estão na extremidade esquerda das árvores. Seguindo esse critério, apenas o literal $trabalha(mordomo, patrao)$ seria retirado a fim de se obter a mesma Contração de antes, ou seja, $C_{esq}(C_0 : suspeito(mordomo)) = \{trabalha(mordomo, patrao)\}$.






Esse método oferece um meio seguro de se obter uma Contração a um custo, em geral, bem mais baixo, dependendo da forma pela qual o problema for especificado. Além de verificar o Princípio da Conservatividade mencionado anteriormente.

### $C_{EE}$ - Retirada mínima com Entrincheiramento Epistêmico

O método anterior estabalece um critério de decisão entre literais a retirar no momento da Contração. É possível também que o próprio usuário estabeleça esse critério determinando uma ordenação de Entrincheiramento Epistêmico dos literais (ou dos predicados). Usando essa ordenação, o esquema abaixo forneceria um método para a realização da Contração através de $C_{EE}$.

Para cada árvore de prova $\mathcal{A}'$ para o literal $L$ a partir do Conjunto Definido $S$, $C_{EE}$ é definida da seguinte forma, iniciando-se o processo a partir da raiz de $\mathcal{A}'$:

Seja o nó atual $N$, rotulado pelo literal $L'$

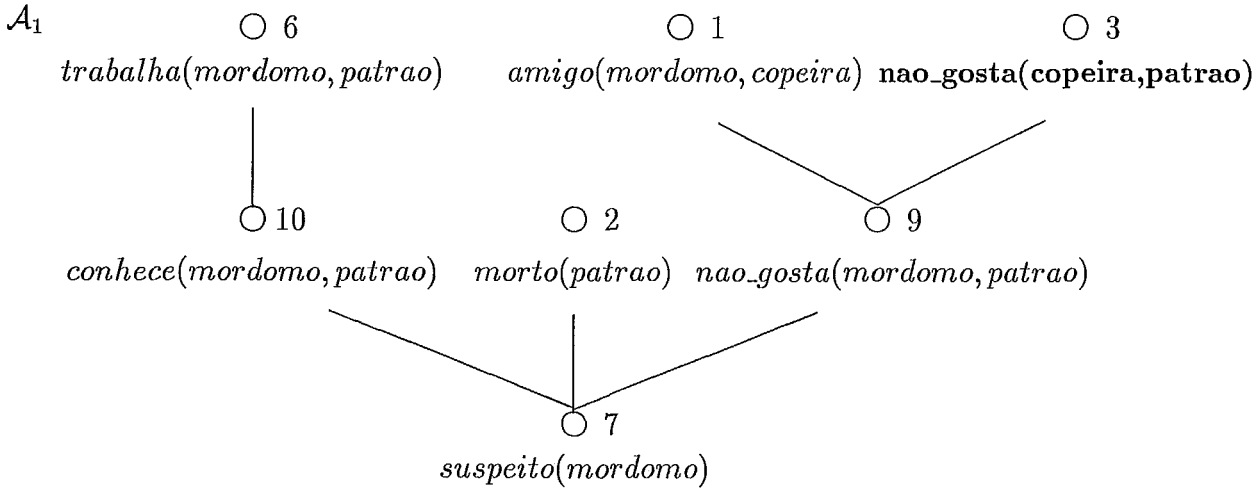
- Se $N$ é uma folha, então $C_{EE}(S : L)^{\mathcal{A}'} = \{L'\}$.
- Se $N$ não é uma folha, então seja $S'$ o conjunto formado por todos os literais que rotulam os filhos de $N$. $C_{EE}(S : L)^{\mathcal{A}'} = C_{EE}(S : L")^{\mathcal{A}"}$, onde $L"$ é o literal que possui o menor Entrincheiramento Epistêmico dentre os literais de $S'$ e $\mathcal{A}"$ é a árvore de prova de $L"$ sub-árvore de $\mathcal{A}'$.

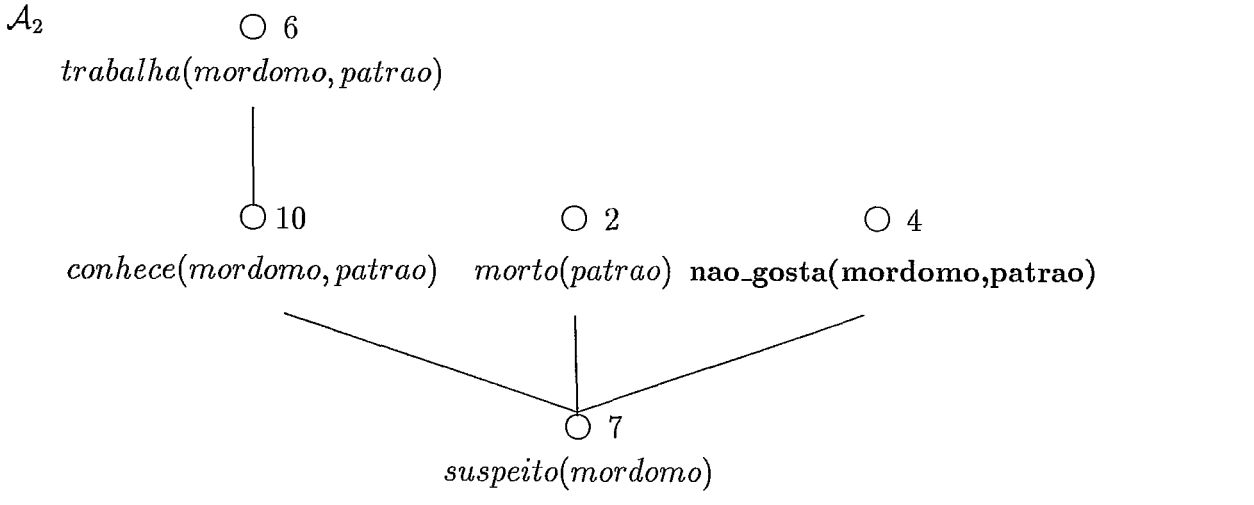
Suponha, ainda considerando o exemplo anterior, que a seguinte ordenação tivesse sido fornecida:

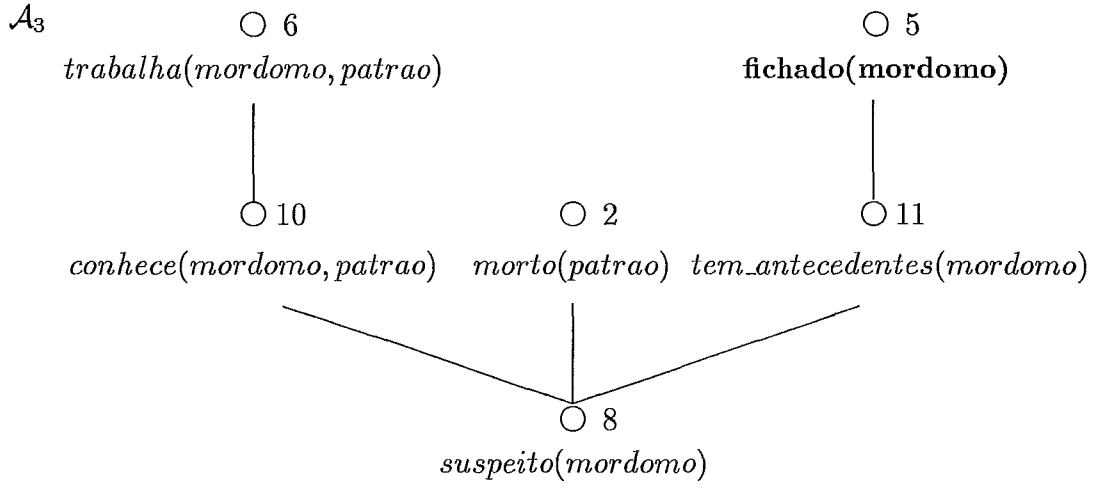
**Ordenação de Entrincheiramento Epistêmico**

$$\begin{array}{c} fichado(mordomo) \leq nao\_gosta(mordomo, patrao) \leq \\ nao\_gosta(copeira, patrao) \leq tem\_antecedentes(X) \leq \\ trabalha(mordomo, patrao) \leq conhece(X, Y) \leq \\ amigo(mordomo, copeira) \leq suspeito(X, Y) \leq morto(patrao) \end{array} \tag{7.1}$$

Para ordenação acima, os literais a retirar segundo $C_{EE}$, podem ser vistos em negrito nas árvores de prova a seguir:






Dessa forma, segundo a ordenação 7.1, $C_{EE}(C_0 : suspeito(mordomo)) =$

$\{nao\_gosta(copeira, patrao), nao\_gosta(mordomo, patrao), fichado(mordomo)\}$.

$C_{EE}$ é um critério que possibilita ao usuário estabelecer uma preferência em relação à retirada de literais.

Como uma floresta de prova pode ser mapeada em todos os ramos de sucesso de uma árvore de LSD($f$)-refutação de um Conjunto quase-Definido, é possível demonstrar o seguinte resultado:

**Proposição 7.1** *As funções de Contração $C_{max}$, $C_{esq}$ e $C_{EE}$ induzem a verificação do quarto postulado das Contrações proposto por [Gär88]:*

$(K^-4)$ *Se* $\nvdash A$ *então* $A \notin S_A^-$

**Prova:**

O conceito de tautologia não se aplica aqui (por restrição da Linguagem), logo deve-se provar que $A \notin S_A^-$, ou em outras palavras, que $A$ não é uma conseqüência lógica de $S_A^-$.

Se $A$ é uma conseqüência lógica de $S_A^-$ então existe uma LSD($f$)-refutação a partir de $S_A^- \cup \{\leftarrow A\}$. Vamos então provar que não existe uma LSD($f$)-refutação a partir de $S_A^- \cup \{\leftarrow A\}$, onde $-$ é definida como em $C_{esq}$ (a prova para $C_{max}$ e $C_{EE}$ segue diretamente).

1º caso: $A$ não é uma conseqüência lógica de $S$, então não existe uma LSD($f$)-refutação de

$S \bigcup \{\leftarrow A\}$ e como as funções respeitam $(K^-3)$ (veja caso (i), no início desta seção), $S_A^- = S$, e portanto não existe uma LSD($f$)-refutação a partir de $S_A^-$.

2º caso: $A$ é uma conseqüência lógica de $S$. Nesse caso, pelo Teorema da Completude para a Resolução-LSD, existe uma LSD($f$)-refutação de $S \bigcup \{\leftarrow A\}$. Seja a seqüência $C = C_1, C_2, \ldots, C_n$ uma LSD($f$)-refutação de $S \bigcup \{\leftarrow A\}$. Pela definição de LSD($f$)-refutação, $\exists r < n$ tal que $C_r = \leftarrow A$. A prova será feita por Indução no tamanho $k$ do sufixo da refutação $C_{r+1}, \ldots, C_n$.

Para $k = 1$, então $r + 1 = n$ e a cláusula auxiliar utilizada na $f$-extensão de $D_n$ é da forma

$$A' \leftarrow$$

tal que existe um umg de $\{A, A'\}$. Por definição, a árvore de prova de $A$ só tem um nó rotulado por $A$ e nesse caso, $C_{esq}(S : A) = \{A\}$, de modo que $S_A^- = S - \{A \leftarrow\}$ e conseqüentemente não poderá haver uma LSD($f$)-refutação de $S_A^- \bigcup \{\leftarrow A\}$.

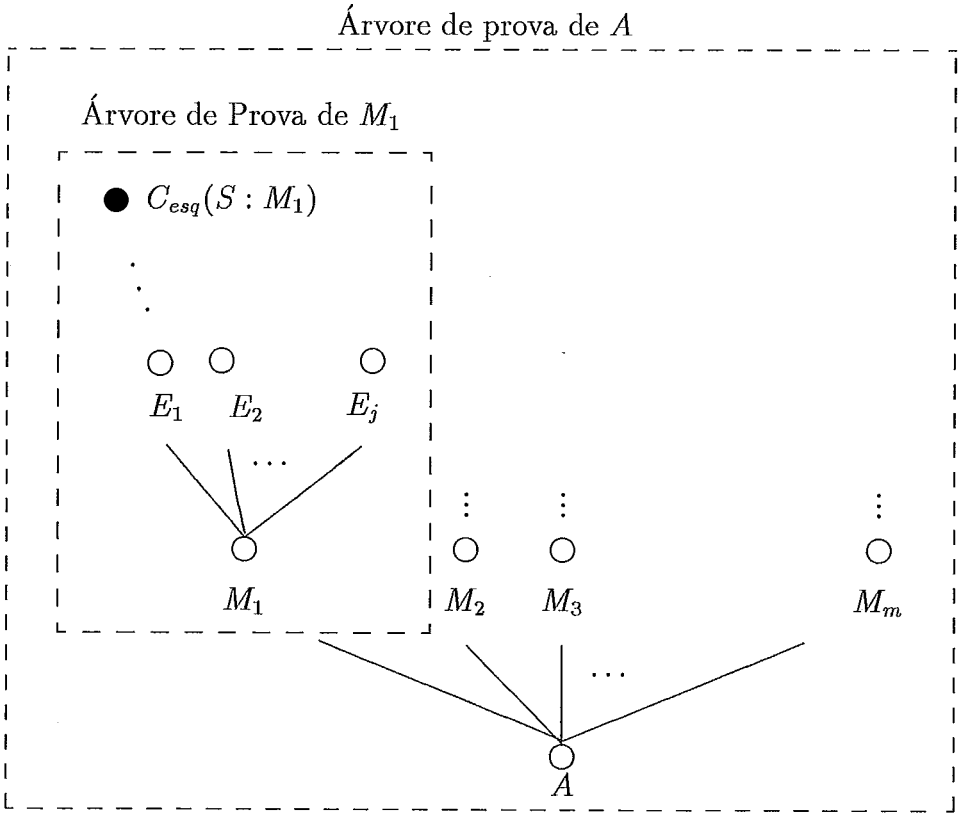
Suponha que a proposição seja verdadeira para sufixos de refutação de tamanho até $k - 1$. Vamos provar que a mesma é verdadeira para sufixos de tamanho $k$.

Parte de uma LSD($f$)-refutação de sufixo de tamanho $k$ pode ser vista no esquema abaixo:

$$\begin{array}{rcl} C_r & \leftarrow & A \\ C_{r+1} & \leftarrow & M_1, M_2, \ldots, M_m \\ & \vdots & \\ C_{r+K} & & \square \end{array}$$

onde $C_{r+1}$ é a cláusula obtida pela $f$-extensão de $C_r$ com $C_j$ para algum $j < r$. Agora consideremos a LSD($f$)-refutação de $S \bigcup \{C_{r+1}\}$. O sufixo dessa refutação possui tamanho $k - 1$. Pelo Teorema da Correção, $M_1$, $M_2$, ..., $M_m$ é uma conseqüência lógica de $S$ e portanto também $M_1$. Claramente, o sufixo da refutação de $S \bigcup \{\leftarrow M_1\}$ terá um tamanho menor ou igual a $k - 1$. Pela Hipótese de Indução, $M_1$ não é uma conseqüência lógica de $S_{M_1}^-$.

Agora observe a seguinte árvore de LSD-prova para $A$ a partir de $S$:




Pela definição de $C_{esq}$, $C_{esq}(S : A) = C_{esq}(S : M_1)$. Assim, $S^-_{M_1} = S^-_A$ e então $M_1$ não é uma conseqüência lógica de $S^-_A$. Logo, $A$ não é uma conseqüência lógica de $S^-_A$.

□

Como cada árvore de prova é mapeada num ramo de sucesso da árvore de LSD($f$)-refutação, essa árvore de refutação para o conjunto contraído não terá nenhum ramo de sucesso e conseqüentemente não haverá uma LSD($f$)-refutação de $S^-_A \cup \{\leftarrow A\}$.

Esse resultado é igualmente válido para $C_{max}$ e $C_{EE}$. Em geral, qualquer função que retire pelo menos uma das folhas de cada uma das árvores de prova do literal respeita ($K^-4$), pois assim o fazendo estará cortando cada ramo de sucesso na árvore de LSD($f$)-refutação correspondente.

### 7.2.3 Expansões em Conjuntos Definidos

Assim como no caso das Contrações, as Expansões às vezes podem ser executadas de mais de uma forma.

DEFINIÇÃO 7.1 :

*Uma* Expansão Direta *de um literal $l$ num Conjunto Definido $S$, corresponde a um novo Conjunto Definido $S_l^+$ tal que:*

$$S_l^+ = \begin{cases} S & \text{se } l \text{ é uma conseqüência lógica de } S \\ S \cup \{l \leftarrow\} & \text{caso contrário} \end{cases}$$

*Uma* Expansão Indireta *de um literal $l$ num Conjunto Definido $S$ é possível se $l$ não for uma conseqüência lógica de $S$ e existir uma cláusula $C$ em $S$ da forma*

$$L \leftarrow A_1, A_2, \ldots, A_n$$

*tal que $L$ unifica com $l$, com umg $\theta$ e os literais $(A_1, A_2, \ldots, A_n)\theta$ são expandidos direta ou indiretamente em $S$.*

É fácil constatar que a forma pela qual foram definidas as Expansões acima, verifica o postulado $(K^+4)$.

Note que uma Expansão Indireta sempre recai na Expansão Direta de alguma propriedade. Além disso, o modo Indireto utiliza-se da inclusão de proposições menos epistemologicamente entrincheiradas a fim de se obter a Expansão, conforme pode ser observado por $(EE2)$ e $(EE3)$:

$(EE2)$ Para quaisquer $A$ e $B$, se $A \vdash B$ então $A \leq B$.

$(EE3)$ Para quaisquer $A$ e $B$ em $K$, $A \leq A \wedge B$ ou $B \leq A \wedge B$.

Uma Expansão Indireta pode exigir diversos passos, uma vez que cada literal do antecedente pode ser o conseqüente de outras regras e assim sucessivamente. Na Expansão Indireta sempre é acrescentado pelo menos um literal a mais do que na Direta: o literal a expandir propriamente dito e um ou mais literais que dêem suporte à regra. É portanto uma Expansão maior. Uma conseqüência imediata disso é que esse tipo de Expansão não verifica o sexto postulado das Expansões $(K^+6)$.

**Exemplo 7.3** *Expansão do literal voa(foguete)*

*A realização das Expansões será ilustrada utilizando-se o Conjunto Definido descrito a seguir e que será referido por Cenário Original:*

***Cenário Original***

1. $tem\_propulsao(foguete)$
2. $voa(X) \leftarrow aerodinamico(X), tem\_propulsao(X).$
3. $aerodinamico(X) \leftarrow tem\_asas(X).$

Uma Expansão Direta de $voa(foguete)$ no Cenário acima implicaria na adição do próprio fato $voa(foguete)$ ao Cenário (figura 7.3).

**Cenário Expandido Diretamente**

1. $voa(foguete)$
2. $tem\_propulsao(foguete)$
3. $voa(X) \leftarrow aerodinamico(X), tem\_propulsao(X).$
4. $aerodinamico(X) \leftarrow tem\_asas(X).$

Figura 7.3: Expansão Direta de $voa(foguete)$

Na Expansão Indireta há duas possibilidades:

i) acrescentar o literal $tem\_asas(foguete)$ (figura 7.4); ou

ii) acrescentar o literal $aerodinamico(foguete)$ (figura 7.5).

A propriedade $aerodinamico(X)$ foi definida como uma característica dos objetos que têm asas. Evidentemente, nem tudo que é aerodinâmico tem asas. Similarmente $\forall x[aerodinamico(x) \wedge tem\_propulsao(x) \rightarrow voa(x)]$, mas pode haver a necessidade da Expansão de algumas instâncias de $voa(X)$ que não possuam as propriedades de serem aerodinâmicas e terem propulsão. Uma Expansão Direta em relação à instância de $voa(X)$ nesses casos é mais recomendável. A disponibilidade de uma ordenação poderia auxiliar na escolha do tipo de Expansão mais adequada.

Uma Expansão que é Indireta em relação a um predicado pode ser Direta em relação a outro. Por exemplo, a inclusão de $tem\_asas(aviao)$ é uma Expansão Direta com relação a $tem\_asas(aviao)$, mas Indireta em relação a $aerodinamico(aviao)$.

**Cenário Expandido Indiretamente I**

1. $tem\_asas(foguete)$
2. $tem\_propulsao(foguete)$
3. $voa(X) \leftarrow aerodinamico(X), tem\_propulsao(X).$
4. $aerodinamico(X) \leftarrow tem\_asas(X).$

Figura 7.4: Expansão Indireta de $voa(foguete)$ via $tem\_asas(foguete)$

**Cenário Expandido Indiretamente II**

1. $aerodinamico(foguete)$
2. $tem\_propulsao(foguete)$
3. $voa(X) \leftarrow aerodinamico(X), tem\_propulsao(X).$
4. $aerodinamico(X) \leftarrow tem\_asas(X).$

Figura 7.5: Expansão Indireta de $voa(foguete)$ via $aerodinamico(foguete)$

No exemplo acima, é claro que uma Expansão Indireta do literal $voa(foguete)$ através do literal $aerodinamico(foguete)$ seria preferível a uma Expansão Indireta através do literal $tem\_asas(foguete)$, pois os foguetes não têm asas. Por $(EE3)$ e pela definição das propriedades no Cenário Original, temos que:

$$tem\_asas(foguete) \leq aerodinamico(foguete) \leq voa(foguete)$$

Isso pode sugerir que, sob o ponto de vista racional, quanto mais epistemologicamente entrincheirada uma proposição dentre as que efetuariam a Expansão, mais segura seria a Expansão obtida. Segue que uma Expansão Direta acrescenta no Cenário a proposição mais Epistemologicamente Entrincheirada dentre as que realizariam a Expansão desejada.

Conforme definida, a Expansão Direta verifica os seis postulados das Expansões propostos por [Gär88] (veja seção 7.4).

## 7.3 Revisões em Conjunto pseudo-Definidos

### 7.3.1 Considerações Iniciais

Revisão é uma mudança não-monotônica no estado de crença, pois ao se aceitar uma nova crença, aquelas que a contradiziam devem ser inicialmente descartadas a fim de manter-se a consistência do Conjunto de Crenças.

Através da Identidade de Levi, que descreve o processo de Revisão em função da Contração e da Expansão, obtemos um método pelo qual pode-se efetuar uma Revisão de um Conjunto de Crenças com relação a uma proposição.

$(*)\quad K_A^* = (K_{\neg A}^-)_A^+$

Em Conjunto Definidos, nenhum tipo de negação é suportada como conseqüência, pois não são permitidos literais negativos no antecedente das cláusulas. Ou seja, nenhum tipo de informação "negativa" pode ser deduzida, tornando sem sentido uma análise dos processos de Revisão.

Os Conjuntos pseudo-Definidos permitem a dedução de informações negativas, através da regra da Negação por Falha Finita (NFF), sob a Hipótese de Mundo Fechado. A negação contudo, não corresponde à negação clássica e sim uma variante sua mais fraca.

Além disso, a NFF só se aplica a literais básicos negativos. E a dedução só é possível se a árvore de refutação do literal complementar for finita e não tiver nenhum ramo de sucesso. A seguir são feitas algumas considerações sobre os Processos de Revisão nesse tipo de conjunto.

### 7.3.2 Revisão de Literais Positivos

A seguir será apresentada uma proposição que se mostrará bastante útil na análise dos processos de Revisão em Conjuntos quase-Definidos:

**Proposição 7.2** *Seja $L$ um literal positivo e $S$ um Conjunto pseudo-Definido. A Revisão de $S$ por $L$, definida através da Identidade de Levi, é equivalente a uma Expansão de $L$ em $S$. Ou seja, $S_L^* = S_L^+$.*

**Prova:**

A Identidade de Levi define uma Revisão da seguinte forma:

$$S^*_L = (S^-_{\neg L})^+_L$$

Em função disso, há dois casos a se considerar:

1º caso: $\neg L$ é uma conseqüência lógica de $S$.

Como $\neg L$ só pode ser obtido por NFF, então uma prova para $L$ não foi encontrada. A Contração de $\neg L$ exige uma Expansão de $L$ e nesse caso, $S^-_{\neg L} = S^+_L$ e portanto $S^*_L = (S^+_L)^+_L = S^+_L$, para qualquer função de Expansão + que respeite $(K^+4)$.

2º caso: $\neg L$ não é uma conseqüência lógica de $S$.

Nesse caso, $S^-_{\neg L} = S$, para qualquer função de Contração − que respeite $(K^-3)$, e portanto $S^*_L = (S)^+_L = S^+_L$

□

### 7.3.3 Revisão de Literais Negativos

As considerações acerca da Revisão de Literais Negativos requer um cuidado maior, porque a regra da NFF nem sempre pode ser aplicada a fim de possibiltar uma dedução desses literais.

Seja $\neg L$ um literal negativo e $S$ um Conjunto pseudo-Definido. A Revisão de $S$ por $\neg L$, através da identidade de Levi, é definida da seguinte forma:

$$S^*_{\neg L} = (S^-_L)^+_{\neg L}$$

1º caso: $L$ é uma conseqüência lógica de $S$.

Se $L$ foi obtido através da aplicação de uma regra e não de um fato, ao se realizar sua Contração, a revisão de $\neg L$ estará sendo automaticamente efetuada, pois a NFF forneceria uma dedução para $\neg L$, sendo a Expansão desnecessária. Contudo, se $L$ foi obtido através de um fato e não existir nenhuma outra cláusula em $S$ em que $L$ apareça, então $fecho(S^-_L)$ não conterá a definição completa de $L$ e não será possível deduzir $\neg L$ com a NFF.

2º caso: $L$ não é uma conseqüência lógica de $S$.

Nesse caso, $S^-_L = S$, para qualquer função de Contração − que respeite $(K^-3)$, e portanto $S^*_L = (S)^+_{\neg L}$. Mais uma vez a Expansão é desnecessária, mesmo porque

não é possível uma Expansão propriamente dita de literais negativos. Se alguma cláusula em $S$ possuir o literal $L$, então sua definição completa estará em $fecho(S)$ e portanto $\neg L$ será uma conseqüência lógica de $fecho(S)$. Ainda assim, a derivação de $\neg L$ só será possível se ele for básico, pois a Resolução LSDNF não é Completa (veja Seção 4.5.1).

## 7.4 Postulados de Mudanças Epistemológicas p/Programação em Lógica

A seguir serão feitas algumas considerações sobre a utilização dos postulados de [Gär88] para regularem as Mudanças Epistemológicas em Conjuntos Definidos e pseudo-Definidos. As observações relacionadas aos literais negativos consideram que a negação é obtida através da regra da NFF sob a Hipótese de Mundo Fechado.

### Considerações relacionadas aos postulados de Contração

($K^-1$) $K_A^-$ é um Conjunto de Crenças

Se considerarmos que, no nosso caso, um Conjunto de Crenças é um Conjunto de Cláusulas Definidas, então pode-se tomar esse postulado como válido.

($K^-2$) $K_A^- \subseteq K$

Para Conjuntos Definidos é sempre válido. Considerando-se a verificação do Princípio da Economia Informativa, no caso extremo (quando $A$ não é uma conseqüência lógica de $K$) $K_A^- = K$, e então $K_A^- \subseteq K$.

Para Conjuntos pseudo-Definidos, nem sempre é verificado. Suponha o caso em que $K = \{p(a), p(b)\}$. $K_{p(a)}^- = \{p(b)\}$. Pela regra da NFF, sob a Hipótese de mundo fechado, $\neg p(a) \in K_{p(a)}^-$. Contudo, $\neg p(a) \notin K$ e logo $K_{p(a)}^- \not\subseteq K$.

($K^-3$) Se $A \notin K$, então $K_A^- = K$

Esse postulado foi tomado com parâmetro na definição das funções de Contração apresentadas aqui, é portanto válido para as mesmas.

($K^-4$) Se $\not\vdash A$ então $A \notin K_A^-$

Veja Proposição 7.1.

($K^-5$) Se $A \in K$ então $K \subseteq (K_A^-)_A^+$

A motivação para a estipulação desse postulado foi determinar um princípio de recuperação entre as Contrações e Expansões. Para esse contexto específico, o mesmo não parece razoável, uma vez que as Contrações tendem a retirar fatos que suportam a conseqüência das proposições e as Expansões tendem a expandir os próprios literais relacionados às proposições (Expansão Direta). Seria possível efetuar sua verificação num contexto onde apenas fatos fossem utilizados (possivelmente num Banco de Dados Lógicos – veja Seção 3.8). Com a utilização de regras, é difícil se garantir que todos os literais retirados para atingir a Contração sejam recuperados numa Expansão feita a seguir. A proposição em si é recuperada, mas todos os fatos que a suportavam não necessariamente são.

EXEMPLO 7.1 :

*Sejam* $S = \{p(a), r(a), q(X) \leftarrow p(X), r(X)\}$ *um Conjunto Definido, – uma função de Contração definida segundo o critério* $C_{esq}$ *e* $+$ *uma função de Expansão Direta.*

$$S_{q(a)}^- = \{r(a), q(X) \leftarrow p(X), r(X)\}$$

$$(S_{q(a)}^-)_{q(a)}^+ = \{q(a), r(a), q(X) \leftarrow p(X), r(X)\}$$

*Na seqüência acima podemos observar que* $p(a) \in S$, *mas* $p(a) \notin (S_{q(a)}^-)_{q(a)}^+$.

($K^-6$) Se $\vdash A \leftrightarrow B$, então $K_A^- = K_B^-$

($K^-7$) $K_A^- \cap K_B^- \subseteq K_{A \wedge B}^-$

($K^-8$) Se $A \notin K_{A \wedge B}^-$ então $K_{A \wedge B}^- \subseteq K_A^-$

Por restrição da Linguagem, os três postulados acima não se aplicam.

### Considerações relacionadas aos postulados de Expansão

A seguir seguem algumas considerações sobre a aplicação dos postulados propostos por Gärdenfors para regularem as Expansões em Conjuntos Definidos e Conjuntos pseudo-Definidos.

($K^+1$) $K_A^+$ é um Conjunto de Crenças

As considerações para esse postulado são similares às do postulado equivalente para o caso das Contrações.

($K^+2$) $A \in K_A^+$

Esse postulado é válido independentemente da forma como a expansão seja realizada.

($K^+3$) $K \subseteq K_A^+$

Para Conjuntos Definidos, o postulado é sempre válido.

Para Conjuntos pseudo-Definidos, nem sempre é verificado. Tomemos como exemplo o conjunto $S = \{p(a), p(b)\}$. $S^+_{\neg p(a)} = \{p(b)\}$. $p(a) \in S$, mas $p(a) \notin S^+_{\neg p(a)}$. Quando se está trabalhando com Teorias da Lógica de Primeira Ordem, esse problema não existe pois se $A \in K$, $K^+_{\neg A} = K_\perp \supseteq K$.

($K^+4$) Se $A \in K$, então $K_A^+ = K$

Esse postulado pode ser assegurado. A forma pela qual foram definidas as Expansões na Seção 7.2.3, por exemplo, assegura sua verificação.

($K^+5$) Se $K \subseteq H$, então $K_A^+ \subseteq H_A^+$

Pela Expansão Direta é sempre válido.

Por Expansão Indireta, da forma como as mesmas foram definidas, nem sempre pode ser assegurado. Suponha o caso em que

$$K = \{q(X) \leftarrow p(X)\},$$

$$H = \{q(X) \leftarrow p(X),\ q(X) \leftarrow r(X)\}$$

e duas funções de Expansão Indireta, $+'$ e $+"$ tais que

$$K^{+'}_{q(a)} = \{p(a),\ q(X) \leftarrow p(X)\} \text{ e}$$

$$H^{+"}_{q(a)} = \{r(a),\ q(X) \leftarrow p(X),\ q(X) \leftarrow r(X)\}.$$

Claramente, $K \subseteq H$ e $H^{+"}_{q(a)}$ é uma Expansão Indireta válida de $q(a)$. Contudo, $p(a) \in H^{+'}_{q(a)}$, mas $p(a) \notin H^{+"}_{q(a)}$. Esse exemplo serve para ilustrar a possibilidade de haver também funções de ordenação para guiar as Expansões Indiretas (uma espécie de $C_{EE}$ para Expansões). Tal ordenação pode ser diferente nos dois conjuntos (mesmo porque podem existir elementos em $H$ que não estejam em $K$). Sendo assim, é possível que numa ordenação dessas $r(a) \geq p(a)$ e a escolha fosse pelo literal mais Epistemologicamente Entrincheirado, quando então o postulado não se verificaria.

($K^+6$) Para todos os Conjuntos de Crenças $K$ e sentenças $A$, $K_A^+$ é o menor conjunto que satisfaz ($K^+1$) a ($K^+5$)

Apenas as Expansões Diretas verificam esse postulado.

### Considerações relacionadas aos postulados de Revisão

Conforme efetuado para as Contrações e Expansões seguem algumas considerações sobre a aplicação dos postulados propostos por Gärdenfors para regularem as Revisões em conjuntos de Cláusulas de Horn.

($K$*1) $K_A^*$ é um Conjunto de Crenças

Similarmente aos postulados equivalentes das Expansões e Contrações.

($K$*2) $A \in K_A^*$

Veja Seção 7.3.

($K$*3) $K_A^* \subseteq K_A^+$

Veja Seção 7.3.

($K$*4) Se $\neg A \notin K$, então $K_A^+ \subseteq K_A^*$

Caso i) $A$ é um literal positivo. Pela Proposição 7.2, $K_A^* = K_A^+$.

Caso ii) $A$ é um literal negativo. Seja $L$ um literal positivo tal que $A = \neg L$. Se $\neg A \notin K$, então $\neg\neg L \notin K$. Se a prova de $\neg\neg L$ falha então é porque a prova de $\neg L$ é bem-sucedida e portanto a prova de $L$ falha finitamente.

$$\begin{aligned} K_A^+ &= K_{\neg L}^+ = K \text{ pois a prova de } \neg L \text{ é bem-sucedida} \\ K_A^* &= (K_{\neg A}^-)_A^+ = (K_{\neg\neg L}^-)_{\neg L}^+ \\ &= K_{\neg L}^+, \text{ pois a prova de } \neg\neg L \text{ falha, } K_{\neg\neg L}^- = K \\ &= K, \text{ pois a prova de } \neg L \text{ é bem-sucedida} \end{aligned}$$

Logo, se $\neg A \notin K$, então $K_A^+ \subseteq K_A^*$.

($K$*5) $K_A^* = K_\perp$ sse $\vdash \neg A$

Não se aplica, pois não podemos ter um Conjunto de Cláusulas Definidas Inconsistente.

($K$*6) Se $\vdash A \leftrightarrow B$, então $K_A^* = K_B^*$

($K$*7) $K_{A \wedge B}^* \subseteq (K_A^*)_B^+$

($K$*8) Se $\neg B \notin K_A^*$, então $(K_A^*)_B^+ \subseteq K_{A \wedge B}^*$

Também não se aplicam.

## 7.5 Conclusões

Esse Capítulo procurou analisar o comportamento das Mudanças Epistemológicas em Conjuntos de Cláusulas Definidas e Cláusulas pseudo-definidas. Foram apresentadas alternativas para a realização dos processos de Contração e Expansão e consideradas as conseqüência nos conjuntos resultantes. Através da Identidade de Levi, é possível também construir funções de Revisão a partir das Contrações e Expansões definidas.

A Negação por Falha Finita é uma das poucas formas disponíveis de se obter a negação em Conjuntos pseudo-Definidos. As implicações de seu uso com essa finalidade, face aos processos de Revisão, foram também consideradas.

Por fim, os principais postulados propostos por [Gär88] para nortearem as Mudanças Epistemológicas foram analisados tendo em vista sua utilização na Programação em Lógica.

# Capítulo 8

# PROLOG MODAL DE AÇÃO COM REVISÃO DE CRENÇAS

## 8.1 Introdução

O Prolog Modal de Ação com Revisão de Crenças é um formalismo voltado para o problema da geração de planos que utiliza os conceitos de Revisão de Crenças a fim de realizar as mudanças de estado determinadas pela execução das ações.

Cada cenário (um estado do sistema) pode ser visto como um conjunto de cláusulas definidas. As ações são aplicadas a esses conjuntos de forma a transformá-los num novo conjunto cujas propriedades reflitam as características do novo cenário.

As mudanças no conjunto determinadas pelas ações consistem basicamente na Contração de propriedades que se espera não mais valerem no estado imediatamente posterior à execução da ação e na Expansão de propriedades que passam então a valer nesse estado.

A utilização dos métodos propostos para obtenção de Expansões e Contrações em conjuntos de cláusulas definidas é discutida tendo em vista o atual contexto do problema da geração de planos.

## 8.2 Linguagem de Primeira Ordem

A linguagem de primeira ordem para esse mecanismo é a mesma apresentada para a versão sem Revisão de Crenças, exceto por algumas modificações necessárias para acomodar os operadores.

DEFINIÇÃO 8.1 : *Alfabeto*

- *Um conjunto de símbolos predicativos n-ários;*
- *Um conjunto infinito de variáveis;*
- *Um conjunto de símbolos funcionais n-ários;*
- *Um conjunto de constantes;*
- *Um conjunto finito de nomes de ações, denominado Ac;*
- *Símbolos lógicos:* $\leftarrow$, [, ], (,),+,−, / *e* ,.

DEFINIÇÃO 8.2 : *Símbolos da linguagem*

- *termo: é uma variável, uma constante do alfabeto, ou uma expressão da forma* $f(t_1, t_2, ..., t_n)$, *onde* $f$ *é um símbolo funcional n-ário e* $t_1$,$t_2$,...,$t_n$ *são termos do alfabeto.*
- *literal positivo: é uma expressão da forma* $p(t_1, t_2, ..., t_n)$ *onde* $p$ *é um símbolo predicativo n-ário e* $t_1, t_2, ..., t_n$ *são termos*
- *ação: Se* $\alpha^n$ *é um nome de ação n-ário e* $t_1, t_2, ..., t_n$ *são termos então* $[\alpha(t_1, t_2, ..., t_n)]$ *é uma ação.*
- $[\![\alpha]\!]^n$*:* $[\alpha_1][\alpha_2]...[\alpha_n]$, *onde os* $\alpha_i$*'s são ações executadas seqüencialmente*
- *átomo puro: é uma expressão da forma* $p(t_1, t_2, ..., t_n)$ *onde* $p$ *é um símbolo predicativo n-ário e* $t_1, t_2, ..., t_n$ *são termos*
- *fatos: átomos puros da linguagem*

DEFINIÇÃO 8.3 : *Cláusulas de Horn não-Modais e Cláusulas de Horn Modais*

***Claúsulas de Horn não-Modais***

*São regras do tipo* $q \leftarrow a_1, a_2, ..., a_m$, *onde cada* $a_i$ $(1 \leq i \leq m)$ *é um literal positivo e* $q$ *é um átomo puro.*

***Cláusulas de Horn Modais***

*São regras do tipo* $[\![\alpha]\!]^n(d_1, d_2, ..., dk)^-/(s_1, s_2, ..., s_l)^+ \leftarrow a_1, a_2, ..., a_m$, *onde:*

*i)* $[\![\alpha]\!]^n$ *é uma seqüência não vazia de ações*

*ii) cada* $a_i$ $(1 \leq i \leq m)$, $d_i$ $(1 \leq i \leq k)$ *e* $s_i$ $(1 \leq i \leq l)$ *é um literal positivo.*

A seqüência $[\![\alpha]\!]^n$ $(n > 0)$ é chamada de *parte modal do conseqüente* ou *prefixo* e, $d_1, d_2, ..., dk$ e $s_1, s_2, ..., s_l$ constituem, respectivamente, as partes positiva e negativa do conseqüente.

O item i) é requerido para assegurar que a mudança de propriedades resulta num novo cenário, no caso o cenário atingido através da execução das ações de $[\![\alpha]\!]^n$ no cenário onde a regra foi aplicada; o item ii) porque os Cenários serão representados por Conjuntos Definidos.

Observe que o antecedente de uma regra modal pode ser visto como uma conjunção de literais, o que não ocorre com a parte negativa do conseqüente da mesma, pois se assim fosse, bastaria realizar a Contração de um dos literais e a Contração da conjunção seria obtida (por $(EE3)$ e pelo **Princípio da Conservatividade**).

Um Cenário é uma representação de um estado do sistema através da descrição das propriedades nele verificadas. Essa representação é efetuada por um Conjunto de Cláusulas Definidas. Segue-se das definições dadas acima que um Cenário é descrito por um conjunto de fatos e um conjunto de regras não-modais.

DEFINIÇÃO 8.4 : *Cenário*

*Um* Cenário *representando informação acerca de um estado do sistema é representado por um Conjunto Definido.*

DEFINIÇÃO 8.5 : *Aplicabilidade de Regras*

*Uma regra modal R do tipo*

$$[\![\alpha]\!]^n(d_1, d_2, ..., d_k)^- / (s_1, s_2, ..., s_l)^+ \leftarrow a_1, a_2, ..., a_m$$

*é dita aplicável a um cenário C sse houver uma LSD(f)-refutação a partir do conjunto* $C \cup \{\leftarrow a_1, a_2, \ldots, a_m\}$.

## 8.3 Função de Contração

Sejam **C** o Conjunto de todos os Cenários e **L** o conjunto de todos os literais. A dinâmica das Contrações será determinada pela função $\delta : \mathbf{C} \times \mathbf{L} \longrightarrow 2^{\mathbf{L}}$, de pares de Cenários e Literais ($\mathbf{C} \times \mathbf{L}$) em conjuntos de Literais ($2^{\mathbf{L}}$). A função $\delta$ é do tipo centralizada nas instâncias e será definida aqui em termos da retirada mínima de fatos guiada pelo literal mais à esquerda das árvores de prova ($C_{esq}$). Não há portanto retirada de regras.

A imagem de $\delta$ é o conjunto das partes de **L** porque é possível haver mais de uma árvore de prova para o literal $l$, em cujo caso $\delta$ leva no conjunto de fatos a serem removidos. Assim se $\delta(C, l) = D$, então $D$ é um subconjunto de fatos do cenário $C$ contendo os fatos que devem ser removidos de $C$ a fim de se obter uma contração de $l$.

EXEMPLO 8.1 : *Contração de livre(a)*

*Seja o seguinte conjunto de cláusulas definidas, que define o cenário C:*

***Cenário** C*

1. $mesa(a)$
2. $mesa(b)$
3. $limpo(a)$
4. $limpo(b)$
5. $livre(X) \leftarrow mesa(X), limpo(X)$

*E a seguinte regra modal R:*

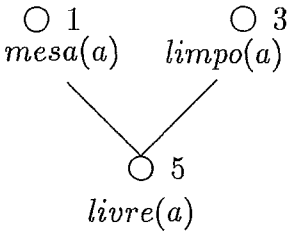
$[empilha(X,Y)](livre(X), limpo(Y))^{-}/(sobre(X,Y))^{+} \leftarrow livre(X),\ limpo(Y).$

*A regra R acima é aplicável ao cenário C. Uma das substituições de variáveis em R que a tornam aplicáveis a C é $\{X/a, Y/b\}$. Para essa instância de R os literais a contrair são $limpo(b)$ e $livre(a)$, cujas árvores de prova são as seguintes:*

*Prova de limpo(b)*

○ 4
*limpo(b)*

*Prova de livre(a)*




*livre(a)*

*Assim temos que,* $\delta(C, limpo(b)) = \{limpo(b)\}$ *e* $\delta(C, livre(a)) = \{mesa(a)\}$. *Note que se a retirada máxima de fatos tivesse sido adotada, teríamos*

$$\delta(C, livre(a)) = \{mesa(a), limpo(a)\}.$$

Ao se definir o mecanismo de prova uma alternativa útil é abstrair o mesmo dos detalhes sobre como as Contrações e Expansões são efetuadas. Isso é feito estendendo-se a função $\delta$ definida acima para uma função $-$ de conjuntos de Cenários e Literais ($\mathbf{C} \times \mathbf{L}$) em conjuntos de Cenários ($\mathbf{C}$) da seguinte forma:

$$C_l^- = C - \delta(C, l)$$

onde $C_l^-$ é a Contração do cenário $C$ em relação ao literal $l$.

No exemplo anterior temos que $\delta(C, livre(a)) = \{mesa(a)\}$. Logo,

$C^-_{livre(a)}$

1. $mesa(b)$
2. $limpo(a)$
3. $limpo(b)$
4. $livre(X) \leftarrow mesa(X), limpo(X)$

Note que a propriedade $livre(a)$ não é mais uma conseqüência do cenário resultante, garantindo a verificação dos postulados ($K^-1$), ($K^-2$), ($K^-3$) e ($K^-4$).

## 8.4 Função de Expansão

As Expansões são determinadas pela função $\theta : \mathbf{C} \times \mathbf{L} \longrightarrow 2^{\mathbf{L}}$, de pares de Cenários e Literais ($\mathbf{C} \times \mathbf{L}$) em conjuntos de Literais ($2^{\mathbf{L}}$). O tipo de Expansão adotado aqui é o Direto. A motivação para isso é que esse é o menor tipo de Expansão, adiciona exatamente a propriedade determinada, além de ser o mais simples de implementar. A Expansão Direta, ao contrário da Indireta, garante a verificação de todos os postulados para a Expansão, incluindo ($K^+6$).

A definição de $\theta$ é a seguinte:

$$\theta(C, l) = \begin{cases} \emptyset & \text{se há uma prova para } l \text{ em } C \\ \{l\} & \text{caso contrário} \end{cases}$$

Assim como para as Contrações, a função $\theta$ é estendida para função + de conjuntos de Cenários e Literais ($\mathbf{C} \times \mathbf{L}$) em conjuntos de Cenários ($\mathbf{C}$):

$$C_l^+ = C \cup \theta(C, l) = \begin{cases} C & \text{se há uma prova para } l \text{ em } C \\ C \cup \{l\} & \text{caso contrário} \end{cases}$$

Utilizando a mesma instância de $R$ do exemplo anterior:

$C^+_{sobre(a,b)}$

1. $mesa(a)$
2. $mesa(b)$
3. $limpo(a)$
4. $limpo(b)$
5. $sobre(a, b)$
6. $livre(X) \leftarrow mesa(X), limpo(X)$

A forma como foi definida a função + garante a verificação de todos os postulados para Expansão (($K^+1$) a ($K^+6$)). Uma Expansão Indireta não verificaria o sexto postulado (($K^+6$)).

## 8.5 Aplicação das Regras Modais

É necessário ainda definir em que ordem as Expansões e Contrações serão realizadas. O cenário $\mathcal{C}'$ obtido a partir da aplicação de uma regra modal $R$ aplicável a um cenário $\mathcal{C}$ é obtido da seguinte forma:

i) para cada literal $d_i$ presente na parte negativa da regra R, retire o fato $\delta(\mathcal{C}, d_i)$, definido pela função de Contração $\delta$, do cenário $\mathcal{C}$, obtendo por fim o cenário $\mathcal{C}''$.

ii) para cada literal $a_i$ presente na parte positiva da regra R, acrescente o fato $\theta(\mathcal{C}'', a_i)$, definido pela função de Expansão $\theta$, ao cenário $\mathcal{C}''$, obtendo por fim o cenário $\mathcal{C}'$.

Uma especificação mal efetuada pode ocasionar alguns efeitos indesejados. Em particular, a Expansão pode ser anulada pela Contração ou vice-versa. O exemplo da Figura 8.1 ilustra melhor uma situação onde isso ocorre.

$C_0$

1. $a_1$
2. $a_2$
3. $a_1 \leftarrow a_3$
4. $p_1 \leftarrow a_1,\ a_2$
5. $[\alpha](p_1)^- / (a_3)^+ \leftarrow p_1$

Figura 8.1: Exemplo de um problema mal especificado

Se a Contração de $p_1$ for realizada e em seguida a Expansão de $a_3$ teremos a evolução de cenários mostrada na Figura 8.2.

$(C_0)^-_{p_1}$

1. $a_2$
2. $a_1 \leftarrow a_3$
3. $p_1 \leftarrow a_1, a_2$

$((C_0)^-_{p_1})^+_{a_3}$

1. $a_2$
2. $a_3$
3. $a_1 \leftarrow a_3$
4. $p_1 \leftarrow a_1, a_2$

Figura 8.2: Realizando a Contração e depois a Expansão

Para obter a Contração de $p_1$, um dos fatos que suportavam sua conseqüência, $a_1$, foi retirado conforme indicado pela função de Contração $-$. Quando a Expansão de $a_3$ é realizada logo a seguir, uma Expansão Indireta de $a_1$ é obtida e consequentemente também de $p_1$. Nesse caso, a Expansão de $a_3$ anulou o efeito da Contração de $p_1$ realizada anteriormente.

$(C_0)^+_{a_3}$

1. $a_1$
2. $a_2$
3. $a_3$
4. $a_1 \leftarrow a_3$
5. $p_1 \leftarrow a_1, a_2$.

$((C_0)^+_{a_3})^-_{p_1}$

1. $a_2$
2. $a_1 \leftarrow a_3$
3. $p_1 \leftarrow a_1, a_2$.

Figura 8.3: Realizando a Expansão e depois a Contração

Por outro lado, se a Expansão de $a_3$ fosse realizada e em seguida a Contração de $p_1$, teríamos a seqüência correspondente à Figura 8.3. O fato $a_3$ é adicionado de forma Direta, conforme determinado pela função de Expansão $+$. No entanto, a Contração de $p_1$ exige a retirada de todos os literais situados na extremidade esquerda das suas àrvores de prova, que são $a_1$ e $a_3$. Nesse caso, a Contração de $p_1$ anulou o efeito da Expansão de $a_3$ realizada anteriormente.

## 8.6 Algoritmo de prova

O procedimento Prova da Figura 8.5 funciona de forma progressiva. O mesmo recebe um cenário e um *Goal* a ser satisfeito. Se for possível satisfazer esse *Goal* em algum cenário, o procedimento retorna um plano referente à seqüência de regras aplicadas.

Cada $CENARIO$ recebido por Prova é inicialmente testado a fim de se determinar se o mesmo satisfaz *Gol*. Em caso positivo não há nada a se fazer, o procedimento retorna $SUCESSO = verdadeiro$ e $PLANO = vazio$.

Se $CENARIO$ não satisfaz *Gol*, havendo disponibilidade de uma regra de transformação a ele aplicável, a mesma é aplicada gerando um novo cenário. O procedimento é então chamado de forma recursiva para o novo cenário $NOVO\_CENARIO$, retornando um plano intermediário que na realidade é um sub-plano da solução para o problema original, uma vez que uma parte do mesmo já foi solucionada pelos procedimentos anteriores. O plano é obtido adicionando-se todas as ações associadas às regras escolhidas. A primeira ação do plano é aquela do procedimento pai e assim sucessivamente até a última ação que corresponde à regra escolhida no penúltimo procedimento do ramo de sucesso, pois nenhuma regra é escolhida pelo último deles.

Quando em um cenário não há mais disponibilidade de regras a aplicar o procedimento retorna $SUCESSO = falso$ para o nível anterior, em busca de outras alternativas de solução nesse nível.

O uso de uma heurística a fim de guiar a escolha das regras em $(*)$ pode reduzir significativamente o espaço de busca.

Para o mecanismo se tornar completo, é necessário tornar a estratégia de busca completa. Uma forma de fazer isso é usar uma busca em largura ao invés de uma busca em profundidade. É possível também alocar um novo processo a cada nova alternativa de aplicação de uma regra (em $*$). Nesse caso, cada processo buscaria a solução partindo de um cenário diferente. Quando um desses processos achasse uma solução, o mesmo iria comunicá-la ao processo de nível superior e assim sucessivamente até o processo pai, onde o plano completo seria obtido, podendo então esse processo interromper a busca de outras soluções pelos processos filhos.

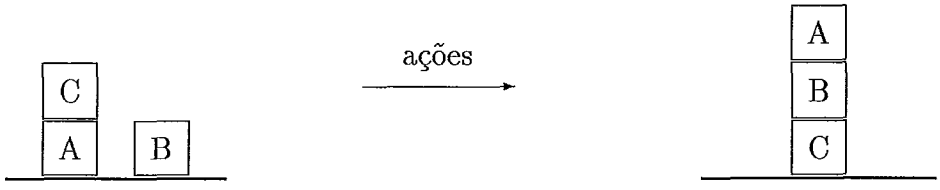



Figura 8.4: Cenário Inicial e cenário objetivo do problema

A seguir temos um exemplo de execução para o problema do mundo dos blocos, no qual a disposição dos blocos no Cenário Inicial e no Cenário Objetivo está como na Figura 8.4.

Prova($CENARIO$,$Gol$)

$PLANO = \emptyset$

Se $CENARIO$ satisfaz $Gol$ então

$SUCESSO \leftarrow verdadeiro$

Senão

$SUCESSO \leftarrow falso$

Enquanto $SUCESSO = falso$, $CENARIO$ não satisfaz $Gol$ e houverem regras aplicáveis a $CENARIO$ faça:

$$* \begin{cases} R \leftarrow \text{ alguma regra aplicável a } CENARIO \\ \text{aplique } R \text{ a } CENARIO \text{ gerando } NOVO\_CENARIO \\ (SUCESSO, PLANO\_INT) \leftarrow \text{Prova}(NOVO\_CENARIO, Gol) \end{cases}$$

Fim_enquanto

Se $SUCESSO = verdadeiro$ então

$PLANO \leftarrow R + PLANO\_INT$

Fim_se

Fim_se

Retorne $SUCESSO$, $PLANO$

Figura 8.5: Algoritmo de prova

A Figura 8.6 ilustra o estado dos procedimentos PROVA no ramo de sucesso; a variável $NOVO\_CENARIO$ foi substituída por $NOVO\_CEN$. A Figura 8.7 contem a evolução da descrição dos cenários. A regra $livre(X) \rightarrow mesa(X), limpo(X)$ foi omitida das descrições. A evolução dos cenários pode ser melhor observada na Figura 8.8.

$$PROVA^0 \left( \left\{ \begin{array}{l} sobre(c,a); mesa(a); \\ limpo(c); mesa(b); limpo(b); \\ livre(X) \leftarrow mesa(X),\ limpo(X) \end{array} \right\}, sobre(a,b) \wedge sobre(b,c) \right)$$

$R_i^0 = desempilha(c,a)$

$NOVO\_CEN_i^0 =$

$$\left\{ \begin{array}{l} limpo(a); mesa(a); limpo(b); \\ limpo(b); limpo(c); mesa(c); \\ livre(X) \leftarrow mesa(X),\ limpo(X) \end{array} \right\}$$

$PROVA^{0.i}(NOVO\_CEN_i^0, sobre(a,b) \wedge sobre(b,c))$

$R_j^{0.i} = empilha(b,c)$

$NOVO\_CEN_j^{0.i} =$

$$\left\{ \begin{array}{l} sobre(b,c), limpo(b); mesa(c); \\ limpo(a); mesa(a); \\ livre(X) \leftarrow mesa(X),\ limpo(X) \end{array} \right\}$$

$PROVA^{0.i.j}(NOVO\_CEN_j^{0.i}, sobre(a,b) \wedge sobre(b,c))$

$R_k^{0.i.j} = empilha(a,b)$

$NOVO\_CEN_k^{0.i.j} =$

$$\left\{ \begin{array}{l} sobre(b,c), sobre(a,b); \\ limpo(a); mesa(c); \\ livre(X) \leftarrow mesa(X),\ limpo(X) \end{array} \right\}$$

$PROVA^{0.i.j.k}(NOVO\_CEN_k^{0.i.j}$, sobre(a,b) $\wedge$ sobre(b,c))

$NOVO\_CEN_k^{0.i.j}$ satisfaz $Gol$

$PLANO^{0.i.j.k} = \emptyset$

$PLANO^{0.i.j} = empilha(a,b) + \emptyset$

$PLANO^{0.i} = empilha(b,c) + empilha(a,b) + \emptyset$

$PLANO^0 = desempilha(c,a) + empilha(b,c) + empilha(a,b) + \emptyset$

Figura 8.6: Estado dos procedimentos no ramo de sucesso

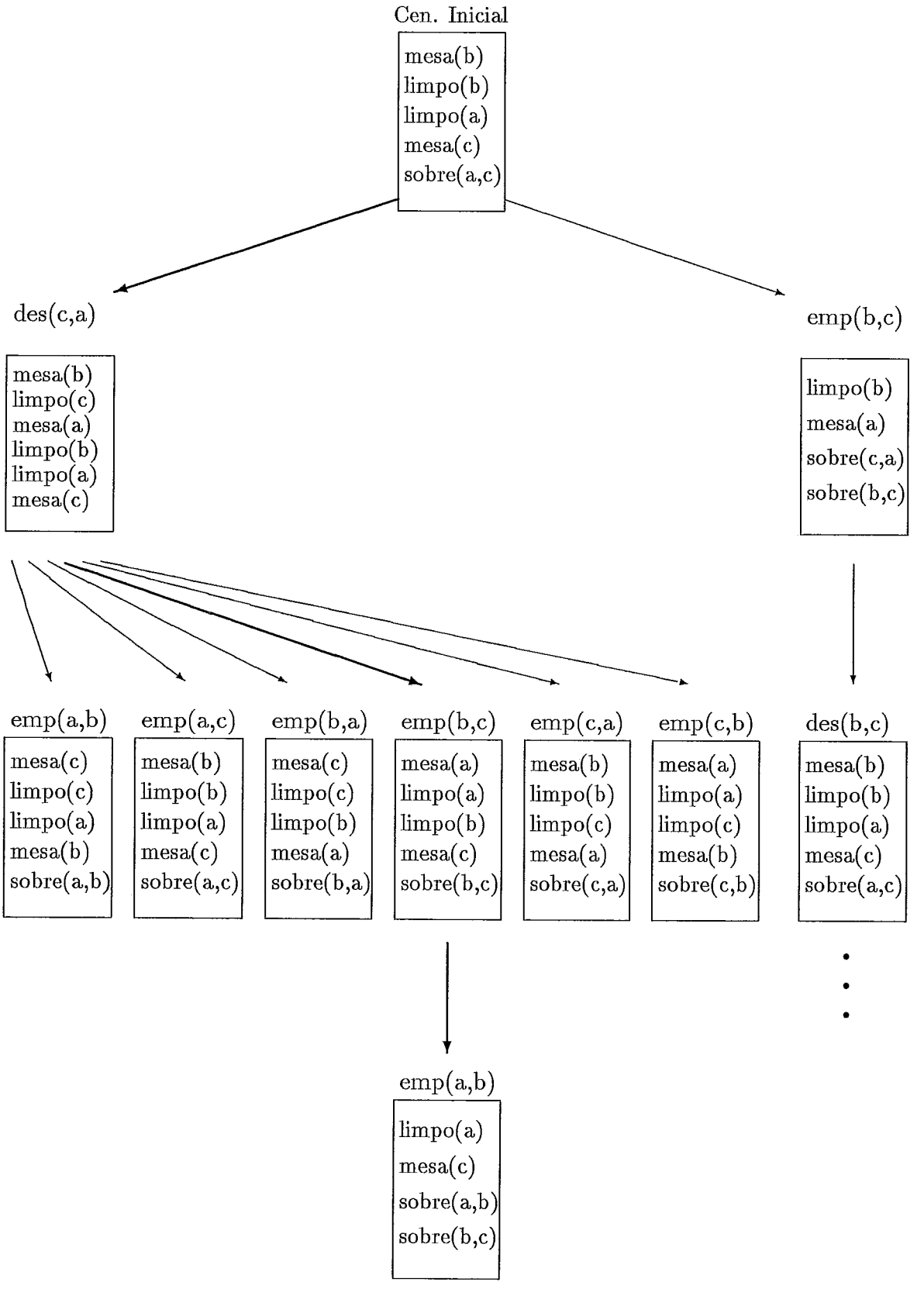



Figura 8.7: Descrição dos cenários

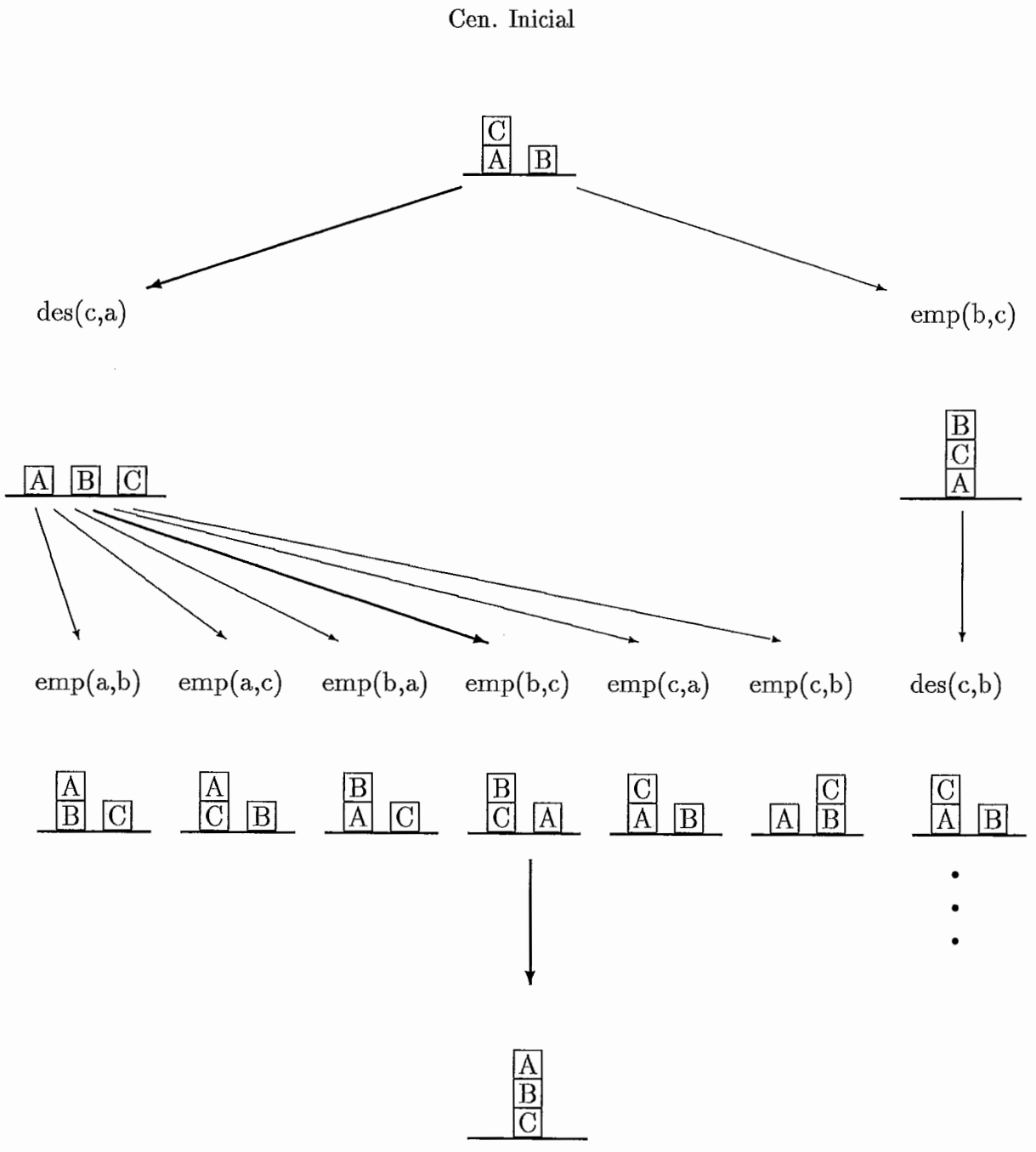



Figura 8.8: Evolução dos cenários

# 8.7 Considerações sobre Completude

O sentido dado ao termo Completude aqui, refere-se à propriedade de que, partindo-se da especificação dada ao problema, se existir um plano que leve o sistema do Cenário Inicial a um Cenário onde as propriedades do *Goal* sejam verificadas, então o algoritmo o encontra. As definições dadas a seguir seguem a terminologia adotada em [War74].

## 8.7.1 Definições Auxiliares

Um plano $P = [\alpha_1][\alpha_2]...[\alpha_n]$ é *aplicável* a um cenário $C$ sse:

i) A regra associada à ação $[\alpha_1]$ é aplicável a $C$; e

ii) Para cada regra $R_i$, associada à ação $[\alpha_i]$ $(2 \leq i \leq n)$, $R_i$ é aplicável ao cenário obtido a partir de $C$ pela execução de $[\![\alpha]\!]^{i-1}$.

Um plano $P = [\alpha_1][\alpha_2]...[\alpha_n]$, *soluciona* um *Goal* $G$ a partir de um cenário inicial $C_0$, sse $P$ é aplicável a $C_0$ e $G$ é verificado no cenário atingido a partir de $C_0$ pela execução de $P$.

Um plano $P = [\alpha_1][\alpha_2]...[\alpha_n]$ é *minimal* para um *Goal* $G$ se todas as ações de $P$ são necessárias. Uma ação $[\alpha_i]$ é necessária se:

i) $[\alpha_i]$ ocasiona uma Expansão de uma das propriedades de $G$; ou

ii) $[\alpha_i]$ ocasiona uma Expansão de uma propriedade que é pré-condição de uma ação necessária.

Um plano $P$ é dito *ótimo* para um *Goal* $G$ a partir de um Cenário Inicial $C_0$ se não houver um plano de tamanho menor que $P$, que solucione $G$ a partir de $C_0$.

Segue que todo plano ótimo é também minimal.

Um gerador de planos é *completo* se ele fornecer um plano ótimo para qualquer *Goal* $G$, sempre que esse plano existir.

## 8.7.2 Prova de Completude

A proposição abaixo estabelece a Completude do algoritmo, no sentido de que se existe um plano ótimo que solucione um determinado *Goal* a partir de um Cenário Inicial dado, então existe uma derivação desse plano pelo algoritmo.

**Proposição 8.1** *Dado um plano $P$ de tamanho $N$, tal que $P$ é ótimo para um* Goal $G = a_1,a_2,\ldots,a_n$ *a partir do Cenário $C_0$, existe uma derivação de $P$ pelo algoritmo.*

**Prova:**

Se $N = 0$ é porque $C_0$ satisfaz $G$ e então o algoritmo termina na primeira iteração retornando um plano vazio.

Suponha que o algoritmo forneça planos ótimos de tamanho $k-1$.

Seja um plano ótimo $P = [\alpha_1][\alpha_2]...[\alpha_k]$, que resolve o *Goal* $G = a_1, a_2, ..., a_n$ a partir de $C_0$. Como $P$ é ótimo, é também minimal. Logo $[a_k]$ é necessária e resolve pelo menos um dos componentes de $G$ (pois não pode adicionar nenhuma propriedade necessária a uma outra ação, uma vez que é a última). Seja $G' = r_1, r_2, ..., r_m$ um *Goal* formado pelos elementos de $G$ satisfeitos pelo plano $P' = [\alpha_1][\alpha_2]...[\alpha_{k-1}]$. Pela Hipótese de Indução, o plano $P'$ é obtido pelo algoritmo para o *Goal* $G'$.

Isso significa que na $(k-1)$-ésima iteração, uma regra associada à ação $\alpha_{k-1}$ foi escolhida e que o cenário resultante de sua aplicação satisfaz $G'$ na $k$-ésima iteração.

Suponha agora que ao invés de $G'$, o *Goal* $G$ tenha sido submetido ao algoritmo. Por construção, existem elementos em $G$ que não são resolvidos por $P'$. Conseqüentemente, o cenário resultante da aplicação de $\alpha_{k-1}$ não satisfará o *Goal* $G$ na iteração $k$ e o algoritmo prosseguirá por mais uma iteração.

Num determinado instante, uma regra associada à ação $\alpha_k$ será escolhida e aplicada ao cenário recebido na iteração $k$. O algoritmo será então chamado novamente, mas como $\alpha_k$ resolve o restante dos componentes de $G$, o cenário recebido na iteração $k+1$ satisfará o *Goal* de forma que o algoritmo dessa iteração retorna um plano vazio.

Analogamente, o algoritmo da iteração $k$ retorna um plano igual a $[\alpha_k]$ que adicionada ao plano $P'$ resulta no plano $P$.

□

A Figura 8.9 ilustra o esquema de execução do método no ramo de sucesso. $C_0$ = Cenário Inicial, e por abuso de linguagem $[\alpha_i]C$ denota o cenário obtido a partir de $C$ pela execução de $\alpha_i$.

**Iteração 1**

Prova($C_0$,$G$)

Ação $\alpha_1$ escolhida.

$\vdots$

**Iteração k**

Prova($[\alpha_1][\alpha 2]...[\alpha_{k-1}]C_0$,$G$)

Ação $\alpha_k$ escolhida.

**Iteração k + 1**

Prova($[\alpha_1][\alpha 2]...[\alpha_k]C_0$,$G$)

O cenário $[\alpha_1][\alpha 2]...[\alpha_k]C_0$ satisfaz $G$

Figura 8.9: Esquema de execução no ramo de sucesso

## 8.8 Conclusões

Esse capítulo propôs o uso dos mecanismos de Mudanças Epistemológicas a fim de se evitar o Problema de *Frame* num formalismo voltado para o Problema da Geração de Planos.

Por utilizar uma Linguagem semelhante à Linguagem das Cláusulas de Horn, o formalismo poderia usar uma máquina PROLOG a fim de executar as provas de propriedades necessárias. Quando o *Goal* não pode ser refutado de um estado do sistema (representado por um Conjunto Definido), parte da prova incompleta poderia ser usada por uma heurística a fim de se determinar a melhor regra a aplicar ao cenário e prosseguir com a refutação, numa estratégia semelhante à do STRIPS.

As mudanças de estados seriam então realizadas através das funções de Contração e Expansão definidas no Capítulo 7. A Completude do gerador de planos implementado em PROLOG, contudo, só seria atingida se fosse possível tornar o processo de busca dessa linguagem completo.

# Capítulo 9

# CONCLUSÕES

Este trabalho pode ser visto como uma aplicação da Programação em Lógica ao Problema da Geração de Planos.

Na primeira parte foi proposto um formalismo, o Prolog Modal de Ação, com semântica procedural similar à do PROLOG. Esse formalismo é baseado numa Lógica Modal de Ação e permite a utilização de literais com modalidades, podendo ser considerado uma extensão do método de Resolução-LSD.

A representação de um problema é efetuada em duas partes. A primeira contem a descrição do estado inicial do sistema, através de fatos e a segunda, chamada de Especificação, um conjunto de Cláusulas Definidas e Cláusulas Definidas Modais. As cláusulas definidas podem ser aplicadas aos fatos do cenário inicial para obter a derivação de novas propriedades. Derivações de propriedades de outros cenários que não o Inicial são obtidas através da aplicação das regras Modais.

A distinção é importante porque é assumido que uma regra do tipo da regra da Necessitação dos Sistemas Modais pode ser aplicada sobre as cláusulas da Especificação de modo que essas cláusulas representem propriedades de todos os Cenários.

Sob o ponto de vista de uma Lógica Modal, o sistema possui o axioma $K$, que foi utilizado para preservar a implicação lógica através dos cenários. A seqüência de ações obtida na investigação desses cenários futuros é utilizada como um Plano para obtenção de propriedades desejadas fazendo os sistema funcionar como um formalismo adequado ao problema da geração de planos.

Nesta etapa do trabalho não foram considerados mecanismos do tipo da Negação por Falha Finita a fim de se obter negação e simplificar a especificação dos problemas.

Uma extensão do trabalho seria efetuar uma análise da utilização desses mecanismos de obtenção de negação. Sobre esse aspecto existem diversas alternativas já propostas, como por exemplo o trabalho de [GS86].

A análise seria interessante pois a negação simplifica bastante a especificação dos problemas de geração de planos. Em particular, porque a definição das regras associadas às ações podem incluir às vezes muitas restrições de aplicação, sobretudo sobre os objetos manipulados. O ideal é que as regras sejam tão genéricas quanto possível e essa generalidade é mais facilmente obtida se for possível estipular quais objetos não podem ser manipulados pelas mesmas.

Em alguns casos, a Hipótese de Mundo Fechado parece bastante atraente para reduzir o esforço de especificação. Por outro lado, a Negação por Falha Finita não parece capaz de captar o sentido pretendido pela negação na definição dos problemas. Tomemos como exemplo a seguinte Cláusula, utilizada na representação do Problema dos Três Blocos do Capítulo 6:

$$[emp(X,Y)]limpo(Z) \leftarrow dif(Z,Y), limpo(Z)$$

O Objetivo do predicado $dif(Z,Y)$ aí é simplesmente estabelecer que o único bloco que deixa de estar sem nenhum outro sobre ele após a ação de empilhar um bloco $X$ sobre um bloco $Y$ é o próprio bloco $Y$. Suponha que uma Cláusula do tipo $igual(X,X)$ tivesse sido utilizada para descrever que qualquer bloco $X$ é igual ao bloco $X$, utilizando o processo de unificação para efetuar o teste e a Hipótese de Mundo Fechado para deduzir que tudo o mais é diferente. Com a Negação por Falha Finita seria intuitivo então requerer a desigualdade entre o bloco $X$ e o bloco $Y$ com o literal $not(igual(X,Y))$. Dessa forma, não seria necessário explicitar a igualdade ou desigualdade entre todos os elementos do Domínio. Essa alternativa apesar de atraente não fornece uma solução para o problema. Suponha que a variável $Y$ esteja instanciada à constante $a$, mas a variável $Z$ esteja livre. A prova para $not(igual(Z,a))$ vai falhar pois esse é um literal negativo mas não é básico. Contudo, o sentido pretendido era inicialmente achar algum $Z$ que não fosse igual a $a$. Esse problema em alguns casos poderia ser contornado se a escolha de literais com variáveis livres fosse adiada até que essas variáveis se tornassem instanciadas, principalmente no caso de literais negativos, como proposto no sistema MU-PROLOG.

Uma deficiência do sistema é ter que utilizar regras de *frame*. Isso faz com que um grande número de regras tenham que ser acrescentadas de modo a possibilitar a dedução em cenários futuros de propriedades que não são afetadas pela execução das

ações. Além disso, em termos de computação, existe um grande esforço computacional devido à necessidade de trazer propriedades via regras de *frame* de cenário em cenário, quando essas propriedades são necessárias num cenário específico.

Em função dessa deficiência, decidiu-se analisar o comportamento das Mudanças Epistemológicas em Conjuntos de Cláusulas Definidas, tendo em vista sua utilização no Problema da Geração de Planos. Essa análise foi efetuada no Capítulo 7, onde foram definidos modos de se realizar as Contrações e Expansões e suas relações com os postulados originais propostos por [Gär88]. Como a Revisão pode ser definida em função das Expansões e Contrações, sua utilização em Conjuntos pseudo-Definidos foi também considerada.

Finalmente, foi apresentada uma versão do formalismo introduzido na primeira etapa, que utiliza as funções de Contração e Expansão do Capítulo 7 a fim de obter as mudanças de cenários associadas às execuções das ações. Essa versão ainda pode ser vista como tendo semântica procedural similar à do PROLOG. No entanto, o conjunto de Fatos que antes eram associados ao Cenário Inicial agora toma uma dimensão dinâmica, pois passa a ser atualizado em função da execução de cada ação. Sob o ponto de vista do *Goal*, a prova para o mesmo é executada da mesma forma que seria numa máquina PROLOG comum. Quando a prova de um literal falha é requerida a aplicação de uma regra modal para possibilitar a continuação da prova. O uso de uma heurística nesse ponto seria de grande interesse, pois reduziria o espaço de busca.

Não há uma implementação da versão com Revisão de Crenças. Essa é uma outra proposta de extensão desse trabalho.

# REFERÊNCIAS BIBLIOGRÁFICAS


[AM85] C. E. Alchourrón and D. Makinson. On the logic of theory change: Safe contraction. *Studia Logica*, 44:405–422, 1985.

[BM91] M. R. F. Benevides and T. S. E. Maibaum. Avoiding the frame-problem - a proof theoretical approach for the planning problem, 1991.

[CdC87] J. Cunningham and M. da Costa. Computational logic and modal action logic. Technical report, GEC Research Laboratories, Marconi Research Centre, Great Baddow, Chelmsford, England, 1987. Alvey FOREST Deliverable Report.

[CGF87] Marco A. Casanova, Fernando A. C. Giorno, and Antônio L. Furtado. *Programação em Lógica e a Linguagem Prolog.* Editora Edgard Blücher Ltda., Caixa Postal 5450 - CEP 01051 - São Paulo - SP - Brasil, 1987.

[CKRP73] A. Colmerauer, H. Kanoui, P. Roussel, and R. Pasero. Un système de communication homme-machine en français. Technical report, Université d'Aix-Marseille, 1973. Groupe de Recherche en Intelligence Artificielle.

[Cla] K. L. Clark. Predicate logic as a computational formalism. Technical report, Department of Computing, Imperial College. Research report 79/59.

[Cla78] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Databases.* Plenum Press, 1978.

[dC86] L. Fariñas del Cerro. Molog: A system that extends prolog with modal logic. *New Generation Computing*, 4:35–50, 1986.

[FN71] R. E. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. In *Proceedings of 2nd IJCAI*, Imperial College, London, England, 1971.

[FR86] Norman Foo and A. Rao. *DYNABELS.* Department of Computer Science, Sydney University, 1986.

[FUV83] R. Fagin, J. D. Ullman, and M. Y. Vardi. On the semantics of updates in databases. Technical report, Proceedings of Second ACM SIGACT-SIGMOD, New York: Association for Computing Machinery, 1983.

[Gär88] Peter Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States.* A Bradford Book - The MIT Press, Cambridge, Massachusetts - London, England, 1988.

[GN69] Ernst G. and A. Newell. Gps: A case study in generality and problem solving. In *ACM Monograph Series.* Academic Press, New York, New York, 1969.

[GS86] D. M. Gabbay and M. J. Sergot. Negation as inconsistency 1. *Journal of Logic Programming*, 1:1–35, 1986.

[Her67] J. Herbrand. Researches in the theory of demonstration. In J. van Heijenoort (Ed.), editor, *From Frege to Gödel: A Source Book in Mathematical Logic, 1879-1931*, pages 525–581. Harvard University Press, 1967.

[Hil74] R. Hill. Lush-resolution and its completeness. Technical report, Department of Artificial Intelligence, Edinburgh University, 1974. DCL Memo 78.

[Kho88] Samit Khosla. *System Specification: A Deontic Approach.* PhD thesis, Department of Computing, Imperial College, U.K., 1988.

[Kow74] R. A. Kowalski. Predicate logic as a programming language. In *IFIP-74*, pages 569–574, 1974.

[Lak70] I. Lakatos. *Falsification and the methodology of scientific research programmes.* I. Lakatos and A. Musgrave, eds. Cambridge: Cambridge University Press, 1970.

[Lev7a] I. Levi. Direct inference. *The Journal of Philosophy*, 74:5–29, 1977a.

[Lif87] V. Lifschitz. On the semantics of STRIPS. In M. Georgeff and A. Lansky, editors, *Proceedings of the Conference of Reasoning about Actions and Plans*, Los Altos, CA, USA, 1987. Morgan Kaufmann.

[Llo84] J. W. Lloyd. *Foundations of Logic Programming.* Symbolic Computation. Springer-Verlag, 1984.

[Mai87] T.S.E. Maibaum. A logic for the formal requirements specification of real-time/embedded systems. Technical report, Dept. of Computing, Imperial College, London, 1987.

[Nil80] N. J. Nilsson. *Principles of Artificial Intelligence.* Morgan Kaufmann Publishers, Inc., Los Altos,California 94022, 1980.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *JACM*, 12(1):23–41, January 1965.

[Sus73] G. J. Sussman. *HACKER: A computational model of skill acquisition.* PhD thesis, Massachussets Institute of Technology, 1973.

[War74] David H. D. Warren. A system for generating plans. Technical report, Department of Artificial Intelligence, Edinburgh University, June 1974. Memo 76.