Revisão de modelos CTL

Paulo de Tarso Guerra Oliveira

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação
Orientador: Profa. Dra. Renata Wassermann

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq

São Paulo, dezembro de 2010

Revisão de Modelos CTL

Esta versão definitiva da dissertação
contém as correções e alterações sugeridas pela
Comissão Julgadora durante a defesa realizada
por Paulo de Tarso Guerra Oliveira em 16/12/2010.

Comissão Julgadora:

- Profa. Dra. Renata Wassermann (orientadora) - IME-USP
- Profa. Dra. Leliane Nunes de Barros - IME-USP
- Prof. Dr. Paulo Eduardo Santos - FEI

A Deus, sempre.
A Viviane, por todo apoio e amor.

Agradecimentos

Primeiro, agradeço a Deus pelo dom da vida, por todas as oportunidades e por todas as realizações que Ele me proporcionou.

Agradeço aos meus pais, pelo apoio e dedicação incondicional para que eu alcançasse meu objetivo. Agradeço toda minha família, por toda oração e força que recebi nestes anos.

Agradeço muito a minha namorada, por todo seu companheirismo e amor. Agradeço por todos os momentos que vivemos juntos, por todas as experiências que passamos. Mais uma vez, obrigado por me fazer feliz.

Agradeço a minha orientadora, professora Renata Wassermann, por guiar-me pelos caminho certos e tornar possível a realização deste trabalho. Agradeço a todos os meus professores, em especial àqueles do grupo LIAMF, por todo apoio acadêmico e por toda dedicação em compartilhar conhecimento.

Agradeço ao Conselho Nacional de Desenvolvimento Científico e Tecnológico, cujo financiamento tornou este trabalho possível, e ao Instituto de Matemática e Estatística (IME) da Universidade de São Paulo (USP) que acolheu-me e tornou-se minha segunda casa.

Por fim, agradeço a todos os meus amigos, de todas as naturalidades e nacionalidades, sem os quais esses anos de mestrado não teriam sido os mesmos.

Resumo

Verificação de modelos é uma das mais eficientes técnicas de verificação automática de sistemas. No entanto, apesar de poder lidar com verificações complexas, as ferramentas de verificação de modelos usualmente não fornecem informação alguma sobre como reparar inconsistências nestes modelos. Nesta dissertação, mostramos que abordagens desenvolvidas para a atualização de modelos CTL inconsistentes não são capazes de lidar com todos os tipos de alterações em modelos. Introduzimos então o conceito de revisão de modelos: uma abordagem baseada em revisão de crenças para o reparo de modelos inconsistentes em um contexto estático. Relacionamos nossa proposta com trabalhos clássicos em revisão de crenças. Definimos um operador de revisão de modelos e mostramos que este obedece postulados de racionalidade clássico de revisão de crenças. Propomos um algoritmo de revisão com base no algoritmo utilizado pela abordagem de atualização de modelos. Discutimos sobre problemas e limites do algoritmo proposto, e mostramos que essa estratégia de adaptação não é uma solução apropriada.

Palavras-chave: revisão de crenças, lógica de árvore computacional, verificação de modelos.

Abstract

Model checking is one of the most robust techniques in automated system verification. But, although this technique can handle complex verifications, model checking tools usually do not give any information on how to repair inconsistent system models. In this dissertation, we show that approaches developed for CTL model update cannot deal with all kinds of model changes. We introduce the concept of CTL model revision: an approach based on belief revision to handle system inconsistency in a static context. We relate our proposal to classical works on belief revision. We define an operator for model revision and we show that it obeys the classical rationality postulates of belief revision. We propose an algorithm for model revision based on the algorithm used by the model update approach. We discuss problems and limitations of our proposed algorithm and show that this strategy of adaptation is not an appropriate solution.

Keywords: belief revision, computation tree logic, model checking.

Sumário

Lista de Figuras **ix**

1 Introdução **1**
1.1 Motivação e Objetivos 2
1.2 Contribuições 3
1.3 Organização do Trabalho 3

2 Lógica de Árvore Computacional **5**
2.1 Sintaxe e Semântica da Lógica Temporal CTL 5
2.2 Verificação de Modelos CTL 8
2.2.1 Algoritmo de Verificação por Rotulação 9

3 Revisão de Crenças **11**
3.1 Estados Epistêmicos: Representação e Operadores 11
3.2 Teoria AGM 12
3.2.1 Postulados de Racionalidade 12
3.2.2 Relações de Identidade 13
3.3 Construções 14
3.3.1 Partial Meet 14
3.3.2 Sistema de Esferas 15
3.4 Reformulação da Teoria AGM 17
3.4.1 Reformulação dos Postulados 17

4 Atualização de Crenças **19**
4.1 Diferença entre Atualização e Revisão de Crenças 19
4.2 Abordagem de Modelos Possíveis 21
4.3 Postulados de Racionalidade 21
4.3.1 Postulados KM para Atualização 21
4.3.2 Postulados KM para Remoção 22
4.3.3 Relações entre Atualização e Remoção 22
4.4 Construções 23

5 Atualização de Modelos **25**
5.1 Definição do Princípio da Mudança Mínima em Modelos CTL 25
5.2 Propriedades Semânticas da Atualização de Modelos CTL 28
5.3 Propriedades Computacionais da Atualização de Modelos CTL 28

5.3.1 Subclasse Tratável da Atualização de Modelos CTL 29

6 Revisão de Modelos **31**

6.1 Atualização Modelos em um Contexto Estático 31

6.2 Revisão Clássica e Lógica Temporal CTL 33

6.3 Primeiras abordagens: Belief Revision NuSMV 34

6.4 Operador de Revisão de Modelos 34

6.4.1 Propriedades Semânticas 35

7 Algoritmo para Revisão de Modelos CTL **37**

7.1 Considerações Preliminares 37

7.2 Principais Funções do Algoritmo 38

7.2.1 Lidando com Operadores Temporais 39

7.2.2 Lidando com Fórmulas Proposicionais 41

7.2.3 Lidando com Conjunções, Disjunções e Negação 42

8 Discussão **47**

9 Conclusões **51**

9.1 Trabalhos Futuros 51

A Propriedades Lógicas **53**

Referências Bibliográficas **55**

Lista de Figuras

2.1 Operadores temporais da lógica CTL. 6
2.2 Exemplo de estrutura de Kripke 7
2.3 Rotulação de estados com a subfórmula $AF\psi_1$ 9
2.4 Rotulação de estados com a subfórmula $E(\psi_1 U \psi_2)$ 10

3.1 Representação gráfica de um sistema de esferas. 15
3.2 Revisão $K * \alpha$ no sistema de esferas. 16

4.1 Ilustração do conceito de proximidade entre modelos. 20

5.1 Modelo exemplo M 26
5.2 À esquerda, a atualização de M utilizando PU2 e PU5. À direita, a atualização de M utilizando PU3. 26
5.3 Ilustração do conceito de *ordenação por proximidade*. 27

6.1 Modelagem que satisfaz a especificação ψ. 32
6.2 Exemplos de modelos de ψ que não satisfazem ϕ. 32
6.3 Exemplos de possíveis atualizações de um modelos. 33
6.4 Eliminando da inconsistência no BrNuSMV [Sou07]. 34

8.1 Problema de utilizar uma entrada limitada. 48
8.2 Problema de não gerar todas as modificações minimais. 49

Capítulo 1

Introdução

Verificar formalmente um sistema é uma maneira de assegurar que erros de projeto não se propaguem para a implementação do sistema. No entanto, as atuais ferramentas de verificação não possuem mecanismos que auxiliem o projetista no reparo de modelos de sistema incorretos. Neste trabalho apresentamos uma abordagem baseada em revisão de crenças para o reparo destes modelos, a qual denominamos *Revisão de Modelos*.

A importância da verificação de sistemas fica evidente no fracasso no primeiro lançamento do projeto Ariane 5. Em 4 de junho de 1996, o foguete francês Ariane 501 explodiu segundos após seu lançamento ao espaço. As condições meteorológicas eram aceitáveis, o que dificultou ainda mais entender o porquê da explosão. Uma comissão de investigação concluiu que o incidente foi ocasionado por um erro no *software* responsável pelo movimento do foguete, mais especificamente, por um erro de conversão entre um ponto flutuante de 64-bits e um inteiro de 16-bits. Após este lançamento mal sucedido, as equipes envolvidas no projeto Ariane 5 passaram a realizar extensivas verificações no modelo dos futuros foguetes, para que anomalias não pusessem em risco as novas missões.

A complexidade de alguns sistemas faz com que seus projetos estejam mais suscetíveis a erros. Ao mesmo tempo, detectar tais erros se torna uma tarefa cada vez mais difícil. Os sistemas computacionais estão cada vez mais presentes no nosso cotidiano (redes de telefonia, sistemas bancários, controle de semáforos, etc), com isso cresce a demanda por métodos que atestem a correção destes sistemas.

Proposta por Clark e Emerson, a *verificação de modelos* [CES86, CGP99, HR04] permite que o projetista verifique quando um modelo de sistema satisfaz (ou não) uma dada especificação formal. Uma especificação formal nada mais é que o conjunto de propriedades que definem o comportamento do sistema. Quando alguma dessas propriedades não é satisfeita, temos o que chamamos de inconsistência de especificação. Nas últimas décadas, trabalhos como [ADK+05, CCK+02, CCGR99] aprimoraram significativamente a técnica de verificação de modelos, resultando em rápidos algoritmos e no desenvolvimento de eficientes ferramentas de verificação.

Contudo, a maioria das ferramentas de verificação, quando encontram uma inconsistência de especificação, apenas informam o caminho que a ocasiona (caminho contraexemplo). Essa informação é extremamente útil para projetistas, pois poderão direcionar o reparo para um ponto específico do modelo. Porém, pode não ser suficiente para indicar como o modelo deve ser corrigido. Quanto mais complexo o sistema, mais complexa é sua modelagem e assim, mais difícil é corrigir seus erros. Em geral, ferramentas de verificação não possuem mecanismos capazes de auxiliar o projetista na tarefa de correção, o que não as credencia como ferramentas completas para o reparo de sistemas.

Sendo possível implementar a verificação de modelos por meio de algoritmos rápidos e eficazes, foi natural buscar ferramentas capazes de corrigir automaticamente modelos inconsistentes. Essa busca conduziu a pesquisas sobre métodos para geração automática de modificações, que fossem passíveis de integração com as ferramentas de verificação existentes.

O trabalho [BEGL99] foi o primeiro nessa linha. Buccafurri et al. desenvolveram um arcabouço formal integrando verificação de modelos e revisão abdutiva de teorias para realizar o diagnóstico e

reparo de erros em programas concorrentes. O raciocínio abdutivo é utilizado para encontrar modificações no sistema a fim de que este satisfaça todas as propriedades de uma especificação formal. Nessa abordagem, um sistema é modelado segundo uma estrutura de Kripke[1] e as modificações correspondem a uma sequência de adições ou remoções de transições de estados que torna o modelo consistente com a especificação.

Apesar da abordagem de Buccafurri et al. ser bem sucedida para seus propósitos, o conceito de modificação adotado é um tanto quanto restrito. Modificar apenas transições entre estados nem sempre é suficiente para que um modelo satisfaça uma dada propriedade. Em diversas situações, o reparo de um modelo só pode ser obtido através da modificação de outros elementos da estrutura de Kripke, como por exemplo, por meio da adição de um novo estado ou da alteração da função de rotulação. As pesquisas então seguiram por métodos que suprissem as deficiências da abordagem de [BEGL99].

1.1 Motivação e Objetivos

O dinamismo de conjuntos de crenças [AGM85, Win88] é um tópico bastante estudado em Inteligência Artificial. Este dinamismo se refere à como um agente deve mudar suas crenças frente a necessidade de agregar novas informações, mesmo que elas sejam conflitantes com o que ele acredita ser verdade. Se no contexto de verificação de modelos virmos a especificação como nossa crença e a propriedade inconsistente como a informação a ser agregada, uma técnica baseada em mudança de crenças nos indicaria como mudar nossos modelos a fim de satisfazer dada propriedade. Integrar verificação de modelos e mudança de crença pode vir a gerar uma poderosa ferramenta para o reparo automático de modelos.

A mudança de conjunto de crenças possui duas principais vertentes: *revisão de crenças* [AGM85] e *atualização de crenças* [KM91, HR99]. A revisão de crenças trata de como um agente deve incorporar informações sobre um mundo estático, onde a nova crença refina ou corrige o conhecimento do agente sobre o mundo. Já a atualização de crenças lida com informações provenientes de mudanças no mundo, mudanças essas que alteram a maneira como o agente deve interpretá-lo.

Zhang e Ding [ZD08] propuseram um arcabouço para atualização de modelos da lógica temporal CTL por meio da integração de verificação de modelos e atualização de crenças. Eles especificaram o princípio da mudança mínima para atualização de modelos e então definiram o conceito de atualização admissível. Os autores também especificaram um procedimento para a atualização de modelos e analisaram suas propriedades semânticas e computacionais.

Apesar da similaridade entre atualização e revisão de crenças, o uso da abordagem incorreta pode acarretar em perda de informação. Neste trabalho, mostramos que a abordagem baseada em atualização de crenças não é apropriada para todos os casos. Argumentamos que é mais adequado utilizar revisão de crenças para mudanças em modelos, quando o reparo de inconsistências de especificações ocorrer em um contexto estático.

Souza e Wassermann [SW07] abordam o uso prático de revisão de crenças para o reparo de modelos inconsistentes. Os autores criaram uma ferramenta capaz de gerar sugestões de reparo para modelos descrito segundo a linguagem SMV. Acreditamos no entanto que a técnica poderia explorar melhor os conceitos de revisão. Acreditamos que é possível desenvolver uma abordagem que possua fundamentos sólidos na teoria clássica de revisão de crenças.

Nosso objetivo é propor um arcabouço formal para revisão de modelos, onde possamos explorar conceitos de revisão de crença no contexto de mudanças em conjuntos de modelos. Nossa abordagem não pretende ser um substituto a atualização de modelos, mas sim uma técnica complementar no objetivo maior de construir um reparador automático de modelos.

[1] Detalhes sobre estruturas de Kripke serão abordados no Capítulo 2.

1.2 Contribuições

As principais contribuições deste trabalho são:

1. Esclarecemos a diferença entre revisão de modelos e atualização de modelos. Mostramos por meio de exemplos que existem situações onde atualização gera resultados menos significativos do que o esperado. Mostramos que revisão de modelos é adequada para essas situações, por preservar informações desejáveis em seu resultado.

2. Propomos um arcabouço formal para revisão de modelos CTL. Utilizamos as operações primitivas de Zhang e Ding [ZD08] para definir o critério de ordenação para identificar conjuntos de modelos alterados minimamente. Então estudamos a relação entre nossa proposta de revisão de modelos CTL e a abordagem tradicional de revisão de crenças. Em especial mostramos que nosso operador de revisão de modelos obedece os postulados AGM para revisão (conforme reformulação de [KM91]).

3. Desenvolvemos um algoritmo formal para revisão de modelos CTL. Dados um conjunto de modelos que satisfazem uma especificação e uma propriedade expressa por uma fórmula CTL, nosso algoritmo é capaz de gerar modelos que satisfazem esta propriedade e que se diferenciam minimamente daqueles fornecidos como entrada. Estes modelos gerados podem ser vistos como possíveis correções para a especificação original.

1.3 Organização do Trabalho

O restante deste trabalho pode ser dividido em três partes principais: a primeira parte, capítulos 2 à 5, introduz conceitos relativos a nossa abordagem; a segunda parte, capítulos 6 à 8, concentra nossas contribuições; a última parte é composta pelo capítulo 9, que trata das considerações finais.

No Capítulo 2 apresentamos a *lógica de árvore computacional*. Especificamos a sintaxe e semântica das fórmulas dessa lógica e definimos formalmente o conceito de modelo CTL. Também formalizamos o conceito de verificação de modelos CTL e exibimos um algoritmo de verificação simples.

No Capítulo 3 apresentamos a teoria de *revisão de crenças*. Introduzimos neste capítulo o conceito de estado epistêmico, maneiras de representá-los e quais são as possíveis mudanças nestes estados. Apresentamos também a Teoria AGM, seus postulados de racionalidade e construções de funções para contração e revisão.

No Capítulo 4 apresentamos a teoria de *atualização de crenças*. Mostramos sua diferença fundamental da teoria de revisão de crenças, um novo conjunto de postulados de racionalidade e um exemplo típico de operador de atualização.

No Capítulo 5 mostramos a abordagem de Zhang e Ding [ZD08] para mudanças em modelos CTL, a *atualização de modelos CTL*. Apresentamos sua métrica para proximidade entre modelos, bem como seu princípio de mudança mínima. Mostramos sua definição de operador de atualização de modelos e parte de sua análise das propriedades semânticas e computacionais da abordagem.

No Capítulo 6 mostramos nossa abordagem para mudanças em modelos CTL, a *revisão de modelos CTL*. Apresentamos a principal diferença em relação à abordagem de [ZD08], definimos uma nova relação de ordem entre modelos e formalizamos nosso operador de revisão de modelos CTL. Analisamos ainda a correspondência da nossa abordagem com a Teoria AGM.

No Capítulo 7 apresentamos nosso algoritmo para revisão de modelos CTL. Recebendo como entrada um conjunto de modelos que satisfazem uma especificação formal e uma propriedade (expressa por uma fórmula CTL), mostramos como calcular um conjunto de modelos que satisfaçam a propriedade e sejam minimamente próximos aos modelos da especificação formal. Nosso algoritmo é baseado nas definições do Capítulo 6.

No Capítulo 8 discutimos questões relacionadas ao algoritmo apresentado. Mostramos as motivações de nossas decisões, apresentamos as limitações do atual algoritmo e as dificuldades relacionadas ao desenvolvimento de um revisor de modelos.

Por fim, apresentamos no Capítulo 9 algumas considerações finais, onde resumimos nossas conclusões, contribuições e listamos tópicos para trabalhos futuros.

Capítulo 2

Lógica de Árvore Computacional

Lógica temporal é um tipo de lógica modal onde é possível representar e raciocinar sobre o tempo. A lógica de árvore computacional [CES86, CGP99], ou simplesmente CTL (do inglês, *computation tree logic*), modela o futuro como ramificações de uma árvore. Aplicada a sistemas computacionais, a lógica temporal CTL é capaz de quantificar sobre os diversos caminhos que uma execução pode tomar, como por exemplo, se dois processos estarão utilizando simultaneamente o mesmo recurso, em um momento qualquer, em algum dos caminhos de execução possíveis.

Neste capítulo apresentaremos a lógica temporal CTL. Na Seção 2.1 veremos como é composto o conjunto de fórmulas da CTL e qual o significado de seus operadores temporais. Na Seção 2.2 introduziremos formalmente o conceito de verificação de modelos CTL e mostraremos um simples algoritmo de verificação.

2.1 Sintaxe e Semântica da Lógica Temporal CTL

Ao definir uma lógica, primeiro devemos especificar como se compõe nessa lógica uma fórmula bem formada. A Definição 2.1 define a linguagem da lógica temporal CTL por meio de uma gramática BNF.

Definição 2.1 (Sintaxe)**.** A lógica CTL é sintaticamente definida pela seguinte *Backus Naur Form*:

$$\begin{aligned}\phi ::= & \top \mid \bot \mid p \mid (\neg\phi) \mid (\phi \vee \phi) \mid (\phi \wedge \phi) \mid (\phi \rightarrow \phi) \mid \mathrm{EX}\,\phi \mid \\ & \mathrm{AX}\,\phi \mid \mathrm{EF}\,\phi \mid \mathrm{AF}\,\phi \mid \mathrm{EG}\,\phi \mid \mathrm{AG}\,\phi \mid \mathrm{E}[\,\phi\,\mathrm{U}\,\phi\,] \mid \mathrm{A}[\,\phi\,\mathrm{U}\,\phi\,]\end{aligned}$$

onde p é um átomo proposicional, $\neg$, $\vee$, $\wedge$ e $\rightarrow$ são operadores lógicos clássicos e os demais constituem os operadores temporais.

Sejam p, q e r átomos proposicionais, as seguintes fórmulas são exemplos de fórmulas CTL bem formadas, conforme a Definição 2.1:

1. $\mathrm{EX}\,(p \rightarrow \mathrm{AX}\,q)$
2. $\mathrm{EF}\,\mathrm{EG}\,(\neg p)$
3. $\mathrm{E}[\,p\,\mathrm{U}\,\mathrm{E}[\,q\,\mathrm{U}\,r\,]\,]$
4. $\mathrm{A}[\,\mathrm{E}[\,p\,\mathrm{U}\,q\,]\,\mathrm{U}\,\mathrm{AF}\,(p \vee r)\,]$
5. $(p \rightarrow (\neg\mathrm{EF}\,q))$
6. $(\mathrm{EG}\,\mathrm{EF}\,p \rightarrow \mathrm{AG}\,\mathrm{AF}\,q)$
7. $(((p \wedge q) \wedge r) \rightarrow \mathrm{EG}\,((p \wedge q) \wedge r)))$
8. $(\neg(\mathrm{E}[\,(\neg q)\,\mathrm{U}\,((\neg p) \wedge (\neg q))\,] \vee \mathrm{EG}\,(\neg q)))$

Também segundo a Definição 2.1, as seguintes fórmulas não são fórmulas CTL bem formadas:

1. FEp
2. EXGp
3. EAF p
4. A$\neg$G$(p \vee q)$
5. A$[\,(p \,\mathrm{U}\, q) \vee (q \,\mathrm{U}\, r)]$

Muitas vezes podemos omitir parênteses em fórmulas CTL sem alterar seu significado. Por exemplo, $(p \to (\neg\mathrm{EF}\, q))$ pode ser escrita como $p \to \neg\mathrm{EF}\, q$ sem que introduzamos ambiguidade a fórmula. Assim, sempre que possível omitiremos tais parênteses em favor da legibilidade.

Cada operador temporal é composto por um quantificador de caminho (E, "existe um caminho", ou A, "para todos os caminhos") seguido por um operador de estado (X, "próximo estado no caminho", U, "até que no caminho", G, "globalmente no caminho", ou F, "futuramente no caminho")[1]. A interpretação destes operadores é ilustrada na Figura 2.1.

Utilizando esses operadores temporais podemos expressar sentenças como: sempre que tenho sono, acabo dormindo em algum momento ($\mathrm{AG}\,(\texttt{sono} \to \mathrm{AF}\,\texttt{dormir})$); é impossível que eu durma e não tenha sono ($\neg\mathrm{EF}\,(\texttt{dormir} \wedge \neg\texttt{sono})$); sempre fico sem sono logo após tomar café ($\texttt{cafe} \to \mathrm{AX}\,\neg\texttt{sono}$); fico com sono até que eu durma ou tome café ($\mathrm{A}[\,\texttt{sono}\,\mathrm{U}\,(\texttt{dormir} \vee \texttt{cafe})\,]$); é possível em algum momento no futuro que eu passe a tomar café para sempre e nunca volte a dormir ($\mathrm{EF}\,\mathrm{AG}\,(\texttt{cafe} \wedge \neg\mathrm{EF}\,\texttt{dormir})$).

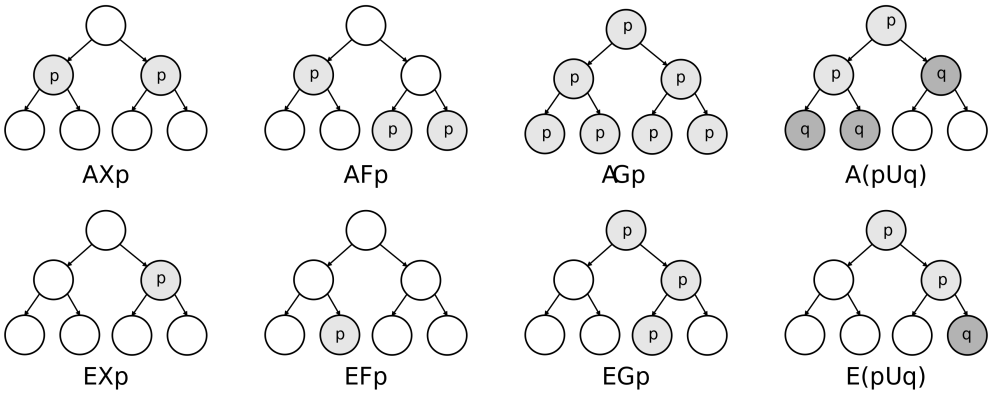

Figura 2.1: *Operadores temporais da lógica CTL.*

Para uma definição formal da semântica da lógica CTL, é necessário antes definir o conceito de estrutura de Kripke. Uma estrutura de Kripke é composta por: um conjunto de estados; um conjunto de estados iniciais; uma relação entre estes estados; e uma associação entre cada estado e um conjunto de valores proposicionais.

Definição 2.2. Seja P um conjunto finito de proposições booleanas, uma *estrutura de Kripke* é uma quádrupla $M = (S, S_0, R, L)$, onde:

- S é um conjunto de estados.
- $S_0 \subseteq S$ é o conjunto de estados iniciais.
- $R \subseteq S \times S$ é uma relação total entre os estados, isto é, para todo $s \in S$ existe $s' \in S$ tal que $R(s, s')$.
- $L : S \to 2^P$ é uma função que rotula cada estado com um conjunto de proposições verdadeiras naquele estado.

[1] As letras E, A, X, U, G e F são acrônimos do inglês para as expressões "*there Exists a path*", "*for All paths*", "*neXt time*", "*Until*", "*Globally*" e "*in the Future*", respectivamente.

A Figura 2.2 ilustra uma estrutura de Kripke $M = (S, S_0, R, L)$ retirada de [Sou07], onde $P = \{a, b, c\}$, $S = \{s_0, s_1, s_2\}$, $S_0 = \{s_0\}$, $R = \{(s_0, s_1), (s_0, s_2), (s_1, s_0), (s_1, s_2), (s_2, s_2)\}$, $L(s_0) = \{a, b\}$, $L(s_1) = \{b, c\}$ e $L(s_2) = \{c\}$.

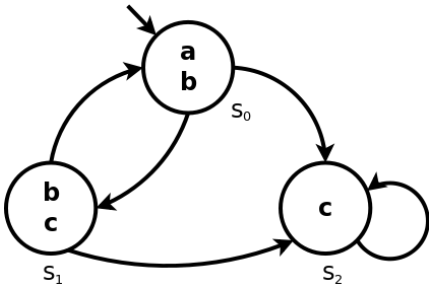

Figura 2.2: *Exemplo de estrutura de Kripke*

Modelos de fórmulas CTL são representados por estruturas de Kripke. Deste ponto em diante, sempre que no texto nos referirmos a um modelo CTL estaremos nos referindo a um modelo representado por uma estrutura de Kripke.

Seja $M = (S, S_0, R, L)$ um modelo CTL, dizemos que $\pi = [s_0, s_1, ..., s_{i-1}, s_i, s_{i+1}, ...]$ é um caminho em M se e somente se para todo $i \geq 0$, $(s_i, s_{i+1}) \in R$. Um estado s pertence a π, denotado por $s \in \pi$, se e somente se $s = s_i$ para algum $i \geq 0$. Se $s = s_0$, dizemos que o caminho π inicia no estado s. Se $\pi = [s_0, ..., s_i, ..., s_j, ...]$ e $i \leq j$, dizemos que $s_i \leqslant s_j$. Se $s_i \leqslant s_j$, mas $s_j \nleqslant s_i$, dizemos que $s_i < s_j$.

Definição 2.3 (Semântica)**.** Seja $M = (S, S_0, R, L)$ um modelo CTL, $s \in S$ um estado de M e ϕ uma fórmula CTL. A relação $(M, s) \vDash \phi$ é valida se e somente se ϕ é verdadeira no estado s de M. Definindo $\vDash$ indutivamente sobre a estrutura de ϕ, temos:

1. $(M, s) \vDash \top$.
2. $(M, s) \nvDash \bot$.
3. $(M, s) \vDash p$ sse $p \in L(s)$.
4. $(M, s) \vDash \neg\phi$ sse $(M, s) \nvDash \phi$.
5. $(M, s) \vDash \phi_1 \vee \phi_2$ sse $(M, s) \vDash \phi_1$ ou $(M, s) \vDash \phi_2$.
6. $(M, s) \vDash \phi_1 \wedge \phi_2$ sse $(M, s) \vDash \phi_1$ e $(M, s) \vDash \phi_2$.
7. $(M, s) \vDash \mathrm{EX}\,\phi$ sse existe um estado $s' \in S$ tal que $(s, s') \in R$ e $(M, s') \vDash \phi$.
8. $(M, s) \vDash \mathrm{AX}\,\phi$ sse para todo estado $s' \in S$ tal que $(s, s') \in R$, temos $(M, s') \vDash \phi$.
9. $(M, s) \vDash \mathrm{EF}\,\phi$ sse existe um caminho $\pi = [s_0, s_1, ...]$ em M tal que $s_0 = s$ e $(M, s_i) \vDash \phi$, para algum $i \geq 0$.
10. $(M, s) \vDash \mathrm{AF}\,\phi$ sse para todo caminho $\pi = [s_0, s_1, ...]$ em M tal que $s_0 = s$, temos $(M, s_i) \vDash \phi$, para algum $i \geq 0$.
11. $(M, s) \vDash \mathrm{EG}\,\phi$ sse existe um caminho $\pi = [s_0, s_1, ...]$ em M tal que $s_0 = s$ e $(M, s_i) \vDash \phi$, para todo $i \geq 0$.
12. $(M, s) \vDash \mathrm{AG}\,\phi$ sse para todo caminho $\pi = [s_0, s_1, ...]$ em M tal que $s_0 = s$, temos $(M, s_i) \vDash \phi$, para todo $i \geq 0$.
13. $(M, s) \vDash \mathrm{E}[\,\phi_1\,\mathrm{U}\,\phi_2\,]$ sse existe um caminho $\pi = [s_0, s_1, ...]$ em M tal que $s_0 = s$, existe $i \geq 0$, tal que $(M, s_i) \vDash \phi_2$ e para todo $j < i$, temos $(M, s_j) \vDash \phi_1$.
14. $(M, s) \vDash \mathrm{A}[\,\phi_1\,\mathrm{U}\,\phi_2\,]$ sse para todo caminho $\pi = [s_0, s_1, ...]$ em M tal que $s_0 = s$, existe $i \geq 0$, tal que $(M, s_i) \vDash \phi_2$ e para todo $j < i$, temos $(M, s_j) \vDash \phi_1$.

Como ilustração, voltemos à Figura 2.2. Com base na Definição 2.3, podemos fazer as seguintes afirmações sobre o modelo descrito na figura:

1. $(M, s_0) \models \top$, por definição.
2. $(M, s_0) \models a \wedge b$, pois $a, b \in L(s_0)$.
3. $(M, s_0) \not\models a \wedge c$, pois $c \notin L(s_0)$.
4. $(M, s_0) \models \mathrm{AX}\, c$, pois $(M, s_1) \models c$ e $(M, s_2) \models c$.
5. $(M, s_0) \models \mathrm{EG}\, b$, pois existe o caminho $\pi = [s_0, s_1, s_0, s_1, ...]$ em M e $(M, s_0) \models b$ e $(M, s_1) \models b$.
6. $(M, s_2) \not\models \mathrm{EG}\, b$, pois $\pi = [s_2, s_2, ...]$ em M é o único caminho iniciando em s_2 e $(M, s_2) \not\models b$.
7. $(M, s_0) \models \mathrm{A}[\, b \,\mathrm{U}\, c\,]$, pois todo caminho iniciado em s_0 constitui de transições entre os estados s_0 e s_1, onde b é verdadeiro, até que ocorre uma transição para s_2, onde b não é verdadeiro, mas c sim.

Semelhante ao que ocorre entre os operadores clássicos $\neg$, $\wedge$, $\vee$ e $\rightarrow$, existe uma certa redundância entre os operadores temporais da lógica CTL. Isso nos permite escolher um pequeno conjunto de operadores, por exemplo EX, AF e EU, e escrever os demais em função destes:

$$\begin{aligned}
\mathrm{AX}\,\phi &= \neg\mathrm{EX}\,\neg\phi \\
\mathrm{EF}\,\phi &= \mathrm{E}[\,\top\,\mathrm{U}\,\phi\,] \\
\mathrm{AG}\,\phi &= \neg\mathrm{EF}\,\neg\phi \\
\mathrm{EG}\,\phi &= \neg\mathrm{AF}\,\neg\phi \\
\mathrm{A}[\,\phi_1\,\mathrm{U}\,\phi_2\,] &= \neg(\mathrm{E}[\,\neg\phi_2\,\mathrm{U}\,(\neg\phi_1 \wedge \neg\phi_2)\,] \vee \mathrm{EG}\,\neg\phi_2)
\end{aligned}$$

Um estrutura de Kripke $M = (S, S_0, R, L)$ satisfaz uma fórmula ϕ se e somente se ϕ é satisfeita em algum dos seus estados iniciais.

$$M \models \phi \text{ sse } (M, s) \models \phi \text{ e } s \in S_0$$

É comum encontrarmos a estrutura de Kripke representada por uma tripla $M = (S, R, L)$, onde não há especificação do conjunto de estados iniciais. Dado $M = (S, S_0, R, L)$, podemos construir $M' = (S', R', L')$ equivalente a M por meio da adição de um estado especial s^*, tal que $S' = S \cup \{s^*\}$, $R' = R \cup \{(s^*, s) | s \in S_0\}$ e $L'(s) = L(s)$, $s \in S$. Assim,

$$M \models \phi \text{ sse } (M', s^*) \models \mathrm{EX}\,\phi$$

2.2 Verificação de Modelos CTL

Métodos de verificação formal permitem que sistemas, descritos por uma linguagem formal, sejam validados frente a um conjunto de propriedades. A verificação de modelos [CGP99, HR04] é talvez o método de verificação mais amplamente utilizado.

A verificação de um sistema passa por três etapas básicas: modelagem, especificação e verificação [CGP99]. A **modelagem** é a tarefa de traduzir o sistema para uma linguagem que possa ser interpretada pelo verificador. A linguagem pode variar entre diferentes implementações, porém características comuns prevalecem, como precisão e não ambiguidade. A **especificação** consiste em estabelecer as propriedades que o modelo deve satisfazer, seguindo algum formalismo lógico. A verificação de modelos utiliza como base lógicas do tipo temporal [HR04], eficazes em descrever propriedades de sistemas computacionais. Por último, a **verificação** consiste em executar o algoritmo verificador, que indica se o modelo é valido (ou não) de acordo com a especificação. Quando negativo, a maioria dos verificadores de modelos adiconalmente retorna a indicação do caminho contraexemplo.

Formalmente, verificar um modelo consiste em determinar quando $M, s \models \phi$ é válido, onde ϕ é uma fórmula em lógica temporal, M um modelo de estados finito, s um estado do modelo e $\models$ uma relação de satisfação [HR04].

Neste trabalho trataremos de apenas uma da vertentes da verificação de modelos, a *verificação de modelos CTL* . Assumiremos que o modelo M descreve um sistema segundo uma estrutura de Kripke e que a fórmula ϕ está especificada segundo a lógica temporal CTL.

A Seção 2.2.1 apresenta o algoritmo de verificação por rotulação, como descrito em [HR04]. Este é um algoritmo básico de verificação de modelos, alternativas que propiciam melhoria de performance também podem ser vistas nesta referência.

2.2.1 Algoritmo de Verificação por Rotulação

O algoritmo de *verificação de modelos por rotulação* recebe como entrada um modelo M (estrutura de Kripke) e uma fórmula CTL ϕ. Sua saída é um conjunto de estados onde ϕ é satisfeita. Como definimos anteriormente, a verificação de um modelo frente uma fórmula ϕ consiste em verificar a validade de ϕ para algum de seus estados iniciais. Obtendo como saída um conjunto de estados onde ϕ é satisfeita, o problema se resume a uma busca dos estados iniciais do modelo no resultado do algoritmo.

Um fator importante é que o algoritmo não precisa tratar todos os tipos de fórmula CTL. Como visto na Seção 2.1, algumas fórmulas podem ser escritas em função de outras. Dessa forma, o primeiro passo da execução do algoritmo é aplicar uma tradução dos operadores temporais para resumi-los à um conjunto mínimo, neste caso EX, AF e EU.

O próximo passo consiste em rotular todos os estados de M com o conjunto de subfórmulas de ϕ que estes satisfazem, iniciando das subfórmulas atômicas de ϕ até a própria fórmula ϕ. Seja ψ uma subfórmula de ϕ, e suponha que todos os estados já tenham sido adequadamente rotulados pelas subfórmulas de ψ, rotular os estados s de M com ψ vem da seguinte análise de ψ:

- Se $\psi \equiv p$, então rotule s com p se $p \in L(s)$;
- Se $\psi \equiv \psi_1 \wedge \psi_2$, então rotule s com $\psi_1 \wedge \psi_2$ se s já foi rotulado com ψ_1 e com ψ_2;
- Se $\psi \equiv EX\psi_1$, então rotule s com $EX\psi_1$ se pelo menos um de seus sucessores estiver rotulado com ψ_1;
- Se $\psi \equiv AF\psi_1$:
  - Se algum s está rotulado com ψ_1, então rotule-o com $AF\psi_1$;
  - Rotule com $AF\psi_1$ todos os estados s em que todos sucessores estejam rotulados com $AF\psi_1$ (Ver Figura 2.3).
- Se $\psi \equiv E(\psi_1 U \psi_2)$:
  - Se algum s está rotulado com ψ_2, então rotule-o com $E(\psi_1 U \psi_2)$;
  - Rotule qualquer estado s com $E(\psi_1 U \psi_2)$ se este estiver rotulado ψ_1 e ao menos um sucessor estiver rotulado com $E(\psi_1 U \psi_2)$ (Ver Figura 2.3).

Após rotular todo o modelo com as subfórmulas de ϕ (inclusive a própria fórmula), o passo final é devolver todos os estados rotulados com ϕ. Note que este algoritmo, diferente de outros algoritmos de verificação de modelos, não retorna a indicação de caminho contraexemplo.

A complexidade do algoritmo é $O(f \cdot |S| \cdot (|S| + |R|))$, onde f é o número de conectivos da fórmula. O algoritmo é linear no tamanho da fórmula e quadrático no tamanho do modelo.

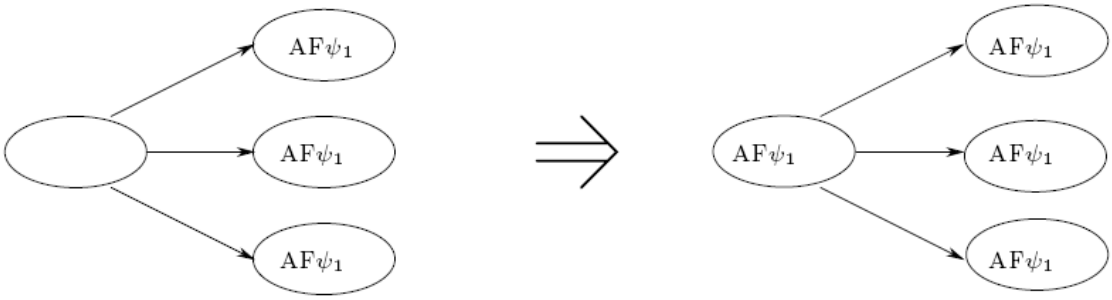

Figura 2.3: *Rotulação de estados com a subfórmula $AF\psi_1$*

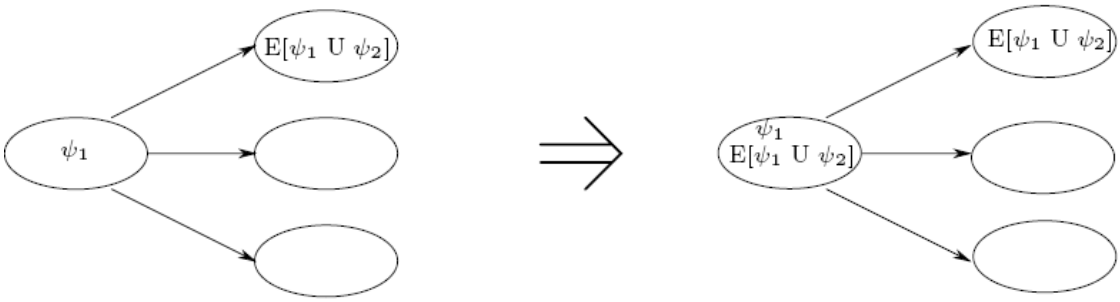

Figura 2.4: *Rotulação de estados com a subfórmula $E(\psi_1 U \psi_2)$*

Capítulo 3

Revisão de Crenças

A teoria de revisão de crenças [AGM85, Gär88] está relacionada à maneira como um agente inteligente deve agir frente a necessidade de incorporar novas informações à suas crenças. Algumas vezes essas informações conflitam com o que o agente acredita e nestas situações, técnicas de revisão de crenças indicam como ele deve incorporar tais informações preservando a consistência de suas crenças.

Neste capítulo exibimos fundamentos da teoria de revisão de crenças. Na Seção 3.1 introduzimos o conceito de estado epistêmico (crença), suas representações e possíveis operações. Na Seção 3.2 apresentamos a Teoria AGM, seus princípios e postulados. Na Seção 3.3 vemos alguns exemplos de construções de operadores de revisão. Por fim, na Seção 3.4 mostramos uma reformulação da Teoria AGM para o caso em que as crenças são representadas por bases finitas.

3.1 Estados Epistêmicos: Representação e Operadores

Há várias maneiras de representar um estado epistêmico, a mais simples e direta consiste em representar crenças como um conjunto de fórmulas logicamente fechado, ou seja, se K é um conjunto de fórmulas e α uma consequência lógica de K, denotado por $\alpha \in Cn(K)$, então $\alpha \in K$ e, portanto, $K = Cn(K)$. Denominamos estes conjuntos de **conjuntos de crenças**.

A representação através de conjuntos de crença é computacionalmente ineficiente, pois os conjuntos podem assumir tamanhos infinitos. Para contornar esse problema, podemos representar um conjunto de crenças através do que chamamos de **base de crenças**. Essa representação parte do princípio que um conjunto de crenças pode ser inferido a partir de um conjunto pequeno de crenças. Uma base de crenças B_K representa um conjunto K se e somente se $Cn(B_K) = Cn(K)$.

Outra maneira de lidar com estados epistêmicos é por meio da representação de **mundos possíveis**. Denotamos por $Mod(\alpha)$ o conjunto de mundos (modelos) onde a sentença α é verdadeira. Desta forma, um conjunto de crenças K é representado pelo conjunto de mundos $W_K = Mod(K)$, onde $Mod(K)$ denota os mundos onde todas as sentenças de K são verdadeiras. Essa representação é comumente utilizada por teóricos, mas vem ganhando utilização prática em sistemas que trabalham com conceitos epistêmicos.

Quando o conjunto de crenças K é consistente, podem ser três as atitudes epistêmicas em relação a uma crença α [Gär88]:

i. Se $\alpha \in K$: α é aceita.

ii. Se $\neg\alpha \in K$: α é rejeitada.

iii. Se $\alpha \notin K$ e $\neg\alpha \notin K$: α é indeterminada.

Mudanças em conjuntos de crenças consistem basicamente em mudar de uma atitude epistêmica para outra, em relação a uma dada fórmula. Estas mudanças se dividem em três tipos [Gär88]: o primeiro tipo ocorre quando mudamos de uma atitude "α é indeterminado" para "α é aceito" ou "α é rejeitado" (isto é, "$\neg\alpha$ é aceito"), denominamos essa mudança de **expansão**, pois consiste

em adicionar uma crença sobre a qual não tínhamos informação nenhuma; o segundo tipo refere-se a mudança de "α é aceito" ou "α é rejeitado" para "α é indeterminado", o que é denominada **contração**, já que consiste em desistir de uma crença (α ou $\neg\alpha$); o último tipo ocorre quando mudamos de "α é aceito" para "α é rejeitado" (e vice-versa), chamamos isso de **revisão**, visto que uma crença, cuja negação era anteriormente aceita, passa a ser aceita. Formalizamos, em função desses três tipos de mudança, as três operações fundamentais sobre conjuntos de crenças: Expansão ($K + \alpha$), Contração ($K - \alpha$) e Revisão ($K * \alpha$).

3.2 Teoria AGM

A Teoria AGM possui esse nome devido as iniciais dos sobrenomes de Carlos E. Alchourrón, Peter Gärdenfors e David Makinson. Estes autores publicaram o trabalho [AGM85], que veio a se tornar um pilar para grande parte da literatura relacionada à crenças dinâmicas. Em seu trabalho, os autores propuseram um conjunto de postulados de racionalidade que um processo de revisão deve obedecer, além de uma série de construções que seguem estes postulados.

Grande parte da literatura assume que a lógica subjacente ao problema de revisão de crenças satisfaz certas propriedades, as chamadas *suposições AGM*. As suposições AGM sobre essa lógica subjacente são: possuir uma linguagem $\mathcal{L}$ fechada sobre os conectivos lógicos clássicos ($\neg$, $\wedge$, $\vee$ e $\rightarrow$); e possuir um operador de consequência Cn supra-clássico, tarskiano, compacto e que satisfaz o teorema da dedução (ver Apêndice A para descrição dessas propriedades).

A lógica proposicional clássica satisfaz todas as suposições AGM. Para efeitos de simplificação, assumiremos neste capítulo esta lógica como lógica subjacente ao problema de revisão de crenças.

3.2.1 Postulados de Racionalidade

A Teoria AGM define um conjunto de postulados de racionalidade, que guiam as operações sob o princípio da mudança mínima. Em síntese, mudar minimamente um conjunto de crenças significa que nenhuma crença deve ser adicionada ou removida se isso não for estritamente necessário ao sucesso da operação. Exceto pela operação de expansão, que é diretamente definida por $K+\alpha = Cn(K \cup \alpha)$, as demais operações possuem um conjunto próprio de postulados de racionalidade.

Postulados AGM para Contração

Dado um conjunto de crenças K e uma crença α, os seis postulados básicos para a operação de contração são [Gär88]:

(K-1) $K - \alpha$ é um conjunto de crenças (*fecho*)

(K-2) $K - \alpha \subseteq K$ (*inclusão*)

(K-3) Se $\alpha \notin K$, então $K - \alpha = K$ (*vacuidade*)

(K-4) Se não $\vdash \alpha$, então $\alpha \notin K - \alpha$ (*sucesso*)

(K-5) Se $\alpha \in K$, então $K \subseteq (K - \alpha) + \alpha$ (*recuperação*)

(K-6) Se $\vdash \alpha \leftrightarrow \beta$, então $K - \alpha = K - \beta$ (*equivalência*)

O postulado **(K-1)** afirma que o resultado da contração de um conjunto de crenças é ainda um conjunto de crenças. O postulado **(K-2)** assegura que nenhuma nova fórmula é adicionada em uma operação de contração. Já o postulado **(K-3)** afirma que o conjunto de crenças permanece inalterado quando é contraído por uma crença que não possui. O postulado **(K-4)** assegura o sucesso da contração. O penúltimo postulado, **(K-5)**, indica que o conjunto de crenças original pode ser recuperado através da expansão pela crença a qual foi anteriormente contraído. Por fim, o

postulado **(K-6)** certifica que duas crenças equivalentes, quando contraídas em um mesmo conjunto de crenças, geram resultados equivalentes.

Além dos seis postulados básicos, dois postulados adicionais tratam da operação de contração quando aplicada a conjunções de fórmulas:

(K-7) $K - \alpha \cap K - \beta \subseteq K - (\alpha \wedge \beta)$

(K-8) Se $\alpha \notin K - (\alpha \wedge \beta)$, então $K - (\alpha \wedge \beta) \subseteq K - \alpha$.

O postulado **(K-7)** afirma que crenças que estejam tanto em $K - \alpha$ quanto em $K - \beta$ devem pertencer a contração de K por $(\alpha \wedge \beta)$. O postulado **(K-8)** diz que se removemos α quando contraímos K por $(\alpha \wedge \beta)$, então qualquer crença que removemos de K quando fazemos $K - \alpha$ também deve ser removida quando o contraímos por $(\alpha \wedge \beta)$.

Postulados AGM para Revisão

Da mesma forma, dados K e α, os seis postulados básicos para operação de revisão são [Gär88]:

(K*1) $K * \alpha$ é um conjunto de crenças (*fecho*)

(K*2) $\alpha \in K * \alpha$ (*sucesso*)

(K*3) $K * \alpha \subseteq K + \alpha$ (*inclusão*)

(K*4) Se $\neg\alpha \notin K$, então $K + \alpha \subseteq K * \alpha$ (*preservação*)

(K*5) $K * \alpha = \mathcal{L}$ se e somente se $\vdash \neg\alpha$ (*consistência*)

(K*6) Se $\vdash \alpha \leftrightarrow \beta$, então $K * \alpha = K * \beta$ (*equivalência*)

O postulado **(K*1)** declara que a revisão de um conjunto de crenças resulta em um conjunto de crenças. O segundo postulado, **(K*2)**, assegura o sucesso da operação de revisão. O postulado **(K*3)** garante que nenhuma informação adicional é acrescida ao conjunto revisado. O postulado **(K*4)** (unido ao postulado (K*3)), afirma que quando a nova informação não é inconsistente com o conjunto de crenças, a revisão é equivalente a uma expansão. O postulado **(K*5)** diz que uma revisão gera um conjunto inconsistente apenas quando a crença é inconsistente. Finalmente, o postulado **(K*6)** certifica que duas crenças equivalentes resultam em conjuntos revisados equivalentes.

Assim como ocorre para contração, existem dois postulados adicionais aos seis básicos:

(K*7) $K * (\alpha \wedge \beta) \subseteq (K * \alpha) + \beta$.

(K*8) Se $\neg\beta \notin K * \alpha$, então $(K * \alpha) + \beta \subseteq K * (\alpha \wedge \beta)$

Os postulados **(K*7)** e **(K*8)**, unidos, afirmam que as interpretações que satisfazem $K * (\alpha \wedge \beta)$ são idênticas às interpretações que satisfazem simultaneamente $K * \alpha$ e β [Gär88].

3.2.2 Relações de Identidade

Contração e revisão podem ser definidos um em função do outro por meio de duas identidades: *identidade de Levi* e *identidade de Harper* [Gär88]. Revisar um conjunto de crenças K com uma fórmula α equivale a contrair K com α e então adicionar α ao resultado:

$$K * \alpha = (K - \neg\alpha) + \alpha \qquad \textit{(identidade de Levi)}$$

Contrair um conjunto de crenças K com uma fórmula α equivale a revisar K com $\neg\alpha$ e então buscar os elementos comuns entre K e o resultado da revisão:

$$K - \alpha = (K * \neg\alpha) \cap K \qquad \textit{(identidade de Harper)}$$

3.3 Construções

Os autores de [AGM85] propuseram construções para funções de contração/revisão capazes de satisfazer todos os postulados propostos. Após eles, outros autores propuseram suas próprias construções. Nesta seção apresentamos uma síntese sobre duas importantes construções: *partial meet* e *sistema de esferas*.

3.3.1 Partial Meet

Uma construção do tipo *partial meet* faz uso do conceito de *conjunto resíduo* para realizar a operação de contração. Um conjunto resíduo $K \perp \alpha$ consiste nos subconjuntos máximos de K que não implicam α. Formalmente uma contração do tipo *partial meet* é definida como segue.

Definição 3.1. Seja K um conjunto de crenças e α uma nova crença, uma *função de contração partial meet* sobre K é dada por

$$K - \alpha = \bigcap \gamma(K \perp \alpha)$$

onde γ é uma função de seleção que satisfaz

i. $\gamma(K \perp \alpha) \subseteq K \perp \alpha$

ii. Se $K \perp \alpha \neq \emptyset$ então $\gamma(K \perp \alpha) \neq \emptyset$, caso contrário $\gamma(K \perp \alpha) = \{K\}$

Teorema 1 ([Gär88]). *Seja $-$ uma função de contração. Para todo conjunto de crenças K, $-$ é uma função de contração* partial meet *se e somente se $-$ satisfaz os postulados (K-1)-(K-6) para contração.*

Quando uma função de seleção γ realiza a escolha dos melhores conjuntos resíduos baseada em alguma relação de preferência $\leqslant$, isto é, $\gamma(K \perp \alpha) = \{K' \in K \perp \alpha \mid K'' \leqslant K'$ para todo $K'' \in K \perp \alpha\}$, dizemos que γ é uma função *relacional* sobre K. Uma contração *partial meet* que utiliza uma função de seleção relacional é denominada *função de contração partial meet relacional.*

Se a função de seleção relacional utiliza uma relação transitiva sobre os conjuntos resíduos, dizemos que essa função é *relacional transitiva* sobre K. Uma contração *partial meet* que utiliza uma função de seleção relacional transitiva é denominada *função de contração partial meet relacional transitiva.*

Teorema 2 ([Gär88]). *Seja $-$ uma função de contração. Para todo conjunto de crenças K, $-$ é uma função de contração* partial meet *relacional transitiva se e somente se $-$ satisfaz os postulados (K-1)-(K-8) para contração.*

Como vimos na Seção 3.2.2, pela identidade de Levi podemos definir uma revisão em função de uma contração. Assim, podemos definir uma construção *partial meet* para revisão utilizando a construção para negação dada pela Definição 3.1.

Definição 3.2. Seja K um conjunto de crenças, α uma nova crença e $-$ uma função de contração *partial meet*, uma *função de revisão partial meet* sobre K é dada por

$$K * \alpha = Cn(\{K - \alpha\} \cup \{\alpha\})$$

Teorema 3 ([Gär88]). *Se a função de contração $-$ satisfaz (K-1)-(K-4) e (K-6), então uma função de revisão $*$ conforme a Definição 3.2 satisfaz (K*1)-(K*6). Se $-$ satisfaz (K-7), então $*$ satisfaz (K*7). Se $-$ satisfaz (K-8), então $*$ satisfaz (K*8).*

3.3.2 Sistema de Esferas

Proposto por Groove [Gro88], este tipo de construção utiliza a representação de mundo possíveis para representar os estados epistêmicos de um agente. Esta abordagem agrupa os mundos possíveis em esferas concêntricas, estabelecendo uma ordem entre estes mundos (ver Figura 3.1). Grove mostra que esta construção possui uma relação direta com a construção de revisão *partial meet* que utiliza funções de seleção relacional transitiva.

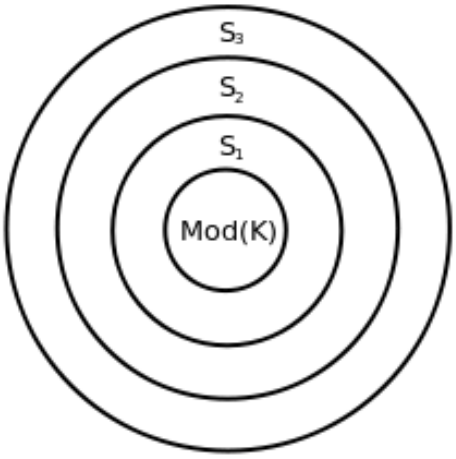

Figura 3.1: *Representação gráfica de um sistema de esferas.*

O sistemas de esferas, por não se ater a operações com conjuntos de fórmulas, representa bem a semântica do problema de revisão de crenças. Podemos ver as crenças de um agente como um conjunto de modelos que este julga como possíveis para descrever o mundo real. Ao precisar adicionar uma crença, o agente toma consciência que apenas aqueles modelos compatíveis com a nova crença são na realidade corretos. Assim, o agente adapta suas crenças escolhendo um conjunto apropriado de modelos, que preserve o máximo possível de informação de suas atuais crenças. Para tanto, o novo conjunto de modelos deve ser compatível com a nova crença e o mais próximo possível, do ponto de vista epistemológico, do atual conjunto de modelos. Esse conceito de proximidade epistemológica é aqui determinado pela forma com que os mundos são dispostos no sistema de esferas.

Seja $\mathcal{L}$ a linguagem da lógica subjacente a revisão, denominamos por *mundo possível* uma extensão maximal consistente de $\mathcal{L}$. O conjunto $\mathbf{M}$ representa todos os mundos possíveis dessa linguagem. Um conjunto de crenças K pode ser representado por um conjunto maximal $Mod(K) \subseteq \mathbf{M}$ tal que se $M \in Mod(K)$, então M é mundo possível para todas as sentenças de $\alpha \in K$. M é um mundo possível para α se $\alpha \in M$. O conjunto $Mod(\alpha)$ denota todos os mundos possíveis para α.

Definição 3.3 ([Gro88]). Um *sistema de esferas centrado em* $Mod(K)$ é uma coleção $\mathbf{S}$ de subconjuntos de $\mathbf{M}$ que satisfazem as seguintes condições:

(S1) $\mathbf{S}$ é totalmente ordenado por $\subseteq$, ou seja, se $S, S' \in \mathbf{S}$, então $S \subseteq S'$ ou $S' \subseteq S$.

(S2) $Mod(K)$ é o $\subseteq$-minimo de $\mathbf{S}$, ou seja, $Mod(K) \in \mathbf{S}$ e para todo $S \in \mathbf{S}$, $Mod(K) \subseteq S$.

(S3) $\mathbf{M} \subseteq \mathbf{S}$ (e é o maior elemento de $\mathbf{S}$).

(S4) Se α é uma sentença e alguma esfera intercepta $Mod(\alpha)$, então existe uma esfera $S_\alpha \subseteq S$ que é a menor esfera de S que intercepta $Mod(\alpha)$.

O conceito de sistema de esferas é formalizado na Definição 3.3. Baseada nesta definição, especificamos a seguir a construção para revisão proposta por Grove [Gro88].

Definição 3.4. Seja K um conjunto de crenças, α uma nova crença e $\mathbf{S}$ um sistema de esferas em $\mathbf{M}$ centrado em $Mod(K)$, uma *função de revisão no sistemas de esferas* $\mathbf{S}$ é dado por

$$K * \alpha = \delta(Mod(\alpha) \cap S_\alpha)$$

onde $S_\alpha \in \mathbf{S}$ é a menor esfera de $\mathbf{S}$ que intersecta $Mod(\alpha)$ (caso não haja intersecção, $S_\alpha = \emptyset$) e $\delta(S)$ é uma função que traduz da representação de mundos possíveis para a de conjuntos de crenças[1].

A Figura 3.2 ilustra o operador de revisão descrito na Definição 3.4. A área escura corresponde a região de modelos que compõem o resultado da revisão. Observe que mesmo $Mod(\alpha)$ interceptando diversas esferas do sistema, o operador estabelece uma relação de preferência entre os mundos intersectados, escolhendo apenas aqueles mais próximos ao centro do sistema. Caso $Mod(\alpha)$ não intersectasse o sistema de esferas, o resultado da revisão seria um conjunto vazio.

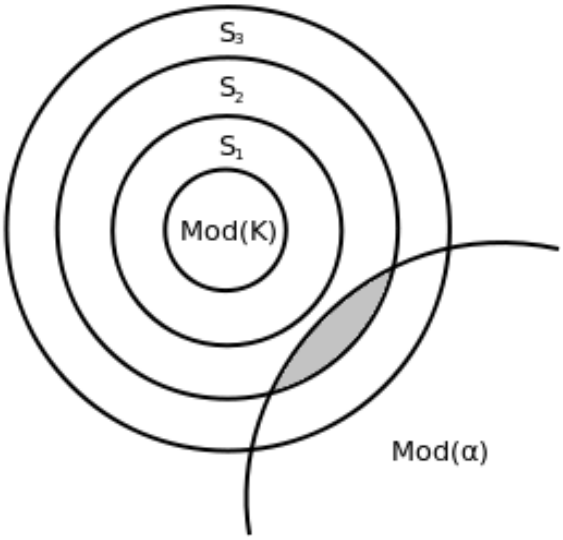

Figura 3.2: *Revisão $K * \alpha$ no sistema de esferas.*

O ponto chave da construção de Grove é a relação de preferência estabelecida pelo sistema de esfera. Os mundos resultantes da revisão de K por α correspondem aos mundos de α que mais se aproximam aos mundos de K, seguindo a ordem determinada pela disposição das esferas. Grove relaciona sua construção aos postulados AGM ao demonstrar os dois teoremas a seguir.

Teorema 4 ([Gro88]). *Seja K um conjunto de crenças e $\mathbf{S}$ um sistema de esferas em $\mathbf{M}$ centrado em $Mod(K)$. Se, para algum α, $K * \alpha$ é realizada por uma função de revisão no sistema de esferas S, então os postulados (K*1)-(K*8) são satisfeitos.*

Teorema 5 ([Gro88]). *Seja $*$ uma função de revisão que satisfaz (K*1)-(K*8). Então para qualquer conjunto de crenças (fixo) K existe uma função de revisão em um sistema de esferas $\mathbf{S}$ em $\mathbf{M}$, tal que $\mathbf{S}$ é centrado em $Mod(K)$ e equivale a $*$.*

Os teoremas 4 e 5 representam uma correspondência direta entre o sistema de esferas e a construção de funções de revisão *partial meet* relacional transitiva. Mais que isso, estes teoremas mostram que o princípio de seleção de modelos utilizado pelo sistema de esferas é característico a todo operador que satisfaça os oito postulados AGM para revisão.

[1] Em [Gro88] essa tradução é realizada por um operador $t(S) = \cap\{x \in S\}$. O autor afirma que $t(Mod(K)) = K$, desde que lógica seja compacta.

3.4 Reformulação da Teoria AGM

É computacionalmente inviável tratar o problema da revisão de crenças da forma realizada pela Teoria AGM, visto o tamanho infinito dos conjuntos de crenças. Nessa seção mostramos uma reformulação da Teoria AGM proposta por Katsuno e Mendelzon [KM89, KM91], onde crenças são representadas por uma única fórmula proposicional.

Seja $\mathcal{L}$ uma linguagem proposicional finita, representamos nossas crenças por uma única fórmula proposicional $\psi \in \mathcal{L}$, onde ψ representa a conjunção dos elementos de uma base de crenças finita K, tal que $Cn(\psi) = Cn(K)$. Com isso, obtemos uma representação fixa e finita de nossas crenças, passível de ser armazenada por um computador.

3.4.1 Reformulação dos Postulados

A restrição na representação das crenças tornam mais simples os processos de revisão e contração. Em [KM89] foi sugerida uma reformulação dos postulados AGM quando estes forem aplicados à revisões (contrações) entre uma base de crenças finita ψ e uma crença α, ambas representadas por fórmulas proposicionais.

Sejam ψ e α duas fórmulas proposicionais, denotamos por $\psi \bullet \alpha$ a contração de ψ por α. Especializando para o caso da lógica proposicional, onde ψ representa uma base de crenças finitas, os postulados (K-1)-(K-6) correspondem ao seguinte conjunto de postulados [KM91]:

(C1) $\psi \vDash \psi \bullet \alpha$.

(C2) Se $\psi \nvDash \alpha$, então $\psi \bullet \alpha \leftrightarrow \psi$.

(C3) Se não $\vDash \alpha$, então $\psi \bullet \alpha \nvDash \alpha$.

(C4) Se $\psi_1 \leftrightarrow \psi_2$ e $\alpha_1 \leftrightarrow \alpha_2$, então $\psi_1 \bullet \alpha_1 \leftrightarrow \psi_2 \bullet \alpha_2$

(C5) $(\psi \bullet \alpha) \wedge \alpha \vDash \psi$

Em [KM91] não há uma reformulação para (K-7) e (K-8). Descrevemos a seguir a reformulação que adotamos para estes postulados:

(C6) $\psi \bullet (\alpha \wedge \phi) \vDash (\psi \bullet \alpha) \wedge (\psi \bullet \phi)$.

(C7) Se $(\psi \bullet (\alpha \wedge \phi)) \wedge \neg\alpha$ é satisfazível, então $\psi \bullet \alpha \vDash \psi \bullet (\alpha \wedge \phi)$.

De forma análoga, sejam ψ e α duas fórmulas proposicionais, denotamos por $\psi \circ \alpha$ a revisão de ψ por α. Novamente, especializando para o caso da lógica proposicional, os postulados (K*1)-(K*8) correspondem ao seguinte conjunto de postulados [KM91]:

(R1) $\psi \circ \alpha \vDash \alpha$.

(R2) Se $\psi \wedge \alpha$ é satisfazível, então $\psi \circ \alpha \leftrightarrow \psi \wedge \alpha$.

(R3) Se α é satisfazível, então $\psi \circ \alpha$ é satisfazível.

(R4) Se $\psi_1 \leftrightarrow \psi_2$ e $\alpha_1 \leftrightarrow \alpha_2$, então $\psi_1 \circ \alpha_1 \leftrightarrow \psi_2 \circ \alpha_2$

(R5) $(\psi \circ \alpha) \wedge \phi \vDash \psi \circ (\alpha \wedge \phi)$

(R6) Se $(\psi \circ \alpha) \wedge \phi$ é satisfazível, então $\psi \circ (\alpha \wedge \phi) \vDash (\psi \circ \alpha) \wedge \phi$

A representação de crenças por meio de fórmulas proposicionais foi a representação adotada por Katsuno e Mendelzon [KM91] para descrever o problema de atualização de crenças (tema do próximo capítulo). Dessa forma, apresentar esta reformulação da Teoria AGM é importante para a compreensão da diferença entre os problemas de revisão e atualização de crenças, um dos pontos fundamentais do nosso trabalho.

Capítulo 4

Atualização de Crenças

A atualização crenças, assim como a revisão de crenças, visa preservar a consistência de conjunto de crenças quando novas informações são inseridas. A distinção fundamental entre esses dois tipos de mudanças está na natureza das novas informações. A revisão de crenças se dá sobre um mundo estático, onde uma nova informação indica que a atual base de crenças pode não descrever corretamente o mundo. Já a atualização de crenças atua sobre um mundo dinâmico, onde as novas informações descrevem mudanças ocorridas no mundo.

A distinção entre revisão e atualização foi primeiro realizada em [KW85] e retomada posteriormente em [KM91]. Neste último, além de definir formalmente esta distinção, Katsuno e Mendelzon mostram que os postulados AGM não são adequados para o novo contexto, formulando assim um novo conjunto de postulados de racionalidade para o processo de atualização: os *postulados KM*. Em [Her98], o autor mostra que revisão e atualização, além de distintas, são incompatíveis, pois não existe um operador capaz de satisfazer tanto os postulados AGM quanto os postulados KM.

Assumiremos aqui as mesmas restrições adotadas por [KM91] (e vistas anteriormente no Capítulo 3). Definimos a lógica proposicional clássica como lógica subjacente ao problema de atualização de crenças. Assumimos também que esta é definida sobre uma linguagem proposicional finita $\mathcal{L}$. Os estados epistêmicos são representados por bases de crenças finitas, condensadas em um única fórmula proposicional. Assumimos que a crença incorporada (removida) de ψ é também representada por uma fórmula proposicional.

Neste capítulo veremos os conceitos fundamentais de atualização de bases de crença. Na Seção 4.1 apresentamos a diferença entre revisão e atualização, conforme descrito em [KM91]. Na Seção 4.2 apresentamos um operador típico de atualização, a Abordagem de Modelos Possíveis (PMA, do inglês *Possible Models Approach*) [Win88]. Por fim, na Seção 4.3 mostramos os postulados KM e sua relação com os postulados AGM.

4.1 Diferença entre Atualização e Revisão de Crenças

Suponha que queremos adequar a base de conhecimentos ψ a uma crença α. Como vimos na Seção 3.3, um operador de revisão (que satisfaça os postulados AGM) busca essencialmente selecionar dentre os modelos que satisfazem α, aqueles que são mais próximos. Essa noção de proximidade deve ser previamente definida por uma relação de ordem entre modelos. No sistemas de esferas, por exemplo, essa relação é dada pelo encadeamento de esferas. Para modelos da lógica proposicional, essa relação é usualmente estabelecida pelo número de átomos proposicionais cujos valores verdade diferem. Os modelos selecionados comporão a nova base de crenças, denotada por $\psi \circ \alpha$. Por outro lado, um operador de atualização seleciona, para cada modelo M de ψ, os modelos de α mais próximos a M. A nova base de crenças, denotada por $\psi \diamond \alpha$, será composta pela união de todos os modelos selecionados.

Suponha que I e J são os únicos modelos para ψ e que K e L são os únicos modelos para α. Suponha também que K é mais próximo a I do que L é, que L é mais próximo a J do que K é, e que K é mais próximo de I do que L é de J. A Figura 4.1 ilustra está situação.

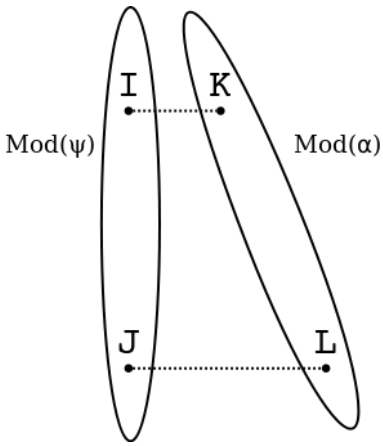

Figura 4.1: *Ilustração do conceito de proximidade entre modelos.*

Na Figura 4.1, a relação de proximidade entre modelos é determinada pela distância entre os pontos que os representam. As linhas pontilhadas indicam os pares de modelos mais próximos entre si. Suponha então que escolhemos K, mas não L, para compor a nova base de crenças porque ele é o modelo em $Mod(\alpha)$ que está ligado ao menor dos segmentos pontilhados. Isto é, escolhemos descartar L porque existe ao menos um modelo de α (o modelo K) que está mais próximo ao conjunto $Mod(\psi)$.

Sob o ponto de vista da revisão, esta escolha é racional. Para fazer essa escolha nos concentramos em I e na proximidade que K possui deste. De fato, ignoramos que J costumava ser um modelo de ψ. Como não há mudanças no mundo, sabemos que α deve ser verdade em todos os mundos possíveis, então podemos esquecer aqueles modelos de ψ que são demasiados diferentes dos que satisfazem α.

Por outro lado, esse raciocínio não é adequado quando fazemos atualização. A informação α descreve o que mudou no mundo real. Os modelos de ψ são os mundos possíveis e sabemos que o mundo real está dentre eles. Mas como não sabemos qual mundo é o real, devemos tratar todos eles com igual ponderação. Nós examinamos cada um dos mundos possíveis e vemos a melhor maneira de mudá-lo, de modo que se tornem um modelo para α. Diferente do que ocorre na revisão, a nova informação não nos permite inferir que alguns dos mundos que tínhamos na verdade não eram possíveis.

O Exemplo 4.1 é baseado no exemplo utilizado por [KM91] e ilustra a distinção entre revisão e atualização de crenças.

Exemplo 4.1. *Suponha que em um quarto existem três objetos: um livro, uma revista e uma mesa. A proposição b significa "o livro está sobre a mesa" enquanto que m significa "a revista está sobre a mesa". Nós acreditamos que* $\psi \leftrightarrow (b \wedge \neg m) \vee (\neg b \wedge m)$*, isto é, ou o livro ou a revista está sobre a mesa, mas não ambos. Assim,* ψ *possui apenas dois modelos: I, b é verdade e m falso; e J, b é falso e m é verdade.*

Suponha agora que nós enviamos um robô para pôr o livro sobre a mesa. Após a operação ser realizada, nós desejamos incorporar em nossa base de crenças a nova informação $\phi \leftrightarrow b$*. Esta nova crença também possui dois únicos modelos: o próprio modelo I e um novo L, onde ambos, livro e revista, estão sobre a mesa. Iremos selecionar dentre os modelos de* ϕ *aqueles mais próximos aos modelos de* ψ*. Nós assumiremos como relação de distância o número de proposições com diferentes valores verdade. O modelo I está, claro, a uma distância 0 dele mesmo e a uma distância 2 de J (ambas proposições diferentes). Por outro lado, o modelo L está a uma distância 1 de ambos I e J (exatamente uma posição de objeto está correta).*

O operador de revisão irá selecionar apenas o modelo I porque este é o modelo mais próximo (distância 0) do conjunto original. Mas intuitivamente isso não nos parece correto: após o robô executar sua ação, todo que sabemos é que o livro estará sobre a mesa, mas continuamos sem saber a posição da revista.

4.2 Abordagem de Modelos Possíveis

A abordagem de modelos possíveis (PMA) [Win88, Win90] é um exemplo típico de método de atualização de crenças. É interessante ressaltar que o trabalho de Winslett precede o trabalho de Katsuno e Mendelzon [KM91], onde está formalizado o conceito de atualização de crenças. A abordagem de modelos possíveis é dada pela Definição 4.1.

Definição 4.1 ([KM91]). Seja ψ a base de crenças e α a nova crença, o operador PMA, simbolizado por $\diamond_{pma}$, é formalmente definido como

$$Mod(\psi \diamond_{pma} \alpha) = \bigcup_{I \in Mod(\psi)} Minimais(Mod(\alpha), I).$$

onde $Mod(\phi)$ denota o conjunto de modelos da fórmula ϕ e a expressão $Minimais(Mod(\alpha), I)$, o subconjunto de modelos de α que mais se aproximam de I.

No PMA, a relação de proximidade entre dois modelos, I_1 e I_2, e uma interpretação I é definida por uma ordenação parcial $\leqslant_I$, tal que $I_1 \leqslant_I I_2$ se e somente se $diff(I, I_1) \subseteq diff(I, I_2)$, onde $diff(A, B)$ representa o conjunto de proposições cujos valores verdade diferem em A e B. Portanto, $J \in Minimais(Mod(\alpha), I)$ se e somente se J é minimal em $Mod(\alpha)$ com relação a $\leqslant_I$, isto é, não existe $J' \in Mod(\alpha)$ tal que $J \nleqslant_I J'$.

Exemplo 4.2. *Voltemos a situação do Exemplo 4.1. Existem dois modelos possíveis para a fórmula ψ: $I_1 = \langle T, F\rangle$ e $I_2 = \langle F, T\rangle$. Do mesmo modo, para a fórmula α temos: $J_1 = \langle T, T\rangle$ e $J_2 = \langle T, F\rangle$. Assim, temos que $J_2 \leqslant_{I_1} J_1$ e $J_1 \leqslant_{I_2} J_2$, pois $Diff(I_1, J_2) = \{\} \subseteq \{m\} = Diff(I_1, J_1)$ e $Diff(I_2, J_1) = \{b\} \subseteq \{b, m\} = Diff(I_2, J_2)$. Deste modo,*

$$\begin{aligned} Mod(\psi \diamond_{pma} \alpha) &= \bigcup_{I \in Mod(\psi)} Minimais(Mod(\alpha), I) \\ &= Minimais(\{J_1, J_2\}, I_1) \cup Minimais(\{J_1, J_2\}, I_2) \\ &= \{J_2\} \cup \{J_1\} \\ &= \{\langle T, T\rangle, \langle T, F\rangle\} \end{aligned}$$

resultando em $\psi \diamond_{pma} \alpha \leftrightarrow b$.

O Exemplo 4.2 ajuda a compreender como a atualização é mais adequada que revisão em determinadas situações. Depois que a ação *por o livro sobre a mesa* é executada, tudo que nós sabemos é que o livro está sobre a mesa, nada sabemos sobre a revista. E é exatamente essa informação que obtemos ao aplicar o PMA.

4.3 Postulados de Racionalidade

Katsuno e Mendelzon [KM91] definem um conjunto de postulados de racionalidade para a construção de operações de atualização. Seja ψ uma base de crenças representada por uma fórmula proposicional e α uma fórmula que representa a nova crença, Katsuno e Mendelzon mostraram que os postulados KM caracterizam os operadores de atualização que selecionam para cada modelo M de ψ os modelos de α mais próximos de M.

4.3.1 Postulados KM para Atualização

Dada uma base de crenças ψ e uma crença α, os oito postulados para atualização são [KM91]:

(U1) $\psi \diamond \alpha \vDash \alpha$.

(U2) Se $\psi \vDash \alpha$, então $\psi \diamond \alpha \leftrightarrow \psi$.

(U3) Se ψ e α são satisfazíveis, então $\psi \diamond \alpha$ é satisfazível.

(U4) Se $\psi_1 \leftrightarrow \psi_2$ e $\alpha_1 \leftrightarrow \alpha_2$, então $\psi_1 \diamond \alpha_1 \leftrightarrow \psi_2 \diamond \alpha_2$

(U5) $(\psi \diamond \alpha) \wedge \beta \vDash \psi \diamond (\alpha \wedge \beta)$

(U6) Se $\psi \diamond \alpha_1 \vDash \alpha_2$ e $\psi \diamond \alpha_2 \vDash \alpha_1$, então $\psi \diamond \alpha_1 \leftrightarrow \psi \diamond \alpha_2$

(U7) Se $\psi \vDash \alpha$ ou $\psi \vDash \neg\alpha$, para todo α, então $(\psi \diamond \alpha_1) \wedge (\psi \diamond \alpha_2) \vDash \psi \diamond (\alpha_1 \vee \alpha_2)$

(U8) $(\psi_1 \vee \psi_2) \diamond \alpha \leftrightarrow (\psi_1 \diamond \alpha) \vee (\psi_2 \diamond \alpha)$

O postulado **(U1)** assegura o sucesso da atualização. O segundo postulado, **(U2)**, afirma que se a sentença α é derivável da base ψ, a operação de atualização deixará esta inalterada. O postulado **(U3)** garante que quando a base de crenças e a nova crença são satisfazíveis, o resultado da atualização é também satisfazível. O postulado **(U4)** assegura a operação de atualização como insensível a sintaxe, onde resultados equivalentes são obtidos quando duas crenças ou dois conjuntos de crenças são equivalentes. O postulado **(U5)** declara que o resultado de atualizar ψ por $\alpha \wedge \beta$ esta contido na expansão por β da atualização de ψ por α. O postulado **(U6)** diz que se uma base de crenças atualizada por α_2 deriva α_1 e essa mesma base atualizada por α_1 deriva α_2, então ambas atualizações produzem o mesmo efeito. O penúltimo postulado, **(U7)**, afirma que um modelo satisfaz a atualização por $\alpha_1 \vee \alpha_2$, sempre que satisfizer a atualização por α_1 e por α_2. Finalmente, o postulado **(U8)** afirma que um modelo satisfaz a atualização de $\psi_1 \vee \psi_2$ por α se e somente ele satisfaz atualização de pelo menos uma das fórmulas separadamente.

4.3.2 Postulados KM para Remoção

Katsuno e Mendelzon também descreveram uma operação análoga à contração, a qual denominaram *remoção*. Do mesmo modo, dado ψ e α, os seis postulados para a operação de remoção são:

(E1) $\psi \vDash \psi \blacklozenge \alpha$.

(E2) Se $\psi \vDash \neg\alpha$, então $\psi \blacklozenge \alpha \leftrightarrow \psi$.

(E3) Se ψ é satisfazível e não $\vDash \alpha$, então $\psi \blacklozenge \alpha \nvDash \alpha$.

(E4) Se $\psi_1 \leftrightarrow \psi_2$ e $\alpha_1 \leftrightarrow \alpha_2$, então $\psi_1 \blacklozenge \alpha_1 \leftrightarrow \psi_2 \blacklozenge \alpha_2$

(E5) $(\psi \blacklozenge \alpha) \wedge \alpha \vDash \psi$

(E8) $(\psi_1 \vee \psi_2) \blacklozenge \alpha \leftrightarrow (\psi_1 \blacklozenge \alpha) \vee (\psi_2 \blacklozenge \alpha)$

O postulado **(E1)** assegura que nenhuma nova fórmula foi adicionada pela remoção. O segundo postulado, **(E2)**, afirma que a base de crenças ficará inalterada quando essa já derivar a negação do que se quer contrair. O postulado **(E3)** garante o sucesso da remoção. O quarto postulado, **(E4)**, assegura a insensibilidade a sintaxe. O postulado **(E5)** diz que a base de crenças original pode ser recuperada pela inclusão da crença removida. Por fim, o postulado **(E8)** afirma que um modelo satisfaz a remoção de α em $\psi_1 \vee \psi_2$ se e somente ele satisfaz a remoção em uma das fórmulas separadamente.

4.3.3 Relações entre Atualização e Remoção

Similar ao que ocorre entre revisão e contração, podemos estabelecer uma relação entre funções de atualização e de remoção. O teorema a seguir é demonstrado por Katsuno e Mendelzon [KM91] e suas identidades se assemelham àquelas apresentadas na Seção 3.2.2.

Teorema 6 ([KM91]). *Sejam ψ e α fórmulas proposicionais denotando, respectivamente, a base de crenças finita e a nova crença.*

1. *Se $\diamond$ é uma função de atualização que satisfaz os postulados (U1)-(U4) e (U8), uma função de remoção $\blacklozenge$ definida como*
$$\psi \blacklozenge \phi \leftrightarrow \psi \vee (\psi \diamond \neg\alpha)$$
satisfaz os postulados (E1)-(E5) e (E8).

2. *Se $\blacklozenge$ é uma função de remoção que satisfaz os postulados (E1)-(E4) e (E8), uma função de atualização $\diamond$ definida como*
$$\psi \diamond \phi \leftrightarrow (\psi \blacklozenge \neg\alpha) \wedge \alpha$$
satisfaz os postulados (U1)-(U4) e (U8).

4.4 Construções

Nesta seção descrevemos uma classe de operadores de atualização que são completamente caracterizados pelos postulados KM. Estes operadores utilizam relações de pré-ordem parcial para estabelecer o conceito de proximidade entre modelos. A Definição 4.2 fornece uma descrição formal de uma construção para esta classe de operadores (essa formulação é parte do Teorema 3.4 apresentado em [KM91]).

Definição 4.2 ([KM91]). Seja $\psi\diamond\alpha$ o resultado da atualização de ψ por α. Se existe um mapeamento de cada modelo $M \in Mod(\psi)$ para uma pré-ordem parcial $\leqslant_M$, tal que para qualquer $M' \neq M$, $M \leqslant_M M'$ e $M' \nleqslant_M M$, então uma função de atualização $\diamond$ pode ser definida por

$$Mod(\psi \diamond \alpha) = \bigcup_{M \in Mod(\psi)} Min(Mod(\alpha), \leqslant_M)$$

onde $Min(A, \leqslant)$ representa os elementos de A minimais segundo a ordem $\leqslant$.

Katsuno e Mendelzon [KM91] mostram que o PMA é uma instância particular da classe acima, onde a relação de pré-ordem parcial é definida pela diferença dos valores verdade entre pares de elementos do conjunto de modelos.

Por fim, o Teorema 7 mostra que o princípio de seleção utilizado pela classe de operadores da Definição 4.2 é completamente caracterizada pelos postulados (U1)-(U8).

Teorema 7 ([KM91]). *Um operador de atualização $\diamond$ pode ser construído pela Definição 4.2 se e somente se $\diamond$ satisfaz os postulados (U1)-(U8).*[1]

[1] O Teorema 7 é uma reformulação do Teorema 3.4 demonstrado em [KM91].

Capítulo 5

Atualização de Modelos

A atualização de modelos CTL é uma abordagem que estende os conceitos clássicos de atualização de crenças à lógica CTL. Essa abordagem é apresentada por Zhang e Ding em [ZD08], onde, dentre outras contribuições, formalizam conceitos básico para a atualização, como o princípio da mudança mínima, e analisam propriedades semânticas e computacionais desta abordagem.

A Seção 5.1 mostra a definição do princípio da mudança mínima para atualização de modelos. Na Seção 5.2 vemos a definição do operador de atualização de modelos CTL e algumas de suas propriedades semânticas. E por fim, na Seção 5.3 descrevemos as características computacionais da abordagem.

5.1 Definição do Princípio da Mudança Mínima em Modelos CTL

Os postulados KM [KM91] capturam a ideia da mudança mínima da operação de atualização. Zhang e Ding buscaram então estender este conceito para o contexto da atualização de modelos CTL. Para tanto, propuseram métricas formais de mudança mínima para a atualização de modelos CTL.

Dada uma fórmula CTL ϕ e um modelo CTL $M = (S, R, L)$, onde S representa o conjunto de estados do modelo, R a relação entre estes estados e L as rotulações de cada estado. Quando $(M, s) \not\models \phi$ para um dado $s \in S$ devemos analisar como modificar M a fim de gerar um modelo M', tal que $(M', s') \models \phi$, onde s' é o estado correspondente a s o novo modelo. Dentre todas as modificações aplicáveis a um modelo CTL, Zhang e Ding identificaram um conjunto de cinco operações básicas capazes de derivar todas as demais mudanças:

PU1: **Adição de uma relação**. Dado $M = (S, R, L)$, um modelo $M' = (S', R', L')$ é obtido de M pela adição de um único novo elemento ao conjunto de relações. Isto é, $S' = S$, $L' = L$ e $R' = R \cup \{(s_i, s_j)\}$, onde $(s_i, s_j) \notin R$ para dois estados $s_i, s_j \in S$.

PU2: **Remoção de uma relação**. Dado $M = (S, R, L)$, um modelo $M' = (S', R', L')$ é obtido de M pela remoção de um único elemento do conjunto de relações. Isto é, $S' = S$, $L' = L$ e $R' = R - \{(s_i, s_j)\}$, onde $(s_i, s_j) \in R$ para dois estados $s_i, s_j \in S$.

PU3: **Mudança na função de rotulação**. Dado $M = (S, R, L)$, um modelo $M' = (S', R', L')$ é obtido de M pela mudança da função de rotulação para um único estado. Isto é, $S' = S$, $R' = R$, $s^* \in S$, $L'(s^*) \neq L(s^*)$ e $\forall s \in (S - s^*)$, $L'(s) = L(s)$.

PU4: **Adição de um estado**. Dado $M = (S, R, L)$, um modelo $M' = (S', R', L')$ é obtido de M pela adição de um único novo elemento ao conjunto de estados. Isto é, $S' = S \cup \{s^*\}$, onde $s^* \notin S$, $R' = R$ e $\forall s \in S$, $L'(s) = L(s)$ e $L'(s^*)$ é o conjunto de proposições verdadeiras no estado s^*.

PU5: Remoção de um estado. Dado $M = (S, R, L)$, um modelo $M' = (S', R', L')$ é obtido de M pela remoção de um único elemento do conjunto de estados. Isto é, $S' = S - \{s^*\}$, onde $s^* \in S$ e $\forall s \in S$, tal que $s \neq s^*$, $(s, s^*) \notin R$ e $(s^*, s) \notin R$, $R' = R$ e $\forall s \in S'$, $L'(s) = L(s)$.

Para ilustrar a aplicação das operações PU1-PU5, considere o modelo $M = \{\{s_0, s_1, s_2\}, \{(s_0, s_0), (s_0, s_1), (s_0, s_2), (s_1, s_1), (s_2, s_2), (s_2, s_1)\}, L\}$, onde $L(s_0) = \{p, q\}$, $L(s_1) = \{q, r\}$ e $L(s_2) = \{r\}$ (Figura 5.1).

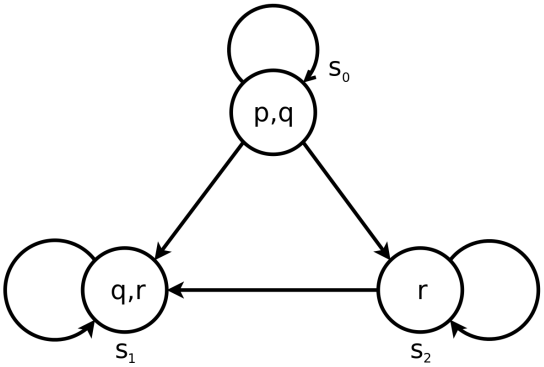

Figura 5.1: *Modelo exemplo M*

Desejamos que no estado s_0 de M a fórmula $AG(\neg q \vee \neg r)$ seja satisfeita. Uma maneira de atualizar M seria remover o estado s_1 do modelo, para tanto aplicamos três vezes a operação PU2, removendo de M as relações (s_0, s_1), (s_1, s_1) e (s_2, s_1), e em seguida aplicamos PU5, removendo então s_1 do modelo. Outra maneira seria simplesmente remover r do estado s_1, fazendo uso da operação PU3 para alterar a função de rotulação neste estado. Ambas atualizações de M satisfazem $(M, s_0) \models AG(\neg q \vee \neg r)$ e seus resultados podem ser vistos na Figura 5.2.

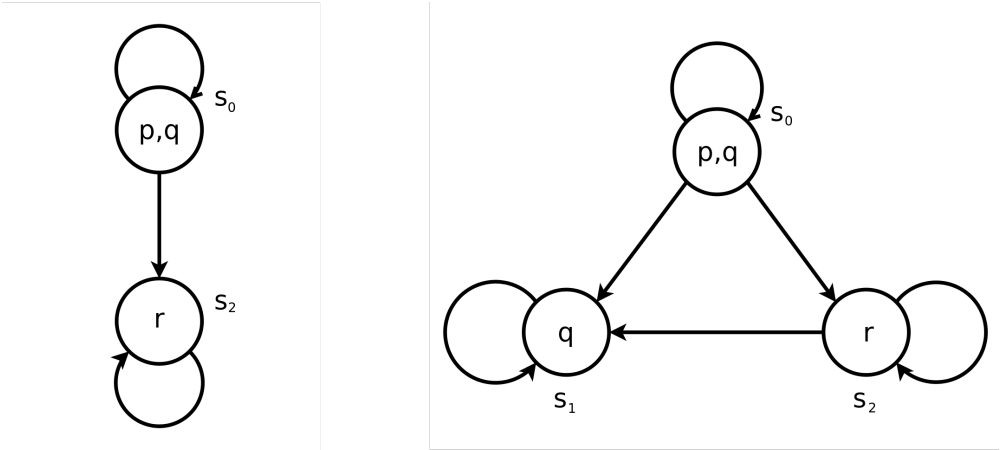

Figura 5.2: *À esquerda, a atualização de M utilizando PU2 e PU5. À direita, a atualização de M utilizando PU3.*

A operação PU3 não é necessariamente *primitiva*, visto que uma combinação das outras quatro operações pode produzir o mesmo efeito. No entanto, PU3 é substancialmente mais simples que utilizar uma sequência de PU1, PU2, PU4 e PU5. PU3 não modifica o nome do estado nem as relações deste são alvos de mudança no modelo original, apenas atribui um conjunto diferente de proposições para tal estado. O uso de PU3 é justificado pelos autores por simplificar a maneira de mensurar a diferença entre dois modelos.

Definidas as operações básicas, precisamos determinar como mensurar o quanto um modelo CTL mudou e, a partir disso, formalizar o critério de mudança mínima para a atualização de modelos

CTL. Zhang e Ding denotaram a diferença entre dois modelos, $M = (S, R, L)$ e $M' = (S', R', L')$, como $Diff(M, M') = (Diff_{PU1}(M, M'), Diff_{PU2}(M, M'), Diff_{PU3}(M, M'), Diff_{PU4}(M, M'), Diff_{PU5}(M, M'))$, onde

- $Diff_{PU1}(M, M') = R' - R$, ou seja, o conjunto das relações adicionadas pelas aplicações de PU1.
- $Diff_{PU2}(M, M') = R - R'$, ou seja, o conjunto das relações removidas pelas aplicações de PU2.
- $Diff_{PU3}(M, M') = \{s | s \in S \cap S' \text{ e } L(s) \neq L'(s)\}$, ou seja, o conjunto de todos os estados afetados pelas aplicações de PU3.
- $Diff_{PU4}(M, M') = S' - S$, ou seja, o conjunto dos estados adicionados pelas aplicações de PU4.
- $Diff_{PU5}(M, M') = S - S'$, ou seja, o conjunto dos estados removidos pelas aplicações de PU5.

Com base na função $Diff(M, M')$, os autores definiram uma relação de distância entre o modelo modificado e o modelo original, o que denominaram ordenação por proximidade (*closeness ordering*). Esta relação é descrita pela Definição 5.1.

Definição 5.1 ([ZD08]). Seja $M = (S, R, L)$, $M_1 = (S_1, R_1, L_1)$ e $M_2 = (S_2, R_2, L_2)$ três modelos CTL. Dizemos que M_1 é pelo menos tão próximo de M quanto M_2, denotado por $M_1 \leqslant_M M_2$, se e somente se para cada conjunto de operações básicas PU1-PU5 que transforme M em M_2, existe um conjunto de operações básicas PU1-PU5 que transforme M em M_1 tal que:

1. Para cada i $(1 \leq i \leq 5)$, $Diff_{PUi}(M, M_1) \subseteq Diff_{PUi}(M, M_2)$
2. Se $Diff_{PU3}(M, M_1) = Diff_{PU3}(M, M_2)$, então $\forall s \in Diff_{PU3}(M, M_1)$, $diff(L(s), L_1(s)) \subseteq diff(L(s), L_2(s))$, onde $diff(A, B) = (A - B) \cup (B - A)$ para quaisquer conjuntos A e B.

Denotamos por $M_1 <_M M_2$ se $M_1 \leqslant_M M_2$ e $M_2 \nleqslant_M M_1$.

Como ilustração, considere o seguinte exemplo retirado de [ZD08]. Seja M, M_1 e M_2 modelos CTL descritos pela Figura 5.3. O modelo M_1 é o resultado da atualização de M pela aplicação de PU1. Por sua vez, M_2 é resultado da atualização de M através das aplicações de PU1, PU2 e PU5. Temos então $Diff_{PU1}(M, M_1) = \{(s_0, s_2)\}$, e $Diff_{PU1}(M, M_2) = \{(s_0, s_2), (s_1, s_0)\}$, que resulta em $Diff_{PU1}(M, M_1) \subset Diff_{PU1}(M, M_2)$. Vemos também que $Diff_{PU2}(M, M_1) = \emptyset$ e $Diff_{PU2}(M, M_2) = \{(s_2, s_3), (s_3, s_0)\}$. Como em nenhum dos casos aplicamos PU3 ou PU4, temos $Diff_{PU3}(M, M_1) = \emptyset$, $Diff_{PU3}(M, M_2) = \emptyset$, $Diff_{PU4}(M, M_1) = \emptyset$ e $Diff_{PU4}(M, M_2) = \emptyset$. Por fim, temos $Diff_{PU5}(M, M_1) = \emptyset$ e $Diff_{PU5}(M, M_2) = \{s_3\}$. Assim, de acordo com a Definição 5.1, concluímos que $M_1 <_M M_2$.

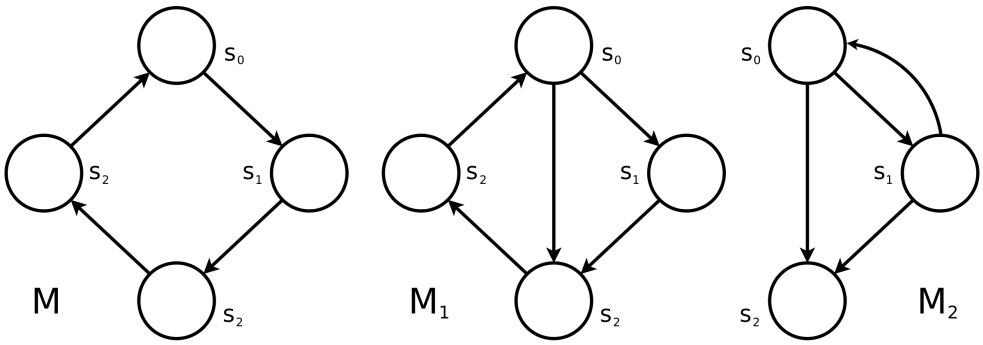

Figura 5.3: *Ilustração do conceito de ordenação por proximidade.*

Dado uma relação de ordem conforme a Definição 5.1, uma atualização segue o critério da mudança mínima, sendo chamada de *atualização admissível*, se e somente se a atualização está de acordo com a definição a seguir [ZD08].

Definição 5.2 ([ZD08]). Seja $M = (S, R, L)$ um modelo CTL e ϕ uma fórmula CTL. Uma atualização é dita *admissível*, denotada por $Update(M, \phi)$, se e somente se ela resulta em um modelo que satisfaz as seguintes condições:

1. $Update(M, \phi) = M'$ e $M' \vDash \phi$.
2. Não existe outro modelo M'' tal que $M'' \vDash \phi$, $M'' <_M M'$.

Denotamos por $Poss(Update(M, \phi))$ o conjunto de todas as atualizações admissíveis de M.

Supondo que M_1 e M_2 (Figura 5.3) satisfazem uma dada fórmula ϕ, por definição M_2 não é considerado uma atualização admissível, pois a atualização M_1 é mais próxima a M ($M_1 <_M M_2$). Além disso, como a modificação que gerou M_1 se constitui de uma única operação primitiva, não existe um modelo M' tal que $M' \neq M$ e $M' <_M M_1$, portanto M_1 é uma atualização admissível de M e, consequentemente, pertence ao conjunto $Poss(Update(M, \phi))$.

5.2 Propriedades Semânticas da Atualização de Modelos CTL

Uma das principais contribuições do trabalho [ZD08] provêm da comparação entre sua proposta e a abordagem tradicional de atualização de crenças proposicionais. Apesar de abordar um problema essencialmente diferente, e de certa forma mais complexo, a ideia de mudança miníma na atualização de modelos CTL está intimamente relacionada ao princípio presente na atualização de crenças.

Como dito na Seção 4.2, os postulados KM generalizam operadores de atualização que podem ser definidos por ordem ou pré-ordem parcial sobre interpretações. Zhang e Ding mostram que a relação de ordem $\leqslant_M$ determinada pela Definição 5.1 é uma relação de ordem parcial, o que é um indício de que os postulados também sejam capazes de generalizar operadores de atualização de modelos CTL.

A fim de tornar possível a comparação entre a atualização de modelos CTL e os postulados (U1)-(U8), o trabalho propõe um operador de atualização baseado no PMA, aplicável a modelos CTL. Este operador é denotado pelo símbolo $\diamond_c$ e seu funcionamento é especificado pela Definição 5.3.

Definição 5.3 ([ZD08]). Seja $M = (S, R, L)$ um modelo CTL, ϕ uma fórmula CTL e $Mod(\psi)$ o conjunto de modelos M, tal que $M \vDash \psi$. O operador de atualização de modelos CTL $\diamond_c$ é definido como:

$$Mod(\psi \diamond_c \phi) = \bigcup_{M \in Mod(\psi)} Poss(Update(M, \phi))$$

Zhang e Ding ([ZD08], Teorema 1) demonstram que o operador $\diamond_c$ satisfaz os postulados (U1)-(U8), mostrando que estes postulados possuem uma abrangência maior que sua tradicional aplicação em lógica proposicional. Sua demonstração também evidencia que os postulados KM são essenciais para qualquer abordagem de atualização de modelos.

5.3 Propriedades Computacionais da Atualização de Modelos CTL

O trabalho [ZD08] mostra que, dados dois modelos M e M', o problema de decidir se M' é um modelo admissível para M é co-NP-Completo. Zhang e Ding mostram de maneira informal como este problema tem sua complexidade limitada pela verificação de modelos.

Para determinar se M' é uma atualização admissível de M que satisfaça uma fórmula ϕ, precisamos verificar se $M' \vDash \phi$ e se este é minimal segundo a ordem $\leqslant_M$. Zhang e Ding [ZD08] exploram o problema complementar a este, verificando se M' *não* é um modelo minimal. Para

isso, (1) supomos outro modelo M'' que satisfaça ϕ e (2) testamos se $M'' <_M M'$. O passo (1) pode ser feito em tempo polinomial, por se tratar apenas da formulação de um novo modelo CTL. Já para a verificação do passo (2), precisamos computar $diff(S, S')$, $diff(S, S'')$, $diff(R, R')$ e $diff(R, R'')$. Estas funções são computadas em tempo polinomial e a partir delas podemos identificar $Diff_{PUi}(M, M')$ e $Diff_{PUi}(M, M'')$, para $i = 1, \ldots, 5$, também em tempo polinomial. Por fim, verificamos se $Diff_{PUi}(M, M') \subseteq Diff_{PUi}(M, M'')$, para $i = 1, \ldots, 5$, e caso $Diff_{PU3}(M, M') = Diff_{PU3}(M, M'')$, verificamos também se $diff(L(s), L(s')) \subseteq diff(L(s), L(s'))$, e se algum desses é um subconjunto próprio, a fim de determinar se $M'' <_M M'$. Ambos passos (1) e (2) podem ser computados em tempo polinomial com uma maquina de Turing não-determinística, ficando então a complexidade do problema determinada pela verificação de satisfação da fórmula ϕ pelo modelo M', que segundo [CGP99] pode ser realizada em tempo $O(|\phi| \times (|S| + |R|))$.

Uma demonstração formal deste resultado pode ser visto em [ZD08], Teorema 5. Como resultado, temos a implicação de que provavelmente não é possível desenvolver um algoritmo capaz de resolver o problema em tempo polinomial. Apesar disso, os autores identificaram uma subclasse de fórmulas CTL cuja atualização pode ser realizada em tempo polinomial, que será descrita na Seção 5.3.1.

5.3.1 Subclasse Tratável da Atualização de Modelos CTL

Zhang e Ding denominaram de **AEClass** o conjunto de fórmulas CTL definido a seguir:

1. fórmulas do tipo $AX\phi$, $AG\phi$, $AF\phi$, $A[\phi_1 U \phi_2]$, $EX\phi$, $EG\phi$, $EF\phi$ e $E[\phi_1 U \phi_2]$, onde ϕ, ϕ_1 e ϕ_2 são fórmulas proposicionais, pertencem a **AEClass**;

2. se ϕ_1 e ϕ_2 pertencem a **AEClass**, então $\phi_1 \vee \phi_2$ e $\phi_1 \wedge \phi_2$ pertencem a **AEClass**;

3. nenhuma outra fórmula pertence a **AEClass**.

A classe **AEClass** é um subconjunto de fórmulas CTL sem operadores temporais aninhados. Esta classe permite que a operação de atualização seja muito mais simples que em outros casos. Em geral, modelos admissíveis em **AEClass** podem ser alcançados por simples operações de adição ou remoção de relações.

Um estado ou caminho em um dado modelo é dito válido para uma fórmula **AEClass** se ele está de acordo com a Definição 5.4.

Definição 5.4 ([ZD08]). Seja $M = (S, R, L)$ um modelo CTL, $\psi \in$ **AEClass**, e $(M, s_0) \not\models \psi$, onde $s_0 \in S$. Um caminho válido ou estado válido de ψ em (M, s_0) é definido como

1. Se ψ é da forma $AX\phi$, então $s \in S$ é um estado válido de ψ em (M, s_0) se $(s_0, s) \in R$ e $L(s) \models \phi$;

2. Se ψ é da forma (a) $AG\phi$, (b) $AF\phi$ ou (c) $A[\phi_1 U \phi_2]$, então um $\pi = [s_0, \ldots]$ é um caminho válido de ψ em (M, s_0) se $\forall s \in \pi$, $L(s) \models \phi$ (caso (a)); $\exists s \in \pi$, $s_0 < s$ e $L(s) \models \phi$ (caso (b)); ou $\exists s \in S$, $s \models \phi_2$ e $\forall s' < s$, $L(s') \models \phi_1$ (caso (c)), respectivamente;

3. Se ψ é da forma $EX\phi$, então $s \in S$ é um estado válido de ψ em (M, s_0) se $L(s) \models \phi$;

4. Se ψ é da forma (a) $EG\phi$, (b) $EF\phi$ ou (c) $E[\phi_1 U \phi_2]$, então um $\pi = [s'_0, \ldots]$, onde $s'_0 \neq s_0$, é um caminho válido de ψ em (M, s_0) se $\forall s \in \pi$, $L(s) \models \phi$ e $L(s_0) \models \phi$ (caso (a)); $\exists s \in \pi$, $s'_0 < s$ e $L(s) \models \phi$ (caso (b)); ou $L(s_0) \models \phi_1$, $\exists s \in S$, $s \models \phi_2$ e $\forall s' < s$, $L(s') \models \phi_1$ (caso (c)), respectivamente;

Dizemos que uma fórmula arbitrária $\phi \in$ **AEClass** possui uma subestrutura válida[1] em um dado modelo se todas as fórmulas atômicas que ocorrem em ψ possuem um estado ou caminho válido com relação a este modelo. Uma fórmula **AEClass** é dita atômica se sua forma é como especificada no item 1 da definição de **AEClass**.

[1] O termo original utilizado por Zhang e Ding foi *valid witness*.

Zhang e Ding demonstram em [ZD08], Teorema 7, que dado um modelo $M = (S, R, L)$ e uma fórmula **AEClass** ψ, tal que $M \not\models \psi$, um modelo admissível $Update(M, \psi)$ pode ser computado em tempo polinomial, desde que ψ possua uma subestrutura válida em M. Por sua vez, verificar se ψ possui uma subestrutura válida também pode ser realizada em tempo polinomial ([ZD08], Teorema 6), o que torna polinomial todo o processo de atualização nessa classe de fórmulas.

Capítulo 6

Revisão de Modelos

Neste capítulo especificamos nossa abordagem para mudança de crenças CTL, a qual denominamos *revisão de modelos CTL*. Nossa abordagem baseia-se em revisão de crenças e difere da atualização de modelos de Zhang e Ding [ZD08] da mesma maneira que diferem revisão e atualização de crenças: o contexto ao qual é aplicada. Ilustramos essa diferença e argumentamos o porquê de não usar diretamente a teoria clássica. Definiremos formalmente nosso operador de revisão e analisaremos sua obediência aos postulados AGM.[1]

Na Seção 6.1 mostramos um caso onde a atualização de modelos não retorna o resultado esperado. Na Seção 6.2 argumentamos que a teoria clássica de revisão de crenças não pode ser diretamente aplicada à revisão de modelos CTL. Por último, definimos na Seção 6.4 nosso operador de revisão de modelos CTL e mostramos sua obediência aos postulados AGM.

6.1 Atualização Modelos em um Contexto Estático

A necessidade de uma nova abordagem (para mudanças em crenças CTL) é motivada pela diferença fundamental entre revisão e atualização de crenças. Como visto no Capítulo 4, aplicamos atualização quando desejamos incorporar à nossas crenças uma informação que retrata uma mudança no mundo. Desejamos deste modo adaptar nossa base de crenças a nova configuração que o mundo assumiu. Por outro lado, aplicamos revisão de crenças quando lidamos com um contexto estático, onde a nova informação nos permite refinar (ou corrigir) nossas crenças sobre um mundo que não mudou. O Exemplo 6.1 ilustra essa diferença.

Exemplo 6.1. *Tomemos o quarto do Exemplo 4.1. Neste quarto existem três objetos: um livro, uma revista e uma mesa. Acreditamos que, ou o livro está sobre a mesa (b) ou a revista está (m), mas não ambos. Suponha porém que, em vez de enviar o robô para pôr o livro sobre a mesa, como no Exemplo 4.1, enviamos o robô apenas para observar a posição do livro, sem alterá-la. Suponha então que quando o robô sai da sala ele nos informa que o livro está sobre a mesa. Desejamos assim incorporar a nova informação* $\phi \leftrightarrow b$ *à nossa base de crenças* $\psi \leftrightarrow (b \wedge \neg m) \vee (\neg b \wedge m)$.

Um operador de atualização geraria a nova base de crença $\psi' \leftrightarrow b$*, enquanto um operador de revisão geraria uma base* $\psi'' \leftrightarrow (b \wedge \neg m)$*. Observe que, mesmo sendo os mesmos resultados do Exemplo 4.1, nesta nova situação* ψ'' *é mais racional: costumávamos acreditar que o livro e a revista não estavam no mesmo lugar, fomos informado sobre a posição do livro e sabemos que a sala não foi alterada, assim não temos motivos para duvidar que a posição da revista ainda seja diferente da posição do livro. Assim, a revisão de crenças produziu um resultado mais informativo que a atualização.*

Observe que nos exemplos 4.1 e 6.1 a base de crenças ψ e a nova informação ϕ são idênticas, o que muda é como o robô interage com o mundo e como interpretamos isso. Acreditamos que o

[1] Uma versão preliminar dessa abordagem foi apresentada na 12a. Conferência Ibero-americana de Inteligência Artificial, Bahía Blanca, Argentina, 2010.

mesmo problema de distinção entre mundos estáticos e dinâmicos acontece também quando lidamos com o reparo de modelos inconsistentes da lógica CTL.

Existem dois principais motivos para modificar a modelagem de um sistema: (1) quando ela falha em satisfazer algum requisito da especificação formal e (2) quando ela precisa ser adaptada a um novo requisito. Quando modificamos uma especificação motivado por (1), repará-la por meio de uma abordagem baseada em revisão de crenças pode gerar um resultado mais informativo que aquele gerado pela atualização de modelos CTL, de um modo similar ao que ocorreu entre revisão e atualização de crenças no Exemplo 6.1.

No Exemplo 6.2 ilustramos uma situação onde um modelo de sistema falha em atender um requisito desejado. Veremos que neste exemplo a atualização de modelos gera possibilidades de solução que fogem à intenção inicial do sistema.

Exemplo 6.2. *Suponha que projetamos um computador alimentado por uma fonte infinita de energia. Nossa intenção é oferecer um equipamento para um uso ininterrupto, porém tivemos que incluir duas restrições no projeto: o computador deve, por razões de segurança, permanecer desligado até chegar ao consumidor final; e o computador deve eventualmente realizar algumas rotinas de atualização de software.*

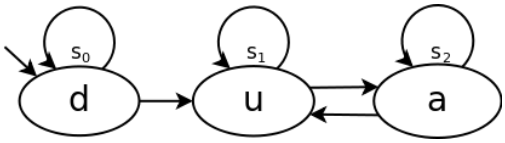

Figura 6.1: *Modelagem que satisfaz a especificação ψ.*

Seja d, u e a proposições que representem, respectivamente, os estados "desligado", "em uso" e "atualizando", a especificação do nosso sistema é dada pela seguinte fórmula CTL

$$\begin{aligned}\psi \;\leftrightarrow\; & d \;\wedge\; AG(d \rightarrow AX(d \vee u)) \;\wedge\; AG(u \rightarrow AX(u \vee a)) \;\wedge\; AG(a \rightarrow AX(u \vee a)) \\ & \wedge\; AG((d \wedge \neg u \wedge \neg a) \vee (\neg d \wedge u \wedge \neg a) \vee (\neg d \wedge \neg u \wedge a))\end{aligned}$$

A Figura 6.1 representa um possível modelo para o sistema. Depois de um tempo, alguns usuários queixaram-se que não estavam conseguindo utilizar o computador porque ele estava em constante atualização. Analisando a queixa, verificamos então que nosso projeto não garante que o estado "em uso" sempre possa ser alcançado. Em outras palavras, descobrimos que existem modelos em $Mod(\psi)$ que não satisfazem a seguinte fórmula CTL

$$\phi \;\leftrightarrow\; EF(u) \;\wedge\; AF(u)$$

A Figura 6.2 representa alguns dos modelos de ψ que não são modelos de ϕ. Acreditávamos que todos os modelos de ψ eram adequados, agora desejamos identificar um novo conjunto de modelos que satisfaçam ϕ e que sejam próximos à especificação original. Para isso, faremos uso de atualização de modelos e de seu operador $\diamond_c$.

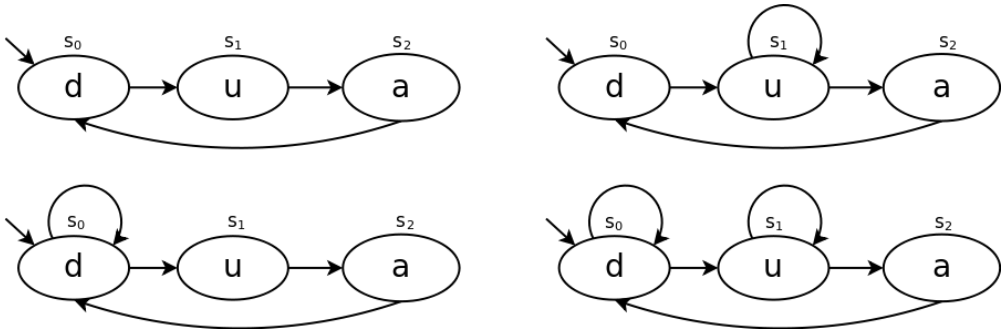

Figura 6.2: *Exemplos de modelos de ψ que não satisfazem ϕ.*

Pela definição de $\diamond_c$, o operador buscará os modelos de ϕ mais próximos aos modelos de ψ, analisando cada um destes individualmente. Tomemos os modelos M, M_1 e M_2 descritos na Figura 6.3. O modelo M pertence a $Mod(\psi)$, logo todo $M' \in Mod(\phi)$ minimal segundo $\leqslant_M$ pertencerá ao conjunto $Mod(\psi \diamond_c \phi)$. Dessa forma, temos que M_1 e M_2 pertencem ao conjunto $Mod(\psi \diamond_c \phi)$.

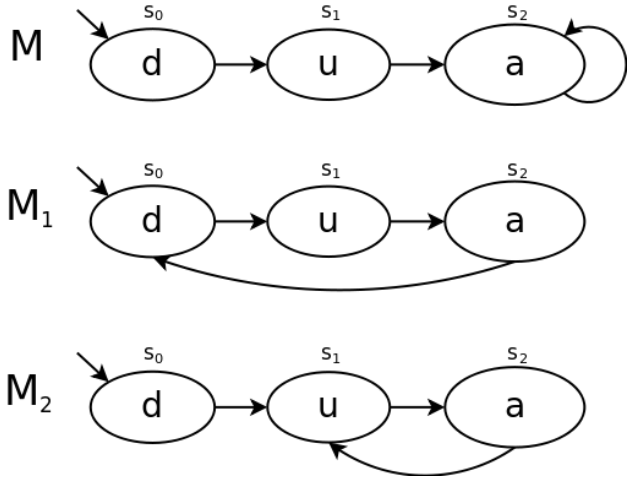

Figura 6.3: *Exemplos de possíveis atualizações de um modelos.*

Além de M_1, outros modelos de $\psi \diamond_c \phi$ possuem a transição de um estado "atualizando" para um estado "desligado". Como podemos ver, a atualização de modelos cogita como solução o desligamento do computador para que ele possa alcançar novamente o estado de uso. Porém isso não nos é intuitivo: por que desligar um computador que pode ficar permanentemente ligado?

O problema de utilizar atualização de modelos no Exemplo 6.2 diz respeito a natureza do novo requisito. O requisito era uma propriedade que o projeto já deveria satisfazer e não uma mudança a qual ele deveria se adequar. Percebemos que nossa especificação era indiferente ao requisito (alguns modelos satisfaziam, outros não) e isso estava incorreto. Porém, o natural seria manter apenas os modelos que já satisfazem a especificação e o requisito, e não adicionar novas possibilidades, como obtemos usando o operador $\diamond_c$. Mais a frente neste capítulo, veremos que uma abordagem baseada em revisão de crenças pode nos fornecer o conjunto de modelos que desejamos.

6.2 Revisão Clássica e Lógica Temporal CTL

Grande parte dos resultados conhecidos de revisão de crenças não se aplicam à lógica temporal CTL. Como vimos, ao tratar o problema da revisão é comum assumirmos algumas propriedades da lógica subjacente ao problema, as chamadas suposições AGM. Hansson e Wasserman [HW02] mostram que as suposições de compacidade e monotonicidade já são suficiente para que construções AGM típicas possam ser aplicadas.

Contudo, a CTL não é uma lógica compacta. Seja p um átomo proposicional e que $K = \{p, AXp, AXAXp, AXAXAXp, ...\}$ nosso conjunto de crenças, podemos inferir de K a fórmula $\alpha \leftrightarrow AGp$, visto que p é verdade em todos os estados alcançáveis. Se a lógica CTL fosse compacta, poderíamos definir um subconjunto finito $K' \subset K$ tal que $\alpha \in Cn(K')$, mas tal conjunto não existe.

O fato de CTL não satisfazer a compacidade significa que não podemos garantir que construções clássicas possam ser aplicadas a ela, ou mesmo que existam construções aplicáveis nos termos AGM. Em [RW06], vemos que a compacidade ainda é uma suposição necessária mesmo se assumirmos a representação de estados epistêmicos por bases de crenças.

Isso motiva a busca por operadores de revisão adequados à lógica temporal CTL, cuja racionalidade se aproxime daquela apresentada pela Teoria AGM.

6.3 Primeiras abordagens: Belief Revision NuSMV

A integração dos conceitos de verificação de modelos e revisão de crenças é abordada nos trabalhos [Sou07, SW07], com a criação do BrNuSMV[2]: um verificador capaz de gerar sugestões de atualização quando um dado modelo se mostra inconsistente com sua especificação. Resultado de um projeto de mestrado, o BrNuSMV consiste no verificador NuSMV [CCGR99] integrado de uma camada que, por meio do uso de técnicas de revisão de crença, é capaz de gerar sugestões de mudança no modelo de entrada. O trabalho propõe um algoritmo de revisão baseado no sistema de esferas para tratar caminhos diagnosticados como inconsistentes pelo verificador de modelos.

O algoritmo inicia verificando a consistência entre a descrição de um sistema e sua especificação. Quando inconsistente, o caminho contraexemplo é identificado e então se inicia o processo de revisão.

Buscando no caminho contraexemplo o último estado consistente, assume-se como crença as proposições válidas neste estado. Cria-se então um sistemas de esfera centralizado nos mundos onde tal crença é valida. Este sistema de esferas é ordenado por uma função d, que indica quantas proposições diferentes um mundo possui da crença (mundos com a mesma "distância" da crença pertencem a mesma esfera). Por fim, um operador de revisão para um sistema de esferas é aplicado, resultando em um pequeno conjunto de mundos consistentes com a especificação.

A última etapa é eliminar a alcançabilidade da inconsistência (Figura 6.4). Há duas possibilidades: eliminando a transição que leva ao estado inconsistente ou adicionando uma transição que se sobreponha a transição inconsistente, direcionando o fluxo a um estado do conjunto resultante da revisão. Essas sugestões são reunidas e retornadas como resultado. Maiores detalhes sobre o algoritmo podem ser vistos em [Sou07].

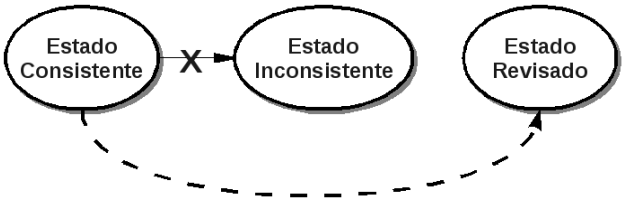

Figura 6.4: *Eliminando da inconsistência no BrNuSMV [Sou07].*

Essa abordagem explorou o uso prático de revisão de crenças e trouxe importantes contribuições para a área, sendo uma das primeiras implementações de revisão em uma ferramenta utilizada em modelagem formal. O trabalho mostra a eficácia da ferramenta, e consequentemente da revisão de crenças, como método de auxílio ao desenvolvimento de sistemas computacionais.

Talvez por ser um passo inicial na integração de verificação de modelos e revisão de crenças, existem no BrNuSMV algumas lacunas: não é possível tratar todos os tipos de fórmulas CTL; o processo de revisão é feito estado à estado, com limitações nos conjuntos de crenças; não há uma análise formal do processo de revisão de modelos. No nosso trabalho preenchemos algumas dessas lacunas. Realizamos uma análise formal do processo de revisão de modelos CTL genéricos e estudamos sua relação com abordagens clássicas de revisão de crenças.

6.4 Operador de Revisão de Modelos

Antes de especificar nosso método de revisão de modelos, precisamos definir uma relação de ordem entre modelos baseada na distância que um modelo possui de um dado conjunto de modelos.

Definição 6.1 (Ordenação por proximidade à conjuntos)**.** Seja $\mathcal{W}$ um conjunto de modelos CTL e $M_1 = (S_1, R_1, L_1)$ e $M_2 = (S_2, R_2, L_2)$ dois modelos CTL quaisquer, dizemos que M_1 é pelo menos tão próximo a $\mathcal{W}$ quanto M_2, denotado por $M_1 \leqslant_{\mathcal{W}} M_2$, se e somente se para cada conjunto de operações básicas PU1-PU5 que transforme um modelo $M' \in \mathcal{W}$ em M_2, existe um conjunto de operações básicas PU1-PU5 que transforme um modelo $M \in \mathcal{W}$ em M_1 tal que:

[2]Acrônimo para *Belief Revision NuSMV*.

1. Para cada i $(1 \leq i \leq 5)$, $Diff_{PUi}(M, M_1) \subseteq Diff_{PUi}(M', M_2)$
2. Se $Diff_{PU3}(M, M_1) = Diff_{PU3}(M', M_2)$, então $\forall s \in Diff_{PU3}(M, M_1)$, $diff(L(s), L_1(s)) \subseteq diff(L'(s), L_2(s))$, onde $diff(A, B) = (A - B) \cup (B - A)$ para quaisquer conjuntos A e B.

Denotamos por $M_1 <_{\mathcal{W}} M_2$ se $M_1 \leqslant_{\mathcal{W}} M_2$ e $M_2 \nleqslant_{\mathcal{W}} M_1$.

Como ilustração, considere novamente o Exemplo 6.2. Sejam M_1 e M_2 os modelos da Figura 6.3 e $\mathcal{W} = Mod(\psi)$, sabemos que M_2 pertence a $\mathcal{W}$, mas M_1 não. Assim, existe um modelo $M \in \mathcal{W}$ $(M = M_2)$ tal que para todo i $(1 \leq i \leq 5)$, $Diff_{PUi}(M, M_2) = \emptyset$. Temos que $Diff_{PUi}(M, M_2) \subseteq Diff_{PUi}(M', M_1)$, para qualquer i $(1 \leq i \leq 5)$ e $M' \in \mathcal{W}$, e portanto que $M_2 \leqslant_{\mathcal{W}} M_1$. Por outro lado, como $M_1 \notin \mathcal{W}$, qualquer que seja $M' \in \mathcal{W}$, ao menos uma das operações PUi deve ser aplicada para transformar M' em M_1. Em outras palavras, existe i $(1 \leq i \leq 5)$, tal que $Diff_{PUi}(M', M_1) \neq \emptyset$, e assim que $Diff_{PUi}(M', M_1) \nsubseteq Diff_{PUi}(M, M_2)$. Temos então $M_1 \nleqslant_{\mathcal{W}} M_2$ e, portanto, $M_2 <_{\mathcal{W}} M_1$.

Com base na Definição 6.1, especificamos a seguir nosso operador de revisão $\circ_c$ para fórmulas da lógica temporal CTL.

Definição 6.2. Dadas duas fórmulas CTL ψ e ϕ, denotamos por $\psi \circ_c \phi$ uma fórmula CTL resultado da atualização de ψ por ϕ, tal que

$$Mod(\psi \circ_c \phi) = Min_{Mod(\psi)}(Mod(\phi))$$

onde $Min_{\mathcal{B}}(\mathcal{A})$ denota o conjunto de todos os modelos minimais de $\mathcal{A}$ com relação a ordem $\leqslant_{\mathcal{B}}$.

Para entender a seleção realizada por $\circ_c$, voltemos ao Exemplo 6.2. Em vez de usar o operador $\diamond_c$ para compor o novo conjunto de modelos, utilizaremos o nosso operador de revisão. Segundo a Definição 6.2, $\circ_c$ selecionará os modelos de ϕ que sejam minimais segundo a ordem $\leqslant_{Mod(\psi)}$, isto é, $Mod(\psi \circ_c \phi)$ conterá apenas os modelos de ϕ que mais se aproximem do conjunto de modelos de ψ. Seja M_1 e M_2 os modelos descritos pela Figura 6.3, vimos que $M_2 <_{Mod(\psi)} M_1$, logo M_1 não é minimal segundo a ordem $<_{Mod(\psi)}$ e portanto $M_1 \notin Mod(\psi \circ_c \phi)$. De fato, como $Mod(\psi) \cap Mod(\phi) \neq \emptyset$, nenhum modelo $M \notin Mod(\psi)$ pertencerá a $Mod(\psi \circ_c \phi)$, pois sempre seremos capazes de escolher um $M' \in Mod(\phi)$ tal que $M' <_{Mod(\psi)} M$. Isso significa que não cogitaremos, por exemplo, desligar o computador como solução para o problema, já que isso não condiz com nossa especificação original. O operador de revisão manterá a coerência com a especificação, selecionando apenas os modelos que já satisfazem esta e o novo requisito, sem a adição de outras possibilidades.

6.4.1 Propriedades Semânticas

O Teorema 8 descreve o principal relacionamento da nossa abordagem com a teoria clássica de revisão de crenças: a obediência aos postulados AGM.

Teorema 8. *O operador $\circ_c$ satisfaz os postulados de revisão (R1)-(R6).*

Demonstração. O postulado (R1) é satisfeito diretamente pela definição de $\circ_c$. Para satisfazer (R2) temos que provar que, para $\psi \wedge \phi$ satisfazível, $Mod(\psi \circ_c \phi) = Mod(\psi \wedge \phi)$. Se $\psi \wedge \phi$ é satisfazível, então $Mod(\psi \wedge \phi) = Mod(\psi) \cap Mod(\phi) \neq \emptyset$. Todo $M \in Mod(\psi) \cap Mod(\phi)$ é modelo de ϕ e tem distância mínima do conjunto $Mod(\psi)$ por já pertencer ao conjunto, logo $Mod(\psi \wedge \phi) \subseteq Mod(\psi \circ_c \phi)$. Suponha agora que $Mod(\psi \circ_c \phi) \nsubseteq Mod(\psi \wedge \phi)$, assim deve existir ao menos um modelo $M' \in Mod(\psi \circ_c \phi)$ tal que $M' \notin Mod(\psi) \cap Mod(\phi)$. Como, por definição, escolhemos sempre modelos de $Mod(\phi)$, necessariamente $M' \notin Mod(\psi)$. Porém, qualquer que seja $M'' \in Mod(\psi) \cap Mod(\phi)$, temos $M'' <_{Mod(\psi)} M'$, o que contradiz o fato de M' ser minimal segundo a ordem $\leqslant_{Mod(\psi)}$. Assim, $Mod(\psi \wedge \phi) \subseteq Mod(\psi \circ_c \phi)$ e portanto $Mod(\psi \wedge \phi) = Mod(\psi \circ_c \phi)$.

Agora provamos que $\circ_c$ satisfaz (R3). Se ϕ é satisfazível, então $Mod(\phi) \neq \emptyset$, o que significa que existem modelos a serem selecionados por $\circ_c$, logo $Mod(\psi \circ_c \phi) \neq \emptyset$ e, portanto, $\psi \circ_c \phi$ é satisfazível.

Para provar que $\circ_c$ satisfaz (R4) é suficiente mostrar que se $\psi_1 \leftrightarrow \psi_2$ e $\phi_1 \leftrightarrow \phi_2$, então $Mod(\psi_1 \circ_c \phi_1) = Mod(\psi_2 \circ_c \phi_2)$. Assumindo $Mod(\psi_1) = Mod(\psi_2)$ e $Mod(\phi_1) = Mod(\phi_2)$, temos que $Mod(\psi_1 \circ_c \phi_1) = Min_{Mod(\psi_1)}(Mod(\phi_1)) = Min_{Mod(\psi_1)}(Mod(\phi_2)) = Min_{Mod(\psi_2)}(Mod(\phi_2)) = Mod(\psi_2 \circ_c \phi_2)$.

Para provar que $\circ_c$ satisfaz (R5), devemos mostrar que $Mod((\psi \circ_c \phi) \wedge \mu) \subseteq Mod(\psi \circ_c (\phi \wedge \mu))$. Se $(\psi \circ_c \phi) \wedge \mu$ não é satisfazível, então $Mod((\psi \circ_c \phi) \wedge \mu) = \emptyset$ e a demonstração é trivial. Consideremos então apenas o caso onde $(\psi \circ_c \phi) \wedge \mu$ é satisfazível. Suponhamos que $Mod((\psi \circ_c \phi) \wedge \mu) \subseteq Mod(\psi \circ (\phi \wedge \mu))$ não seja verdade. Isso quer dizer que existe $M \in Mod((\psi \circ_c \phi)) \cap Mod(\mu)$ tal que $M \notin Mod(\psi \circ (\phi \wedge \mu))$. Por (R1), sabemos que $M \in Mod(\phi)$ e portanto que $M \in Mod(\phi \wedge \mu)$. Para que $M \notin Mod(\psi \circ (\phi \wedge \mu))$ deve existir um modelo $M' \in Mod(\psi \circ (\phi \wedge \mu))$ tal que $M' <_{Mod(\psi)} M$. Porém, $M' \in Mod(\phi)$, o que contradiz o fato de M ser minimal segundo $\leqslant_{Mod(\psi)}$. Logo, $Mod((\psi \circ_c \phi) \wedge \mu) \subseteq Mod(\psi \circ (\phi \wedge \mu))$ é valido.

Finalmente, mostramos que $\circ_c$ satisfaz (R6). Para tanto devemos provar que $Mod(\psi \circ_c (\phi \wedge \mu)) \subseteq Mod((\psi \circ_c \phi) \wedge \mu)$, quando $Mod((\psi \circ_c \phi) \wedge \mu) \neq \emptyset$. Se $\psi \circ_c (\phi \wedge \mu)$ não é satisfazível, a demonstração é trivial. Consideremos então apenas o caso onde esta fórmula é satisfazível. Suponha então que $Mod(\psi \circ_c (\phi \wedge \mu)) \nsubseteq Mod((\psi \circ_c \phi) \wedge \mu)$, isso significa que existe $M \in Mod(\psi \circ_c (\phi \wedge \mu))$ tal que $M \notin Mod(\psi \circ_c \phi) \cap Mod(\mu)$. Por (R1), sabemos que $M \in Mod(\phi \wedge \mu)$ e assim que $M \in Mod(\mu)$. Para que $M \notin Mod(\psi \circ_c \phi)$ todo modelo $M' \in Mod(\psi \circ_c \phi)$ deve ser tal que $M' <_{Mod(\psi)} M$. Como $Mod((\psi \circ_c \phi) \wedge \mu) \neq \emptyset$, sabemos que existe um $M' \in Mod(\phi \wedge \mu)$, o que contradiz o fato de M ser minimal segundo $\leqslant_{Mod(\psi)}$. Logo, $Mod(\psi \circ_c (\phi \wedge \mu)) \subseteq Mod((\psi \circ_c \phi) \wedge \mu)$, quando $(\psi \circ_c \phi) \wedge \mu$ é satisfazível. E isto completa nossa prova.

$\square$

Capítulo 7

Algoritmo para Revisão de Modelos CTL

Neste capítulo descrevemos nosso algoritmo para revisão de modelos. Este algoritmo gera modelos que satisfazem uma dada fórmula CTL e que são próximos a um determinado conjunto de modelos, seguindo os princípios apresentados no Capítulo 6. Tanto a fórmula, quanto o conjunto de modelos devem ser fornecidos como entrada para algoritmo.

Na Seção 7.1 apresentamos algumas considerações preliminares sobre o algoritmo, onde mostramos motivações ao seu desenvolvimento e argumentamos sobre a necessidade de assumir algumas restrições. Na Seção 7.2 descrevemos o algoritmo, suas funções e a ideia principal por trás de cada modificação.

7.1 Considerações Preliminares

O motivo principal para formular um algoritmo de revisão é permitir que nossa abordagem seja integrada a ferramentas reais de diagnóstico e reparo de sistemas. Desejamos assim que um projetista possa fornecer modelos inconsistentes com uma dada especificação, e que o nosso método devolva um conjunto de modelos, semelhante a entrada, mas consistente. Estes modelos representam as possíveis correções para a modelagem do sistema.

Zhang e Ding [ZD08] também propuseram um algoritmo para geração de modelos corrigidos, porém seu algoritmo não corresponde diretamente a abordagem de atualização de modelos. O algoritmo de Zhang e Ding recebe como entrada uma fórmula e um modelo CTL, e devolve como saída um modelo que satisfaz os critérios admissibilidade da atualização de modelos (ver Capítulo 4). Muitos passos deste algoritmo envolvem escolhas não-determinísticas, o que implica que duas execuções distintas para uma mesma entrada podem gerar resultados diferentes. Não está claro, contudo, como esse algoritmo pode ser utilizado para implementar a abordagem de atualização de modelos.

Para implementar a revisão de modelos, poderíamos utilizar o algoritmo de Zhang e Ding [ZD08] para selecionar dentre os modelos $Mod(\phi)$ aqueles estão mais próximos de cada modelo de $Mod(\psi)$, e selecionar neste resultado, aqueles que mais se aproximam do conjunto $Mod(\psi)$ (com os critérios vistos no Capítulo 3). De fato, este esboço é o algoritmo que almejamos (um pseudo-código para ele está em [GW10]), porém ele não é viável computacionalmente.

Um dos problemas de revisar (ou atualizar) os modelos de uma fórmula CTL ψ é que mesmo fórmulas simples podem ter infinitos modelos. Assim, implementar o operador $\circ_c$ utilizando o algoritmo de [ZD08], por meio de chamadas sucessivas para cada um dos modelo em $Mod(\psi)$, fatalmente gera um algoritmo que nunca para. Assim, nossa primeira restrição é que a entrada seja composta por um conjunto finito de modelos $W \subseteq Mod(\psi)$.

Porém, mesmo com a restrição imposta, o algoritmo de Zhang e Ding [ZD08] continua não apropriado para implementar a revisão de modelos. Para a revisão de um conjunto W precisamos de todos os modelos admissíveis em relação a cada um de seus elementos, para então selecionar aqueles mais próximos ao conjunto. Para gerar todas as atualizações admissíveis, além de iterar sobre todos os modelos em W, devemos explorar todas as possíveis escolhas que podem ser tomadas nos passos

não-determinísticos do algoritmo de [ZD08]. O problema é que para algumas dessas escolhas, como as que envolvem seleções de caminhos, podem existir infinitas possibilidades (na Figura 2.2, por exemplo, podemos ver que existem infinitos caminhos iniciando de s_0).

Utilizar o algoritmo de [ZD08] leva a um algoritmo que nunca para. Definimos portanto nosso próprio conjunto de funções para buscar os o conjunto de modelos admissíveis. Nosso método difere do de Zhang e Ding por dois aspectos principais: todas as escolhas são determinísticas; e todas as iterações são feitas sobre elementos finitos, como o conjunto de estados ou o conjunto de transições. Na seção seguinte detalhamos nosso algoritmo.

7.2 Principais Funções do Algoritmo

O algoritmo proposto utiliza subfunções específicas para cada tipo de operador temporal, cada uma responsável por um conjunto próprio de modificações. Além disso, essas subfunções são combinadas por meio de chamadas recursivas à função principal para explorar o aninhamento dos operadores temporais. Esse processo assemelha-se ao realizado pelo algoritmo de verificação de modelos apresentado na Seção 2.2. Todas as principais funções do algoritmo estão descritas na forma de pseudo-códigos.

Algoritmo 1: Revisão$_{CTL}(W, \phi)$

Entrada: W e ϕ, onde $W = \{(M, s_0) \mid M = (S, R, L)$ e $s_0 \in S\}$ e ϕ é uma fórmula CTL.
Saída: W', representando a revisão de W por ϕ.

1 **se** $\exists (M, s_0) \in W$ *tal que* $(M, s_0) \vDash \phi$ **então**
2 | $W' \leftarrow \{(M, s_0) \mid (M, s_0) \in W$ tal que $(M, s_0) \vDash \phi\}$
3 **senão**
4 | $W' \leftarrow \emptyset$
5 | $\phi_t \leftarrow$ tradução de ϕ em termos dos operadores EX, AF e EU
6 | **para cada** $(M, s_0) \in W$ **faça**
7 | | **caso**
8 | | | ϕ_t seja uma fórmula proposicional: $W' \leftarrow W' \cup$ Revisão$_{prop}((M, s_0), \phi_t)$
9 | | | ϕ_t seja do tipo $\neg\phi_1$: $W' \leftarrow W' \cup$ Revisão$_{\neg}((M, s_0), \neg\phi_1)$
10 | | | ϕ_t seja do tipo $\phi_1 \wedge \phi_2$: $W' \leftarrow W' \cup$ Revisão$_{\wedge}((M, s_0), \phi_1 \wedge \phi_2)$
11 | | | ϕ_t seja do tipo $\phi_1 \vee \phi_2$: $W' \leftarrow W' \cup$ Revisão$_{\vee}((M, s_0), \phi_1 \vee \phi_2)$
12 | | | ϕ_t seja do tipo $\mathrm{EX}\,\phi_1$: $W' \leftarrow W' \cup$ Revisão$_{EX}((M, s_0), \mathrm{EX}\,\phi_1)$
13 | | | ϕ_t seja do tipo $\mathrm{AF}\,\phi_1$: $W' \leftarrow W' \cup$ Revisão$_{AF}((M, s_0), \mathrm{AF}\,\phi_1)$
14 | | | ϕ_t seja do tipo $\mathrm{E}[\,\phi_1\,\mathrm{U}\,\phi_2\,]$: $W' \leftarrow W' \cup$ Revisão$_{EU}((M, s_0), \mathrm{E}[\,\phi_1\,\mathrm{U}\,\phi_2\,])$
15 | | **fim**
16 | **fim**
17 | **para todo** $(M', s'_0) \in W'$ *tal que* $\exists (M'', s''_0) \in W'$ *e* $M'' <_W M'$ **faça**
18 | | $W' \leftarrow W' - \{(M', s'_0)\}$
19 | **fim**
20 **fim**
21 **retorna** *W'*

A entrada do Algoritmo 1 consiste de dois elementos: um conjunto finito W de pares (M, s) (onde $M = (S, M, L)$ é um modelo CTL e $s \in S$ seu único estado inicial) e uma fórmula CTL ϕ, representando a fórmula pela qual desejamos revisar. Para simplificar, aqui chamamos de *modelo* o par $(M, s) \in W$. O algoritmo devolve como resultado um conjunto de modelos W' que satisfaz ϕ e é minimamente próximo dos modelos de W, conforme os critérios de mudança mínima definidos no capítulo anterior.

O algoritmo efetua a revisão da seguinte forma: primeiro verificamos se algum dos modelos de entrada satisfaz a fórmula ϕ. Se sim, nosso algoritmo realiza um refinamento nestes modelos,

eliminando aqueles inconsistentes com ϕ (linha 2), compondo assim a resposta do algoritmo. Caso contrário, buscaremos modelos consistentes com ϕ, gerados através de mudanças mínimas nos modelos de entrada. Na linha 5 reescrevemos a fórmula de entrada em termos dos operadores EX, AF e EU. Isso nos permite especificar apenas três funções de modificação, em vez de uma para cada operador temporal. Nas linhas 7-15 realizamos efetivamente as modificações chamando funções especificas de acordo com a estrutura da fórmula ϕ. Com isso derivamos da entrada um conjunto de modelos que satisfazem ϕ. Por fim, nas linhas 17-19, analisamos o conjunto de modelos gerado, onde selecionamos os elementos mais próximos aos modelos de entrada para compor o resultado da revisão de modelos.

A seguir apresentamos como realizar modificações em modelos CTL para que fórmulas de determinado tipo sejam satisfeitas. A Seção 7.2.1 apresenta as ideias chave de como modificar modelos para satisfazer os operadores temporais EX, AF e EU. Na seção 7.2.2 vemos como tratar a satisfação de fórmulas proposicionais. Por fim, na Seção 7.2.3 tratamos da conjunção, disjunção e negação de fórmulas CTL.

7.2.1 Lidando com Operadores Temporais

As funções a seguir fornecem as ideias principais de como modificar minimamente um modelo de modo que este satisfaça uma dada fórmula temporal. Como vimos no Capítulo 2, todas as fórmulas temporais podem ser escritas em função de EX, AF e EU. Mostramos então como adaptar um modelo para satisfazer cada um destes operadores.

Função Revisão$_{EX}((M, s_0), \mathrm{EX}\,\phi)$

Entrada: (M, s_0) e $\mathrm{EX}\,\phi$, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \not\models \mathrm{EX}\,\phi$.
Saída: W tal que $\forall (M', s_0') \in W$, $(M', s_0') \models \mathrm{EX}\,\phi$.

1 $W \leftarrow \emptyset$
2 **para todo** $s \in S$ *tal que* $(s_0, s) \in R$ **faça**
3 | $W \leftarrow W \cup$ Revisão$_{\mathrm{CTL}}(\{(M, s)\}, \phi)$
4 **fim**
5 **para todo** $s \in S$ *tal que* $(M, s) \models \phi$ **faça**
6 | Aplique PU1 para criar um novo modelo $M' = (S', R', L')$:
7 | $S' = S$, $R' = R \cup \{(s_0, s)\}$, $\forall s \in S$, $L'(s) = L(s)$.
8 | $W \leftarrow W \cup \{M'\}$
9 **fim**
10 Aplique PU4 e PU1 para criar um novo modelo $M' = (S', R', L')$:
11 $S' = S \cup \{(s^*)\}$, $R' = R \cup \{(s_0, s^*)\}$, $\forall s \in S$, $L'(s) = L(s)$, $L'(s^*) = \{\#\}$.
12 $W \leftarrow W \cup$ Revisão$_{\mathrm{CTL}}(\{(M, s^*)\}, \phi)$
13 $W \leftarrow$ Expanda(W)
14 **retorna** W

A função Revisão$_{EX}$ modifica o modelo de entrada de três maneiras distintas a fim de satisfazer $\mathrm{EX}\,\phi$. No primeiro caso, linhas 2-4, modificamos os estados vizinhos à s_0 para que estes satisfaçam ϕ. No segundo, linhas 5-9, adicionamos transições de s_0 à estados de M que já satisfazem ϕ. Por último, linhas 10-12, criamos uma transição de s_0 a um novo estado, fazendo com que ϕ seja satisfazível neste estado. Em cada um dos casos buscamos modificações mínimas de (M, s_0) para que este satisfaça $\mathrm{EX}\,\phi$, seja adiando a modificação para um próximo passo do algoritmo, seja adicionando uma única transição, seja adicionando um único novo estado ao modelo.

Na linha 11, ao criamos um novo estado, fazemos com que seu rótulo seja o símbolo # em vez de um conjunto de átomos proposicionais. Nosso objetivo é adiar a escolha desta rotulação até sabermos quais proposições são adequadas, o que ocorrerá apenas ao analisarmos ϕ. Desse modo, adiamos a escolha do rótulo para quando este for necessário à satisfação de uma fórmula.

Nem todos os rótulos # são alterados ao fim das modificações. Por exemplo, para que um

novo estado s satisfaça $\phi \leftrightarrow \mathrm{EX}\, p$, precisamos apenas que ele seja conectado a um estado s', onde $p \in L(s')$. Com isso, mesmo após todas as modificações no modelo, podemos ainda ter $L(s) = \{\#\}$, visto que esse rótulo não é chave para satisfazer a fórmula. Para resolver isso, aplicamos uma função de expansão (linha 13) a fim de substituir modelos que possuem rotulações #, por modelos semelhantes, mas rotulados por proposições.

Função Revisão$_{AF}((M, s_0), \mathrm{AF}\, \phi)$

Entrada: (M, s_0) e $\mathrm{AF}\, \phi$, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \not\models \mathrm{AF}\, \phi$.
Saída: W tal que $\forall (M', s'_0) \in W$, $(M', s'_0) \models \mathrm{AF}\, \phi$.

1 $W \leftarrow \emptyset$
2 $\mathcal{S}_\pi \leftarrow \{S \mid S =$ estados de $\pi = [s_0, s_1, ...]$ em M, tal que $\forall s_i \in \pi$, $(M, s_i) \not\models \phi\}$
3 $\mathcal{S}_\star \leftarrow \{S_\star \mid \forall S \in \mathcal{S}_\pi$, $S_\star \cap S \neq \emptyset$, e $\forall S' \subset S_\star$, $\exists S \in \mathcal{S}_\pi$ tal que $S \cap S' = \emptyset\}$
4 **para todo** $S_\star \in \mathcal{S}_\star$ **faça**
5 | $W' \leftarrow \{(M, s_0)\}$
6 | **enquanto** $\exists (M', s'_0) \in W'$ *e* $\exists s_\star \in S_\star$ *tal que* $(M', s_\star) \not\models \phi$ **faça**
7 | | $W' \leftarrow W' - \{(M', s'_0)\}$
8 | | $W' \leftarrow W' \cup$ Revisão$_{\mathrm{CTL}}(\{(M', s_\star)\}, \phi)$
9 | **fim**
10 | $W \leftarrow W \cup W'$
11 **fim**
12 Aplique PU2 para criar $M' = (S', R', L')$: $S' = S$,
13 $R' = R - \{(s_0, s) \mid (s_0, s) \in R$ e $(M, s) \not\models \mathrm{AF}\, \phi\}$,
14 $\forall s \in S$, $L'(s) = L(s)$.
15 $W \leftarrow W \cup \{(M', s'_0)\}$
16 **retorna** W

A função Revisão$_{AF}$ trata das modificações em (M, s_0) para satisfazer uma fórmula do tipo $\mathrm{AF}\, \phi$. Primeiro guardamos em S_π os conjuntos de estados de caminhos inconsistentes com $\mathrm{AF}\, \phi$. Depois selecionamos em $S_\star$ os conjuntos minimais de estados que contenham um representante de cada conjunto em S_π. Cada elemento de $S_\star$ representa um conjunto mínimo de estados do modelo original que devem ser modificados para que $\mathrm{AF}\, \phi$ seja satisfeito. Essas modificações são então realizadas nas linhas 5-9. Por último, linhas 12-14, criamos um novo modelo que satisfaz $\mathrm{AF}\, \phi$ retirando transições que partem do estado inicial e levam a caminhos inconsistentes. Em cada um dos casos buscamos modificações mínimas de (M, s_0) para que este satisfaça $\mathrm{AF}\, \phi$, seja modificando o mínimo necessário de estados, seja removendo a ligação entre o estado inicial e caminhos inconsistentes.

Função Revisão$_{EU}$((M, s_0), $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$)

Entrada: (M, s_0) e $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \not\models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$.
Saída: W tal que $\forall (M', s_0') \in W$, $(M', s_0') \models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$.

1 $W \leftarrow \emptyset$
2 $S_\star \leftarrow \{s \mid s \in S$ tal que $\exists \pi = [s_0, ..., s, ...]$ em M, onde $\forall s_i < s, (M, s_i) \models \phi_1\}$
3 **para todo** $s_\star \in S_\star$ **faça**
4 | $W \leftarrow W \cup$ Revisão$_{\mathrm{CTL}}(\{(M, s_\star)\}, \phi_2)$
5 | **se** $(M, s_\star) \not\models \phi_1$ *e* $\exists (s_\star, s) \in R$ *tal que* $(M, s) \models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ **então**
6 | | $W \leftarrow W \cup$ Revisão$_{\mathrm{CTL}}(\{(M, s_\star)\}, \phi_1)$
7 | **fim**
8 | **se** $(M, s_\star) \models \phi_1$ **então**
9 | | **para todo** $s \in S$ *tal que* $(M, s) \models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ **faça**
10 | | | Aplique PU1 para criar um novo modelo $M' = (S', R', L')$, tal que
11 | | | $S' = S$, $R' = R \cup \{(s_\star, s)\}$, $\forall s \in S$, $L'(s) = L(s)$.
12 | | | $W \leftarrow W \cup \{M'\}$
13 | | **fim**
14 | | Aplique PU4 e PU1 para criar um novo modelo $M' = (S', R', L')$:
15 | | $S' = S \cup \{(s^*)\}$, $R' = R \cup \{(s_\star, s^*)\}$, $\forall s \in S$, $L'(s) = L(s)$,
16 | | $L'(s^*) = \{\#\}$.
17 | | $W \leftarrow W \cup$ Revisão$_{\mathrm{CTL}}(\{(M, s^*)\}, \phi_2)$
18 | **fim**
19 **fim**
20 $W \leftarrow$ Expanda(W)
21 **retorna** W

A função Revisão$_{EU}$ retorna um conjunto de modelos que satisfazem $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$, modificando a entrada de quatro maneiras distintas. Primeiro denominamos por $S_\star$ o conjunto de estados s tal que existe um caminho entre s_0 e s onde todos os antecessores de s satisfazem ϕ_1. Cada elemento de $S_\star$ representa pontos chave para que mudanças levem a satisfação de $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$. Assim, para cada estado s em $S_\star$ aplicamos as seguinte modificações: (1) modificamos s para satisfazer ϕ_2 e então gerar um caminho consistente com $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ (linha 4); (2) se ϕ_1 não vale em s, mas este é o único empecilho a existência de um caminho consistente, modificamos s para que satisfaça ϕ_1 (linhas 5-7); (3) se ϕ_1 vale em s, adicionamos transições de s a estados onde $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ já vale (linhas 9-13); e (4) se ϕ_1 vale em s, criamos uma transição de s a um novo estado, fazendo com que ϕ_2 seja satisfazível neste estado (linha 14). Em cada um dos casos buscamos modificações mínimas de (M, s_0) para que este satisfaça $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$, seja modificando um único estado, seja adicionando uma única transição entre estados, seja adicionando um único novo estado.

7.2.2 Lidando com Fórmulas Proposicionais

Para satisfazer uma fórmula proposicional devemos redefinir quais conjuntos de proposições são verdadeiras naquele estado, para que a fórmula de entrada seja satisfeita. Como nesse caso a satisfação da fórmula não envolve relações entre estados, a modificação se restringe a escolha adequada do novo rótulo do estado.

Função Revisão$_{prop}((M, s_0), \phi)$

Entrada: (M, s_0) e ϕ, onde $M = (S, R, L)$, $s_0 \in S$.
Saída: W tal que $\forall (M', s'_0) \in W$, $L'(s'_0) \models \phi$.

1 $W \leftarrow \emptyset$
2 **para cada** $P \subseteq 2^{AP}$ *tal que* $P \models \phi$ *e* $\text{diff}(L(s_0), P)$ *é minimal* **faça**
3 | Aplique PU3 para criar um novo modelo $M' = (S', R', L')$:
4 | $S' = S$, $R' = R$, $\forall s \in S - \{s_0\}$, $L'(s) = L(s)$ e $L'(s_0) = P$.
5 | $W \leftarrow W \cup \{M'\}$
6 **fim**
7 **retorna** W

A função Revisão$_{prop}$ é o último elemento das chamadas recursivas feita pelo algoritmo. Quando uma fórmula proposicional ϕ não vale no estado s_0, buscamos as mudanças mínimas na rotulação deste para que ϕ seja satisfeita. Se $L(s_0) = \{\#\}$, fazemos com que o novo rótulo de s_0 seja um conjunto de proposições tal que todos os seus elementos sejam estritamente necessários à satisfação de ϕ[1].

7.2.3 Lidando com Conjunções, Disjunções e Negação

Conjunções e disjunções são tratadas de modo direto, pela combinação de resultado ou pela união deles. No entanto a negação necessita de uma atenção especial, pois modificações que invalidam uma fórmula também devem ser realizadas de modo mínimo.

Função Revisão$_{\wedge}((M, s_0), \phi_1 \wedge \phi_2)$

Entrada: (M, s_0), ϕ_1 e ϕ_1, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \not\models \phi_1 \wedge \phi_2$.
Saída: W tal que $\forall (M', s'_0) \in W$, $(M', s'_0) \models \phi_1 \wedge \phi_2$.

1 $W \leftarrow$ Revisão$_{\text{CTL}}(\{(M, s_0)\}, \phi_1)$
2 $W \leftarrow$ Revisão$_{\text{CTL}}(W, \phi_2)$, restrito por ϕ_1
3 **retorna** W

A função Revisão$_{\wedge}$ é semelhante a apresentada por [ZD08]. Para revisar o modelo pela conjunção ϕ_1 e ϕ_2, buscamos primeiro o conjunto de modelos resultado de revisar (M, s_0) por ϕ_1. Em seguida revisamos estes modelos pela fórmula ϕ_2. A restrição imposta na linha 2 indica que devemos executar a revisão conservando a validade de ϕ_1. Sem essa restrição, a segunda revisão pode desfazer modificações realizadas pela primeira que fossem cruciais a satisfação da fórmula.

Função Revisão$_{\vee}((M, s_0), \phi_1 \vee \phi_2)$

Entrada: (M, s_0), ϕ_1 e ϕ_1, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \not\models \phi_1 \vee \phi_2$.
Saída: W tal que $\forall (M', s'_0) \in W$, $(M', s'_0) \models \phi_1 \vee \phi_2$.

1 $W \leftarrow \emptyset$
2 $W \leftarrow W \cup$ Revisão$_{\text{CTL}}(\{(M, s_0)\}, \phi_1)$
3 $W \leftarrow W \cup$ Revisão$_{\text{CTL}}(\{(M, s_0)\}, \phi_2)$
4 **retorna** W

A função Revisão$_{\vee}$ é a mais simples de todas as funções apresentadas. Nesta função realizamos para cada fórmula da disjunção modificações independentes no modelo de entrada e então combinamos os resultados por uma união simples.

[1] Se $L(s_0) = \{\#\}$, $\text{diff}(L(s_0), P)$ é mínimo se e somente se $\forall P' \subset P$, $P' \not\models \phi$

Função Revisão$_\neg$((M, s_0), $\neg\phi$)

Entrada: (M, s_0) e ϕ, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \vDash \phi$.
Saída: W tal que $\forall (M', s_0') \in W$, $(M', s_0') \nvDash \phi$.

1 **caso**
2 | ϕ seja uma fórmula proposicional: **retorna** *Revisão*$_{prop}((M, s_0), \neg\phi)$
3 | ϕ seja do tipo $\phi_1 \vee \phi_2$: **retorna** *Revisão*$_\wedge((M, s_0), \neg\phi_1 \wedge \neg\phi_2)$
4 | ϕ seja do tipo $\phi_1 \wedge \phi_2$: **retorna** *Revisão*$_\vee((M, s_0), \neg\phi_1 \vee \neg\phi_2)$
5 | ϕ seja do tipo $\mathrm{EX}\,\phi_1$: **retorna** *Revisão*$_{\neg EX}((M, s_0), \mathrm{EX}\,\phi_1)$
6 | ϕ seja do tipo $\mathrm{AF}\,\phi_1$: **retorna** *Revisão*$_{\neg AF}((M, s_0), \mathrm{AF}\,\phi_1)$
7 | ϕ seja do tipo $\mathrm{E}[\,\phi_1\,\mathrm{U}\,\phi_2\,]$: **retorna** *Revisão*$_{\neg EU}((M, s_0), \mathrm{E}[\,\phi_1\,\mathrm{U}\,\phi_2\,])$
8 **fim**

Para tratar a negação decidimos especificar um conjunto especial de funções. Isso porque buscamos modelos que não satisfazem ϕ e que sejam obtidos por modificações mínimas no modelo de entrada. O papel da função Revisão$_\neg$ é chamar subfunções de negação apropriadas, considerando a estrutura da fórmula ϕ.

Função Revisão$_{\neg EX}$((M, s_0), $\mathrm{EX}\,\phi$)

Entrada: (M, s_0) e $\mathrm{EX}\,\phi$, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \vDash \mathrm{EX}\,\phi$.
Saída: W tal que $\forall (M', s_0') \in W$, $(M', s_0') \nvDash \mathrm{EX}\,\phi$.

1 $W \leftarrow (M, s_0)$
2 **enquanto** $\exists (M', s_0') \in W$ *tal que* $(M', s_0') \vDash \mathrm{EX}\,\phi$ **faça**
3 | $W \leftarrow W - \{(M', s_0')\}$
4 | Selecione $s_\star$ tal que $(s_0', s_\star) \in R'$ e $(M', s_\star) \vDash \phi$
5 | $W \leftarrow W \cup$ Revisão$_{\mathrm{CTL}}(\{(M', s_\star)\}, \neg\phi)$
6 | Aplique PU2 para criar $M'' = (S'', R'', L'')$:
7 | $S'' = S'$, $R'' = R' - \{(s_0', s_\star)\}$, $\forall s' \in S', L''(s') = L'(s')$.
8 | $W \leftarrow W \cup \{(M'', s_0'')\}$
9 **fim**
10 **retorna** W

A função Revisão$_{\neg EX}$ modifica (M, s_0) de modo a eliminar a existência de estados vizinhos à s_0 que satisfaçam ϕ. Dois tipos de mudanças são realizadas: modificamos os sucessores de s_0 para que satisfaçam $\neg\phi$ ou eliminamos as transições que partem de s_0 para estados que satisfazem ϕ. Em ambos os casos buscamos modificações mínimas de (M, s_0), seja alterando estados consistentes com a fórmula de entrada, seja eliminando a alcançabilidade destes.

Função Revisão$_{\neg AF}((M, s_0)$, AF $\phi)$

Entrada: (M, s_0) e AF ϕ, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \vDash$ AF ϕ.
Saída: W tal que $\forall (M', s'_0) \in W$, $(M', s'_0) \nvDash$ AF ϕ.

1 $W \leftarrow (M, s_0)$
2 **se** $(M, s_0) \vDash \phi$ **então**
3 $\quad W' \leftarrow$ Revisão$_{\text{CTL}}(\{(M, s_0)\}, \neg\phi)$
4 $\quad W \leftarrow$ Revisão$_{\text{CTL}}(W', \neg$AF $\phi)$
5 **senão**
6 $\quad \mathcal{S}_\pi = \{S_\pi \mid \exists \pi = [s_0, s_1, ...]$ em M tal que $S_\pi \subseteq \pi$, $\forall s \in S_\pi$, $(M, s) \vDash \phi$
7 $\quad\quad$ e $\forall s' \in \pi - S_\pi$, $(M, s') \nvDash \phi\}$
8 $\quad \mathcal{S}_{min} = \{S_\pi \mid S_\pi \in \mathcal{S}_\pi$ e é $\subseteq$-mínimo em $\mathcal{S}_\pi\}$
9 $\quad$ **para todo** $S_{min} \in \mathcal{S}_{min}$ **faça**
10 $\quad\quad W' \leftarrow \{(M, s_0)\}$
11 $\quad\quad$ **enquanto** $\exists (M', s'_0) \in W'$ *e* $\exists s' \in S_{min}$, *tal que* $(M', s') \vDash \phi$ **faça**
12 $\quad\quad\quad W' \leftarrow W' - \{(M', s'_0)\}$
13 $\quad\quad\quad W' \leftarrow W' \cup$ Revisão$_{\text{CTL}}(\{(M', s')\}, \neg\phi)$
14 $\quad\quad$ **fim**
15 $\quad\quad W \leftarrow W \cup W'$
16 $\quad$ **fim**
17 $\quad S_\star = \{s \mid s \in S$ tal que $\exists \pi = [s_0, s_1, ..., s, ...]$ em M, $(M, s) \nvDash \phi$ e $\forall s_i < s$, $(M, s_i) \nvDash \phi\}$
18 $\quad$ **para todo** $s_\star \in S_\star$ **faça**
19 $\quad\quad$ **para todo** $s \in S$ *tal que* $(M, s) \nvDash$ AF ϕ **faça**
20 $\quad\quad\quad$ Aplique PU1 para criar $M' = (S', R', L')$:
21 $\quad\quad\quad\quad S' = S$, $R' = R \cup \{(s_\star, s)\}$, $\forall s \in S, L'(s) = L(s)$.
22 $\quad\quad\quad W \leftarrow W \cup \{M'\}$
23 $\quad\quad$ **fim**
24 $\quad\quad$ Aplique PU4 e PU1 para criar um novo modelo $M' = (S', R', L')$:
25 $\quad\quad\quad S' = S \cup \{(s^*)\}$, $R' = R \cup \{(s_\star, s^*)\}$, $\forall s \in S$, $L'(s) = L(s)$,
26 $\quad\quad\quad L'(s^*) = \{\#\}$.
27 $\quad\quad W \leftarrow W \cup$ Revisão$_{\text{CTL}}(\{(M', s^*)\}, \neg\phi)$
28 $\quad$ **fim**
29 **fim**
30 **retorna** W

A função Revisão$_{\neg AF}$ trata das modificações em (M, s_0) para que a fórmula AF ϕ não seja satisfeita. Primeiro, sempre que $(M, s_0) \vDash \phi$, AF ϕ é satisfeita. Assim, se ϕ vale em s_0, alteramos o modelo para falsificar ϕ e reanalisamos os modelos resultantes. Caso $(M, s_0) \nvDash \phi$, selecionamos os conjuntos de estados que pertencem a um mesmo caminho e que satisfazem ϕ. Denominamos esse conjunto de S_π. Guardamos então em S_{min} os conjuntos minimais de S_π. Cada elemento de S_{min} representa um conjunto mínimo de estados que, caso ϕ não fosse válido neles, existiria um caminho em M onde ϕ nunca seria válido, e portanto AF ϕ também não seria válido. Geramos então essas possíveis modificações nas linhas 9-16. Em seguida, buscamos construir caminhos inconsistentes com AF ϕ, sem que alteremos estados já existentes. Selecionamos os segmentos de caminhos iniciados em s_0, onde ϕ nunca vale. Guardamos estes estados no conjunto $S_\star$. Para cada um desses estados, ou os ligamos a estados onde AF ϕ não é valido, ou os ligamos a um novo estado onde vale $\neg\phi$, criando assim um caminho inconsistente com AF ϕ. Em todos os casos realizamos apenas mudanças estritamente necessárias a falsificação de AF ϕ no modelo de entrada.

Função Revisão$_{\neg EU}$((M, s_0), $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$)

Entrada: (M, s_0) e $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \vDash \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$.
Saída: W tal que $\forall (M', s'_0) \in W$, $(M', s'_0) \nvDash \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$.

1 $W \leftarrow (M, s_0)$
2 **se** $(M, s_0) \vDash \phi_2$ **então**
3 $\quad W' \leftarrow \text{Revisão}_{\text{CTL}}(\{(M, s_0)\}, \neg\phi_2)$
4 $\quad W \leftarrow \text{Revisão}_{\text{CTL}}(W', \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,])$
5 **senão**
6 $\quad \mathcal{S}_\pi = \{S_\pi \mid S_\pi =$ estados s_i de $\pi = [s_0, s_1, ..., s_j, ...]$ em M tal que $(M, s_j) \vDash \phi_2$
7 $\quad\quad$ e $\forall s_i < s_j$, $(M, s_i) \vDash \phi_1 \wedge \neg\phi_2\}$
8 $\quad \mathcal{S}_\star = \{S_\star \mid \forall S_\pi \in \mathcal{S}_\pi,\ S_\star \cap S_\pi \neq \emptyset$ e $\forall S'_\star \subset S_\star,\ \exists S'_\pi \in \mathcal{S}_\pi$ tal que $S'_\star \cap S'_\pi = \emptyset\}$
9 $\quad$ **para todo** $S_\star \in \mathcal{S}_\star$ **faça**
10 $\quad\quad W' \leftarrow \{(M, s_0)\}$
11 $\quad\quad$ **enquanto** $\exists (M', s'_0) \in W'$ *e* $\exists s_\star \in S_\star$, *tal que* $(M', s_\star) \vDash \phi_1$ **faça**
12 $\quad\quad\quad W' \leftarrow W' - \{(M', s'_0)\}$
13 $\quad\quad\quad W' \leftarrow W' \cup \text{Revisão}_{\text{CTL}}(\{(M', s_\star)\}, \neg\phi_1)$
14 $\quad\quad$ **fim**
15 $\quad\quad W \leftarrow W \cup W'$
16 $\quad$ **fim**
17 $\quad$ Aplique PU2 para criar $M' = (S', R', L')$: $S' = S$,
18 $\quad\quad R' = R - \{(s_0, s_*) \mid (s_0, s_*) \in R$ e $(M, s_*) \vDash \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]\}$,
19 $\quad\quad \forall s \in S, L'(s) = L(s)$.
20 $\quad W \leftarrow W \cup \{(M', s'_0)\}$
21 **fim**
22 **retorna** W

Na função Revisão$_{\neg EU}$ buscamos modificações em (M, s_0) para falsificar a fórmula $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ no estado s_0 do modelo M. Primeiro, sempre que $(M, s_0) \vDash \phi_2$, $(M, s_0) \vDash \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$. Assim, caso $(M, s_0) \vDash \phi_2$, alteramos o modelo para que ϕ não seja valido em s_0 e então reanalisamos os modelos resultantes. Caso $(M, s_0) \nvDash \phi$, primeiro selecionamos em S_π os estados de segmentos de caminho que validam $(M, s_0) \vDash \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$. Depois selecionamos conjuntos minimais de estados que contenham um representante de cada conjunto em S_π e os guardamos em $\mathcal{S}_\star$. Cada elemento de $\mathcal{S}_\star$ representa um conjunto mínimo de estados do modelo que devem ser modificados para que $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ seja falsificado. Estas modificações são realizada nas linhas 9-16. Por último, outra modificação possível é retirar a alcançabilidade desses segmentos, o que fazemos na linha 17. Encerrando assim nossa descrição das funções de negação.

No Capítulo 8 abordamos questões sobre este algoritmo. Discutimos suas semelhanças e diferenças em relação a proposta de Zhang e Ding, mostramos limitações de nossa abordagem e evidenciamos o quão complexo é especificar formalmente um método para revisão de modelos.

Capítulo 8

Discussão

A principal discussão deste capítulo vem do fato de não podermos garantir que o resultado do operador $\circ_c$ possa ser obtido, mesmo que parcialmente, por nosso algoritmo de revisão de modelos. Ironicamente, vem exatamente desta afirmação nossa principal contribuição neste tópico: defendemos que não é promissor desenvolver uma abordagem para revisão seguindo o algoritmo de modificações mínimas proposto por Zhang e Ding [ZD08].

A princípio, planejamos utilizar diretamente as funções de modificação de Zhang e Ding, todavia atestamos problemas com estas funções, o que comprometeria nosso algoritmo de revisão. Destes problemas, destacamos a forte dependência que o método deles possui de escolhas corretas em passos não-determinísticos. Essa sensibilidade à escolha compromete até mesmo a parada do algoritmo. Uma boa ilustração desse problema pode ser vista na função Atualização$_{EU}$, retirada de [ZD08].

Função Atualização$_{EU}$((M, s_0), $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$)

Entrada: (M, s_0) e $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$, onde $M = (S, R, L)$, $s_0 \in S$ e $(M, s_0) \not\models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$.
Saída: (M', s'_0), onde $M' = (S', R', L')$, $s'_0 \in S'$ e $(M', s'_0) \not\models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$.

1 **se** $(M, s_0) \not\models \phi_1$ **então**
2 | $(M', s'_0) =$ Atualização$_{CTL}((M, s_0), \phi_1)$
3 **senão**
4 | Faça (a) ou (b):
5 | **(a): se** $(M, s_0) \models \phi_1$ *e existe um caminho* $\pi = [s^*, \cdots]\,(s_0 \neq s^*)$ *tal que* $(M, s^*) \models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ **então**
6 | | Aplique PU1 para criar um novo modelo $M' = (S', R', L')$, tal que
7 | | $S' = S$, $R' = R \cup \{(s_0, s^*)\}$, $\forall s \in S$, $L'(s) = L(s)$.
8 | **fim**
9 | **(b):** Selecione uma caminho $\pi = [s_0, \cdots, s_i, \cdots, s_j, \cdots]$
10 | **se** $\forall s\ s_0 < s < s_i$, $(M, s) \models \phi_1$, $(M, s_j) \models \phi_2$, *mas* $\forall s'\ s_{i+1} < s' < s_{j-1}$, $(M, s') \not\models \phi_1 \vee \phi_2$ **então**
11 | | Aplique PU1 para criar um novo modelo $M' = (S', R', L')$, tal que
12 | | $S' = S$, $R' = R \cup \{(s_i, s_j)\}$, $\forall s \in S$, $L'(s) = L(s)$.
13 | **fim**
14 | **se** $\forall s\ s < s_i$, $(M, s) \models \phi_1$, $\forall s'\ s_{i+1} < s'$, $(M, s') \not\models \phi_1 \vee \phi_2$ **então**
15 | | Aplique PU4 para criar um novo modelo $M' = (S', R', L')$, tal que
16 | | $S' = S \cup \{s^*\}$, $R' = R \cup \{(s_{i-1}, s^*), (s^*, s_i)\}$, $\forall s \in S$, $L'(s) = L(s)$ e $L'(s^*)$ é definido tal que $(M', s^*) \models \phi_2$.
17 | **fim**
18 **fim**
19 **se** $(M', s'_0) \models \mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$ **retorna** (M', s'_0)
20 **senão** Atualização$_{EU}$((M', s'_0), $\mathrm{E}[\,\phi_1 \,\mathrm{U}\, \phi_2\,]$)

Observe que na escolha (b) (linha 9), para que a modificação no modelo seja realizada corretamente, o algoritmo depende fortemente de uma escolha adequada de π. Se por ventura na linha 4 sempre escolhêssemos executar (b), mas nunca selecionássemos um π que casasse com um dos condicionais das linhas 10 e 14, o algoritmo ficaria preso nesta função, realizando infinitas chamadas recursivas. O problema está no fato que temos infinitas possibilidades de caminhos mesmo em estruturas de Kripke simples, e mesmo que sempre selecionemos um π diferente dos anteriores, não temos garantia que em algum momento escolheremos um π adequado.

Concluímos que não poderíamos utilizar as funções de Zhang e Ding diretamente. Especificamos novas funções de modificações modelo-a-modelo, buscando sanar alguns dos problemas apresentados por suas funções. O primeiro passo foi reformulamos a abordagem de Zhang e Ding de modo a torna-la determinística. Fizemos com que todas as escolhas e iterações se deem sobre conjuntos sabidamente finitos, como por exemplo o conjunto de estados de um modelo. Contudo, vale ressaltar que conservamos plenamente a ideia original de sua abordagem: realizar modificações no modelo com o mínimo de operações primitivas a cada chamada de uma função. Vemos assim nossas funções como um aprimoramento das funções de modificação de modelos propostas por Zhang e Ding.

Contudo, mesmo sanando os problemas encontrados no método de Zhang e Ding, duas complicações surgem ao utilizar modificações modelo-a-modelo no contexto da revisão de crenças: primeiro, por não podermos utilizar como entrada todos os modelos de uma especificação; segundo, por não permitir a geração de todas as modificações possíveis para cada modelo de entrada.

Como vimos no capítulo anterior, assumimos a entrada como um conjunto finito de modelos. Isso é necessário para que possamos garantir a parada. Porém, ao fazer isso arriscamos perder informações necessárias para realizar corretamente a revisão. Com isso, não é possível garantir que, para $W \subseteq Mod(\psi)$, temos $\text{Revisão}_{\text{CTL}}(Mod(\psi), \phi) \subseteq Mod(\psi \circ_c \phi)$. Zhang e Ding não encontrariam esse problema no contexto da atualização de modelos, porque a restrição da entrada resultaria em um conjunto ainda contido em $Mod(\psi \diamond_c \phi)$.

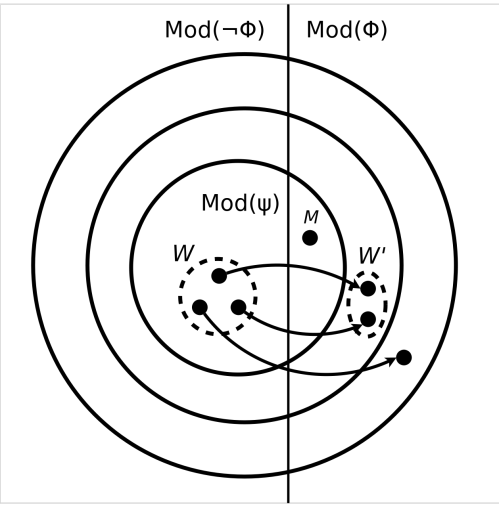

Figura 8.1: *Problema de utilizar uma entrada limitada.*

Tomemos a Figura 8.1 como exemplo para este problema. Suponha que ψ e ϕ são duas fórmulas CTL que possuem um único modelo em comum, M. Suponha também que W é um conjunto, finito, tal que $W \subseteq Mod(\psi)$, mas que $M \notin W$. Ao aplicarmos as modificações minimais a cada modelo de W, de modo que estes satisfaçam a fórmula ϕ, obtemos como resultado o conjunto $W' \subseteq Mod(\psi \diamond_c \phi)$. Porém, mesmo excluindo de W' os modelos não minimais segundo a ordem $\leqslant_W$ (o processo realizado por nosso algoritmo), podemos ver que $W' \not\subseteq Mod(\psi \circ_c \phi) = \{M\}$. O ideal é que a entrada do algoritmo corresponda a todos os modelos de uma especificação, porém isso só é possível se tal especificação possuísse um conjunto finito de modelos.

Contudo, a importância do nosso algoritmo não se restringe às especificações com modelos finitos. O conjunto de entrada W determina o quanto de informação da especificação será levada

em conta na resposta. Quanto maior o conjunto W, maior a proximidade entre o resultado e a especificação. Cabe ao projetista fornecer o conjunto de modelos que ele considere relevante para a revisão.

O segundo problema ao utilizar modificações modelo-a-modelo na revisão é que, mesmo que assumamos $W = Mod(\psi)$ finito, se não gerarmos todas as modificações possíveis para cada $M \in W$, não podemos garantir que o resultado da revisão seja subconjunto de $Mod(\psi \circ_c \phi)$. Considere a Figura 8.2, onde as setas tracejadas indicam transformações que não realizamos. Veja que não geramos exatamente aqueles modelos que deveriam pertencer ao resultado da revisão, assim o refinamento do algoritmo não excluirá os modelos gerados por não detectar modelos mais próximos do que aqueles gerados.

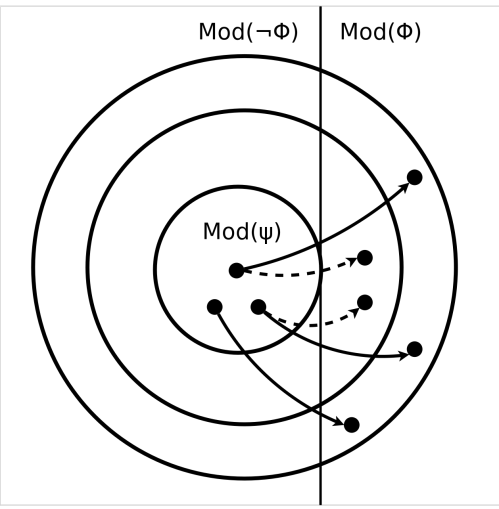

Figura 8.2: *Problema de não gerar todas as modificações minimais.*

O problema ocorre porque a revisão analisa a semelhança entre modelos sob um ponto de vista global, isto é, são comparadas as proximidades que cada um dos modelos possui do conjunto de entrada. Caso deixemos de gerar exatamente aqueles mais próximos à entrada, o refinamento que seleciona os mínimos não surte efeito. O mesmo problema não ocorre na atualização de modelos, pois a análise de semelhança é feita localmente, levando em conta um único modelo e não todo o conjunto de entrada.

Na busca das correções nos modelos, nosso algoritmo realiza uma ampla gama de modificações, mas não é capaz de encontrar todas as possíveis correções. As subfunções do algoritmo capturam as ideias principais das modificações mínimas, mas não consideram uma mescla dessas modificações. Para entender essa afirmação, tomemos o exemplo da Função Revisão$_{AF}$. Para satisfazer uma fórmula AF as duas maneiras de modificar um modelo são: corrigir estados inconsistentes ou desconectar caminhos inconsistentes. Na função Revisão$_{AF}$, primeiro utilizamos todas as correções em estados possíveis e só depois geramos um modelo desconectando os caminhos inconsistentes. Desse modo, não conseguimos gerar os modelos admissíveis que só são alcançados pela mescla dos dois tipos de modificações.

Gerar todas as correções é uma tarefa extremamente difícil, mas vital à revisão de modelos. Nem mesmo o algoritmo não-determinístico de Zhang e Ding seria capaz de gerar todas as modificações de um modelo. Trabalhos futuros incluem buscar formas de garantir que todos os modelos sejam explorados, bem como caracterizar a qualidade das soluções geradas com o método proposto neste trabalho.

Capítulo 9

Conclusões

Verificadores de modelos são ferramentas importantes no desenvolvimento de sistemas, mas que não possuem mecanismos que auxiliam na correção de inconsistências. Diversos autores exploraram a integração entre técnicas de Inteligência Artificial e verificação de modelos [BEGL99, SW07, FW08, ZD08]. Neste trabalho apresentamos um estudo formal de revisão de crenças como técnica para o reparo de modelos inconsistentes. Mostramos que revisão de crença pode ser uma poderosa ferramenta de auxilio à modelagem formal de sistemas computacionais.

Argumentamos que a abordagem de Zhang e Ding [ZD08] não é adequada para todas as situações de reparo de sistemas. Mostramos que, nos casos em que mudanças significam correções e não adaptações, nossa abordagem baseada em revisão de crenças é mais apropriada que uma baseada em atualização. Mostramos que, nestas situações, nosso cálculo de novos modelos conserva um maior grau de coerência com a especificação original do sistema.

A existência de uma revisão AGM é assegurada apenas para lógicas compactas e monotônicas, porém CTL não satisfaz esta primeira propriedade. Não podemos aplicar diretamente a teoria existente para a revisão de modelos CTL.

Apresentamos uma primeira abordagem de integração de revisão de crenças e verificação de modelos, denominada BrNuSMV [Sou07, SW07]. Mostramos que apesar de uma aplicação prática, essa abordagem possui algumas limitações. Buscamos então preencher algumas das lacunas apresentada por essa.

Propusemos um arcabouço formal para revisão de modelos CTL. Primeiro, definimos o critério de ordenação entre dois modelos CTL (baseado na distância deste a um dado conjuntos de modelos). Formalizamos então nosso operador de revisão, onde, baseado na ordenação definida, estabelecemos o critério de minimalidade para a escolha de modelos consistentes. Mostramos que nosso operador de revisão obedece os postulados de revisão AGM, conforme reformulação de [KM89].

Desenvolvemos um algoritmo formal para revisão de modelos CTL. Dados modelos que satisfazem a especificação formal do sistema e uma propriedade expressa por uma fórmula CTL, nosso algoritmo gera um conjunto de modelos que satisfazem essa propriedade e cujas distâncias dos modelos da especificação original são mínimas. Os elementos gerados como resposta podem ser vistos como possíveis correções à modelagem inconsistente.

Por fim, discutimos questões relacionadas ao algoritmo desenvolvido e argumentamos que algoritmos baseados na abordagem de [ZD08], como o nosso, possuem diversas complicações quando aplicados no contexto de revisão de crenças, o que mostra a necessidade de desenvolver uma estratégia diferente e específica ao problema da revisão de modelos.

9.1 Trabalhos Futuros

Teorema de representação. O Teorema 8 não assegura que nosso operador de revisão possua a completa racionalidade de um operador de revisão AGM. Isso porque para caracterizar uma revisão AGM (*teorema de representação*) devemos mostrar que nossa construção obedece todos os postulados e que qualquer construção que satisfaça estes postulados pode ser definida segundo nossa

construção. Provamos no Teorema 8 apenas a primeira parte do teorema de representação, ficando para um trabalho futuro a segunda parte da prova.

Para entender a importância do teorema de representação, suponha que os postulados de revisão se resumem aos postulados (K*1) e (K*2) e que definimos nossa revisão como $K * \alpha = \{\alpha\}$. Nosso operador obedece os postulados (K*1) e (K*2), mas realiza o processo de um modo extremo. Por outro lado, existe uma gama de construções que satisfazem (K*1) e (K*2) e que conservam mais informações do conjunto K, desse modo nossa construção não constitui uma revisão adequada. A completa demonstração do teorema de representação garante que nossa construção possui a exata racionalidade exigida pela Teoria AGM.

Revisão em lógicas não compactas. A compacidade desempenha um papel importante em diversas demonstrações em revisão de crenças. Na Seção 6.2 apresentamos um exemplo que mostra a não compacidade da lógica CTL, no entanto somos intuitivamente capazes de determinar o resultado da revisão neste exemplo. É possível que a compacidade seja uma exigência mais forte que o necessário para muitas demonstrações. Caso isso seja verdade, investigar este aspecto pode levar ao elo entre a teoria clássica de revisão de crenças e a revisão em lógicas não compactas.

Problema da explosão de modelos. Um dos problemas para computar a revisão de modelos CTL é que uma única modificação pode exigir a análise de todos os conjuntos de estados possíveis. Com isso, buscar todas as possibilidades de modificação faz com que o conjunto de modelos candidatos cresça exponencialmente. A mesma preocupação é apresentada por Ding [Din07], onde se buscou abordagens para contornar o problema da explosão de modelos. Apesar de apresentar alguns métodos, o autor afirma que suas soluções possuem limitações e que necessitam de mais estudos no futuro.

Revisão de modelos LTL. A Lógica Temporal de Tempo Linear (LTL) é outra lógica bastante utilizada em verificação de modelos. A LTL assemelha-se à CTL, exceto no fato de representar o futuro como um único caminho. Um importante passo seria estender os conceitos de revisão de modelos CTL para lógica LTL, visto que ambas abordagens compartilham os mesmos problemas quando relacionadas a teoria clássica de revisão de crenças.

Implementação do revisor de modelos. Construir uma ferramenta para revisão de modelos CTL, baseada no arcabouço proposto neste trabalho, representaria uma importante contribuição prática para a modelagem formal de sistemas. Essa ferramenta auxiliaria na correção de modelagens inconsistentes sem que o projetista de sistemas precisasse conhecer conceitos intrínsecos à revisão de crenças.

Apêndice A

Propriedades Lógicas

A seguinte lista foi compilada em [Was00] e é composta por diversas propriedades lógicas encontradas na literatura. Seja $\mathcal{L}$ um linguagem, B e D subconjuntos de $\mathcal{L}$, $\mathcal{C}$ um operador de consequência e Cn o operador de consequência clássico, uma lógica $\langle \mathcal{L}, \mathcal{C} \rangle$ satisfaz:

Monotonicidade se e somente se $B \subseteq D$ implica $\mathcal{C}(B) \subseteq \mathcal{C}(D)$;

Compacidade se e somente se para todo $\alpha \in \mathcal{C}(B)$ existe algum conjunto finito $D \subseteq B$ tal que $\alpha \in \mathcal{C}(D)$;

Idempotência se e somente se $\mathcal{C}(\mathcal{C}(B)) = \mathcal{C}(B)$;

Idempotência fraca se e somente se $\mathcal{C}(\mathcal{C}(B)) \subseteq \mathcal{C}(B)$;

Inclusão se e somente se $B \subseteq \mathcal{C}(B)$;

Inclusão Incorporada se e somente se $\mathcal{C}(B) \subseteq \mathcal{C}(\mathcal{C}(B))$;

Tarskianicidade se e somente se satisfaz monotonicidade, idempotência e inclusão;

Supraclassicalidade se e somente se $Cn(B) \subseteq \mathcal{C}(B)$;

Teorema da Dedução se e somente se sempre que $q \in \mathcal{C}(B \cup \{p\})$, então $p \to q \in \mathcal{C}(B)$;

Reductio ad absurdum se e somente se sempre que $\perp \in \mathcal{C}(B \cup \{\neg p\})$, então $p \in \mathcal{C}(B)$;

Falsidade se e somente se para todo p, $p \in \mathcal{C}(\{\perp\})$;

Preservação da consistência se e somente se $\perp \notin Cn(B)$ implica $\perp \notin \mathcal{C}(B)$;

Cumulatividade se e somente se $B \subseteq D \subseteq \mathcal{C}(B)$ implica $\mathcal{C}(D) = \mathcal{C}(B)$;

Distributividade se e somente se $\mathcal{C}(B) \cap \mathcal{C}(D) \subseteq \mathcal{C}(\mathcal{C}(B) \cap \mathcal{C}(D))$;

Explosividade se e somente se para todo α e β, $\beta \in \mathcal{C}(\{\alpha, \neg\alpha\})$;

Explosividade fraca se e somente se sempre que $\perp \in \mathcal{C}(B)$, então para todas as sentenças α, $\alpha \in \mathcal{C}(B)$.

Não contravenção se e somente se, para todo α, se $\neg\alpha \in \mathcal{C}(B \cup \{\alpha\})$, então $\neg\alpha \in \mathcal{C}(B)$; e

Não contravenção α-local se e somente se, se $\neg\alpha \in \mathcal{C}(B \cup \{\alpha\})$, então $\neg\alpha \in \mathcal{C}(B)$.

Referências Bibliográficas

[ADK+05] Nina Amla, Xiaoqun Du, Andreas Kuehlmann, Robert P. Kurshan, e McMillan Kenneth L. An analysis of SAT-based model checking techniques in an industrial environment. Em *Correct Hardware Design and Verification Methods*, páginas 254–268. Springer, 2005.

[AGM85] Carlos E. Alchourron, Peter Gärdenfors, e David Makinson. On the logic of theory change: Partial meet contraction and revision functions. *J. Symb. Logic*, 50(2):510–530, 1985.

[BEGL99] Francesco Buccafurri, Thomas Eiter, Georg Gottlob, e Nicola Leone. Enhancing model checking in verification by AI techniques. *Artificial Intelligence*, 112(1-2):57–104, 1999.

[CCGR99] Alessandro Cimatti, Edmund M. Clarke, Fausto Giunchiglia, e Marco Roveri. NuSMV: A new symbolic model verifier. Em *Proc. of CAV*, páginas 495–499. Springer-Verlag, 1999.

[CCK+02] Pankaj Chauhan, Edmund M. Clarke, James H. Kukula, Samir Sapra, Helmut Veith, e Dong Wang. Automated abstraction refinement for model checking large state spaces using SAT based conflict analysis. Em *Proc. of 4th International Conf. on Formal Methods in Computer-Aided Design*, páginas 33–51. Springer-Verlag, 2002.

[CES86] Edmund M. Clarke, E. Allen Emerson, e A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[CGP99] Edmund M. Clarke, Orna Grumberg, e Doron A. Peled. *Model checking*. Springer, 1999.

[Din07] Yulin Ding. *Model Update for System Modifications*. Tese de Doutorado, University of Western Sydney, 2007.

[FW08] M. Finger e Renata Wassermann. Revising specifications with CTL properties using bounded model checking. Em *Proceedings of the 19th Brazilian Symposium on Artificial Intelligence (SBIA'08)*, páginas 157–166. Springer, 2008.

[Gär88] Peter Gärdenfors. *Knowledge in flux*. MIT press, 1988.

[Gro88] Adam Grove. Two modellings for theory change. *Journal of Philosophical Logic*, 17(2):157–170, May 1988.

[GW10] Paulo T. Guerra e Renata Wassermann. Revision of CTL models. Em Angel Kuri-Morales e Guillermo Simari, editors, *Advances in Artificial Intelligence – IBERAMIA 2010*, volume 6433 of *Lecture Notes in Computer Science*, páginas 153–162. Springer Berlin / Heidelberg, 2010.

[Her98] Andreas Herzig. Logics for belief base updating. Em D. Dubois e H. Prade, editors, *Handbook of Defeasible Reasoning and Uncertainty Management*, volume Belief Change, páginas 189–231. Kluwer Academic, Dordrecht, 1998.

[HR99] Andreas Herzig e Omar Rifi. Propositional belief base update and minimal change. *Artificial Intelligence*, 115(1):107–138, 1999.

[HR04] Michael Huth e Mark Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.

[HW02] Sven O. Hansson e Renata Wassermann. Local change. *Studia Logica*, 70(1):49–76, 2002.

[KM89] Hirofumi Katsuno e Alberto O. Mendelzon. A unified view of propositional knowledge base updates. Em *Proc. of IJCAI'89*, páginas 1413–1419. Morgan Kaufmann, 1989.

[KM91] Hirofumi Katsuno e Alberto O. Mendelzon. On the difference between updating a knowledge base and revising it. Em *Proc. of KR*, páginas 387–395. Morgan Kaufmann, 1991.

[KW85] Arthur M. Keller e Marianne Winslett. On the use of an extended relational model to handle changing incomplete information. *IEEE Transactions on Software Engineering*, 11(7):620–633, 1985.

[RW06] Márcio Moretto Ribeiro e Renata Wassermann. First steps towards revising ontologies. Em *Proceedings of the Second Workshop on Ontologies and their Applications (WONTO 2006)*, 2006.

[Sou07] Thiago C. Sousa. Revisão de modelos formais de sistemas de estados finitos. Dissertação de Mestrado, Universidade de São Paulo, 2007.

[SW07] Thiago C. Sousa e Renata Wassermann. Handling inconsistencies in CTL model-checking using belief revision. Em *Proc. of the Brazilian Symposium on Formal Methods*, 2007.

[Was00] Renata Wassermann. *Resource-Bounded Belief Revision*. Tese de Doutorado, Universiteit van Amsterdam, 2000.

[Win88] Marianne Winslett. Reasoning about action using a possible models approach. Em *Proc. of AAAI*, páginas 89–93. Morgan Kaufmann, 1988.

[Win90] M.S. Winslett. *Updating logical databases*. Cambridge University Press, 1990.

[ZD08] Yan Zhang e Yulin Ding. CTL model update for system modifications. *Journal of Artificial Intelligence Research*, 31(1):113–155, 2008.