UNIVERSIDADE FEDERAL DA BAHIA

DISSERTAÇÃO DE MESTRADO

Uma Solução para o Refinamento de Modelos KMTS Baseado em Verificação de Modelos com Jogos

Jandson Santos Ribeiro Santos

Programa de Pós-Graduação em Ciência da Computação

Salvador
25 de Janeiro de 2016

JANDSON SANTOS RIBEIRO SANTOS

UMA SOLUÇÃO PARA O REFINAMENTO DE MODELOS KMTS BASEADO EM VERIFICAÇÃO DE MODELOS COM JOGOS

Esta Dissertação de Mestrado foi apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Mestre em Ciência da Computação.

Orientador: Profa. Dra. Aline Maria Santos Andrade

Salvador
25 de Janeiro de 2016

Sistema de Bibliotecas da UFBA

Santos, Jandson Santos Ribeiro.
Uma solução para o refinamento de modelos KMTS baseado em verificação de modelos com jogos / Jandson Santos Ribeiro Santos. - 2016.
94 f.: il.

Inclui apêndice.
Orientadora: Profª. Drª. Aline Maria Santos Andrade.
Dissertação (mestrado) - Universidade Federal da Bahia, Instituto de Matemática, Salvador, 2016.

1. Teoria dos jogos. 2. Modelos matemáticos. 3. Modalidade (Lógica). I. Andrade, Aline Maria Santos. II. Universidade Federal da Bahia. Instituto de Matemática. III. Título.

CDD - 519.3
CDU - 517.977.8

TERMO DE APROVAÇÃO

JANDSON SANTOS RIBEIRO SANTOS

UMA SOLUÇÃO PARA O REFINAMENTO DE MODELOS KMTS BASEADO EM VERIFICAÇÃO DE MODELOS COM JOGOS

Esta Dissertação de Mestrado foi julgada adequada à obtenção do título de Mestre em Ciência da Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação da Universidade Federal da Bahia.

Salvador, 25 de Janeiro de 2015

Profa. Dra. Aline Maria Santos Andrade
UFBA

Profa. Dra. Renata Wassermann
IME/USP

Prof. Dr. Edward Hermann Haeusler
PUC–Rio

AGRADECIMENTOS

Agradeço à professora Aline Andrade, minha orientadora, por ter me orientado durante o mestrado. Agradeço também pela paciência, dedicação, apoio e incentivo ao longo de todo este tempo.

Não posso esquecer também dos familiares e amigos que me apoiaram durante o período em que realizei o mestrado.

Agradeço à Fundação de Amparo a Pesquisa do Estada da Bahia (FAPESB) pela bolsa concedida a qual tornou este trabalho possível.

“ Como todas as artes, a Ciência da Dedução e da Análise só pode ser adquirida após um aprendizado demorado e paciente, mas a vida não é suficientemente longa para permitir que algum mortal atinja a perfeição máxima nesse campo. Antes de se concentrar nos aspectos morais e mentais do assunto que apresenta as maiores dificuldades, o pesquisador deve começar pelo domínio dos problemas mais elementares. [...] Por mais infantil que esse exercício possa parecer, ele aguça as faculdades de observação, ensinando para onde se deve olhar e o que procurar.”

—ARTHUR CONAN DOYLE (Sherlock Holmes: Um Estudo em Vermelho)

RESUMO

A revisão de modelos é uma técnica baseada na teoria de revisão de crenças, que tem como princípio modificar minimamente os modelos de forma a satisfazer uma dada propriedade. A revisão de modelos pode ser combinada com verificação de modelos para determinar as mudanças estruturais minimais sobre modelos de Kripke com base no resultado da verificação.

As estruturas modais de Kripke (KMTS) são extensões das estruturas de Kripke com modalidades que permitem expressar informações incompletas explicitamente. Um KMTS pode ser interpretado como um conjunto de estruturas de Kripke, e a técnica de verificação de modelos pode ser utilizada para verificar se um KMTS atende à uma determinada especificação, devendo retornar os seguintes valores: verdadeiro quando todas as estruturas de Kripke representadas pelo KMTS satisfazem a especificação, falso quando nenhuma das estruturas de Kripke satisfaz a especificação, ou indefinido quando algumas das estruturas satisfazem e outras não. Uma das contribuições deste trabalho é a proposta de um verificador de modelos para KMTS considerando esta semântica utilizando CTL como linguagem de especificação de propriedades.

A revisão de um KMTS ocorre quando a verificação de modelos resulta em falso ou indefinido. No caso de indefinido, o KMTS deve ser modificado para representar apenas os modelos de Kripke que satisfazem a propriedade requerida. Esta etapa da revisão é chamada de refinamento. Neste caso, de acordo com a revisão de modelos, deve-se aplicar mudanças estruturais minimais sobre o KMTS para obter os modelos resultantes do refinamento. Outra contribuição deste trabalho está em apresentar uma solução para este problema, através de algoritmos propostos com base em um grafo de testemunhas, definido nesta dissertação, que abstrai informações do grafo de um verificador de modelos baseado em jogos de forma a obter apenas os modelos gerados por mudanças minimais sobre o KMTS.

Palavras-chave: Revisão de Modelos, Refinamento de KMTS, CTL, Verificação de modelos com Jogos.

ABSTRACT

The model revision is a technique based on the belief revision theory which aims to change minimally a model in order to satisfy a desired property. It is possible to combine the model revision with model checking techniques to determine the minimal structural changes on Kripke models according to the model checking result.

Kripke Modal Transition System (KMTS) are Kripke structures with modalities that allow to express explicitly incomplete information. A KMTS can be interpreted as a set of Kripke structure and the model checking techniques can be used to verify whether a KMTS satisfies a required specification which should return the following values: true when all the Kripke structure expended by a KMTS satisfy the specification, false when no Kripke structure satisfy the specification, or undefined when some of the structures satisfy the specification and others do not. One of the contributions of this work is the proposal of a model checking for KMTSs as a set of Kripke structures considering properties specified in the Computation Tree Logic (CTL).

The revision of a KMTS occurs when the model checking results in false or undefined. In the case of undefined, the KMTS must be modified in order to represent only Kripke models that satisfy the required property. This revision step is called refinement. In this case, according to model revision, minimal structural changes must be applied over the KMTS to get the Kripke model satisfying the desired property. Another contribution of this paper is to present a solution to this problem through algorithms proposed based on a witness graph, defined in this work, which abstracts the graph of a model checker game in order to get just the generated models by minimal changes over a KMTS.

Keywords: Model Revision, KMTS Refinement, CTL, Model Checking Game.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

Capítulo 1

INTRODUÇÃO

A verificação de propriedades sobre modelos de sistemas pode ser feita de forma automática, com o auxílio de uma técnica formal proposta por (CLARKE; GRUMBERG; PELED, 1999) denominada verificação de modelos. Esta técnica consiste basicamente em receber um modelo de um sistema e a especificação de uma propriedade utilizando alguma lógica formal, como a lógica de árvore computacional (CTL - *Computing Tree Logic*), e verificar se o modelo satisfaz a propriedade. A verificação de modelos é definida sobre as estruturas de Kripke que são modelos, entre outras lógicas, da lógica de árvore computacional.

As estruturas de transições modais de Kripke (KMTS)(HUTH; JAGADEESAN; SCHMIDT, 2001) são extensões das estruturas de Kripke capazes de representar de forma explícita informação parcial, através de indefinições em seus estados e transições. Elas contêm dois tipos de transições: transições *must* que indicam a ocorrência das mesmas no modelo final do sistema e transições *may* que indicam a possibilidade da ocorrência destas transições no modelo final.

Um KMTS pode ser interpretado como um conjunto de estruturas de Kripke, como proposto em (GUERRA; ANDRADE; WASSERMANN, 2013). Nessa interpretação, as transições *may* e átomos indeterminados nos estados podem ser interpretados como presença ou ausência dos mesmos em estruturas de Kripke diferentes. Assim, a presença de uma única transição *may* pode levar um KMTS à representação de duas estruturas de Kripke: uma com a transição em questão presente e uma outra com a transição ausente. O mesmo vale para propriedades indefinidas nos estados do modelo. Desta forma, um KMTS pode ser expandido em um conjunto de estruturas de Kripke.

Ao interpretarmos um KMTS como um conjunto de estruturas de Kripke, a verificação de uma propriedade φ sobre o mesmo tem três resultados possíveis: verdadeiro ($\top$) se todos os modelos representados pelo KMTS satisfazem a propriedade, falso (F) se nenhum dos modelos representados satisfazem a propriedade, ou ainda indefinido ($\bot$) caso existam modelos que satisfazem a propriedade e outros que não a satisfazem .

Se a verificação de modelos resulta em falso ou indefinido, um contra-exemplo que corresponde a um caminho do modelo pode ser fornecido ao projetista a fim de auxiliá-lo no processo de correção que normalmente é feito de forma manual. No entanto, a dificuldade da correção é diretamente proporcional ao tamanho do modelo. Dessa forma, o ideal é um procedimento automático de correção. Esta tarefa pode ser automatizada com o auxílio da revisão de modelos.

1.1 REVISÃO DE MODELOS

A revisão de modelos baseia-se nos conceitos de revisão de crenças (ALCHOURRON; GÄRDENFORS; MAKINSON, 1985) (GÄRDENFORS, 1988) e está associada ao problema de incorporar novas informações que podem ser inconsistentes com o modelo, implicando em mudanças que devem ser aplicadas sobre o mesmo a fim de torná-lo consistente com as novas informações. A mudança na operação de revisão de modelos deve ser mínima, ou seja, deve preservar o máximo possível de informação do modelo original.

No caso do KMTS, quando a verificação de modelos resulta em F ou $\perp$, torna-se necessário repará-lo, a fim de se obter os modelos que satisfazem a propriedade desejada. No primeiro caso, o KMTS deve ser reparado de forma que todas as estruturas de Kripke por ele representadas satisfaçam a propriedade; no segundo caso, deve-se filtrar do conjunto de estruturas de Kripke representadas pelo KMTS os modelos que satisfazem esta propriedade (GUERRA; ANDRADE; WASSERMANN, 2013).

Este trabalho insere-se no contexto de revisão de modelos KMTS, mais especificamente ,quando a verificação de modelos KMTS resulta em indefinido, que é chamada refinamento de modelos KMTS.

1.2 ALGORITMOS PARA REFINAMENTO DE MODELOS

O trabalho (GUERRA; ANDRADE; WASSERMANN, 2013) propõe algoritmos para o refinamento de modelos KMTSs baseado na abordagem de revisão de modelos. Os autores consideram a verificação de modelos com jogos para a lógica CTL com três valores proposta em (GRUMBERG et al., 2007) e (SHOHAM; GRUMBERG, 2007). O algoritmo de verificação de modelos informa as possíveis causas do resultado falso ou indefinido da verificação que são utilizadas para guiar o refinamento. O refinamento é realizado através da combinação das mudanças estruturais definidas pelas causas de falhas, chamadas de testemunhas de falhas, e retorna os modelos que satisfazem a propriedade desejada. Contudo, um filtro deve ser aplicado sobre o conjunto de todas as mudanças geradas a fim de selecionar as minimais, o que é bastante ineficiente devido à capacidade do KMTS em representar modelos de Kripke, que é exponencial em relação ao números de indeterminações.

1.3 PROPOSTA

Nesse trabalho, propomos um verificador de modelos que corresponde à semântica de um KMTS interpretado como um conjunto de estruturas de Kripke e uma solução para o refinamento de modelos KMTS de acordo com esta semântica. A nossa solução de refi-

namento consiste em algoritmos definidos sobre uma estrutura proposta neste trabalho a qual chamamos de grafo de testemunhas. Os algoritmos definidos sobre o grafo de testemunhas consistem basicamente em determinar as mudanças minimais de um vértice através da combinação e seleção das mudanças minimais dos vértices filhos. Esta abordagem reduz o número de comparações em cada vértice na busca das mudanças minimais, visto que mudanças desnecessárias são descartadas nos processos dos vértices filhos. O problema de explosão de estados é inerente à verificação de modelos, e o grafo de testemunhas permite que o refinamento seja realizado sob um espaço de busca menor. Outra vantagem promovida por esta estrutura é o fato de não precisarmos aplicar uma nova verificação de modelos a cada mudança aplicada para verificar se o KMTS modificado atende à propriedade.

O problema de refinamento é um problema difícil computacionalmente, e acreditamos ser possível através do grafo de testemunhas propor heurísticas com o intuito de contornar o problema e encontrar algoritmos eficientes em casos específicos.

As principais contribuições dessa dissertação são:

1. definição de uma estrutura, a qual chamamos de Grafo de Testemunhas, que captura as informações essenciais para o processo de refinamento de modelos KMTS;
2. desenvolvimento de algoritmos para o refinamento que atendem ao critério de minimalidade;
3. proposta de um verificador de modelos combinado com jogos para KMTS interpretado como um conjunto de estruturas de Kripke;
4. prova da complexidade do problema de refinamento e da verificação de modelos KMTS conforme a semântica adotada;
5. formalização e prova de propriedades sobre o problema do refinamento;
6. implementação do refinamento de modelos KMTSs.

1.4 TRABALHOS RELACIONADOS

O trabalho em (HUTH; JAGADEESAN; SCHMIDT, 2001) apresenta uma estrutura de Kripke com indefinições em suas transições para utilização em verificação abstrata de modelos. Os trabalhos em (SHOHAM; GRUMBERG, 2007) e (GRUMBERG et al., 2007) estendem a estrutura anterior adicionando indefinições nos estados do modelo, o qual é chamado de KMTS. Os autores interpretam o KMTS como uma estrutura que abstrai uma estrutura de Kripke. O objetivo deste trabalho é diminuir o tamanho do modelo utilizado na verificação de modelos, devido ao problema da explosão de estados. A técnica utilizada recebe o nome de verificação de modelos abstrata. O resultado da verificação pode resultar em indefinido, e neste caso os trabalhos (SHOHAM; GRUMBERG, 2007), (GRUMBERG et al., 2007) e (CHATZIELEFTHERIOU et al., 2012) propõem o refinamento do modelo KMTS, para obter o valor correto da verificação de modelos, ou seja, $\top$ ou F.

O trabalho (GUERRA; ANDRADE; WASSERMANN, 2013) interpreta o KMTS sobre uma outra ótica, em que o KMTS representa um conjunto de estruturas de Kripke.

Diferentemente da interpretação de (GRUMBERG et al., 2007), quando a verificação de modelos sobre um KMTS resulta em $\perp$, o modelo precisa ser refinado a fim de se obter as estruturas de Kripke que satisfazem a propriedade requerida.

Outros trabalhos como (BRUNS; GODEFROID, 1999) e (BRUNS; GODEFROID, 2000) consideram a abordagem de abstração através de estruturas parciais de Kripke as quais expressam indeterminação apenas em seus estados. O trabalho em (WEHRHEIM, 2008) utiliza a abstração para lidar com a especificação de sistemas parciais através de estruturas de Kripke e restrições expressas em LTL (*Linear Temporal Logic*). Em (HUTH; JAGADEESAN; SCHMIDT, 2001) os autores estendem estas estruturas com transições que representam possibilidades e definem o modelo KMTS. Em (GODEFROID; HUTH; JAGADEESAN, 2001) e (HUTH, 2002) os autores consideram o modelo MTS (*Modal Transitions Systems*) para lidar também com a abordagem de abstração.

Em todos estes trabalhos, a lógica de 3 valores de Kleene (KLEENE et al., 1952) ou uma interpretação equivalente é considerada, contudo estas lógicas não atendem à semântica de KMTS interpretado como um conjunto de modelos CTL.

1.5 ORGANIZAÇÃO

No capítulo 2, apresentamos a lógica CTL, as estruturas de Kripke e a semântica de um KMTS como conjunto de estruturas de Kripke. Mostramos também como estes modelos podem ser usados para a especificação e verificação formal de propriedades de sistemas.

No capítulo 3, apresentamos a verificação de modelos com jogos para lógica CTL proposta por (SHOHAM; GRUMBERG, 2007). Apresentamos inicialmente a verificação de modelos com jogos para a lógica CTL com respeito a uma estrutura de Kripke, e logo em seguida apresentamos a verificação de modelos com jogos para lógica CTL com três valores que consiste em verificar se um KMTS satisfaz uma fórmula CTL.

No capítulo 4, definimos o refinamento de modelos KMTSs e provamos algumas propriedades sobre o assunto. Definimos inicialmente as operações de mudanças primitivas aplicáveis sobre um KMTS, em seguida definimos o conceito de mudança minimal e instância maximal. Por último, provamos que o refinamento de um KMTS interpretado como um conjunto de modelos tem solução única. Para tanto definimos previamente algumas operações sobre mudanças.

No capítulo 5, propomos um verificador de modelos para a lógica CTL com relação a um KMTS interpretado como um conjunto de estruturas de Kripke. Inicialmente, desenvolvemos e apresentamos a semântica da lógica CTL com relação a um KMTS interpretado como um conjunto de modelos. Em seguida, definimos neste trabalho os operadores de interseção e contração sobre KMTSs e os utilizamos para definir o nosso verificador de modelos por contração. Introduzimos os conceitos de conjunto partição de KMTSs e propomos uma estrutura para representar estes conjuntos ao qual chamamos de Árvore Conjunto Partição. Utilizamos esta estrutura para provar a correção do nosso verificador de modelos por contração. Provamos, também, que o problema da verificação de modelos para um KMTSs interpretado como conjunto de modelos é NP-Completo. Os resultados apresentados no capítulo 5 foram publicados em (RIBEIRO; ANDRADE, 2015).

No capítulo 6, apresentamos a nosso proposta para o refinamento de modelos KMTSs. Para tanto, propomos no respectivo capítulo o Grafo de Testemunhas que contém as informações para realizar o refinamento de modelos KMTSs e propomos os algoritmos de refinamento sobre esta estrutura. Provamos a correção destes algoritmos e também que o refinamento de um modelo KMTS interpretado como um conjunto de modelos é NP-Difícil.

Finalmente, no capítulo 7, fazemos as considerações finais e discutimos sobre trabalhos futuros.

1.6 NOTAÇÃO

Usamos as letras gregas φ e ψ para representar fórmulas bem formadas da CTL, representamos símbolos proposicionais e literais por letras minúsculas $(l, p, q, \dots)$. Reservamos as letras gregas α, β, γ e Γ para denotar conjuntos, salvo quando indicado o contrário (por exemplo, indicamos o conjunto de estados de uma estrutura de Kripke, no capítulo 2, pela letra S). Representamos operações primitivas de mudanças pelo símbolo ϱ.

Neste trabalho, consideramos a interpretação de um KMTS como um conjunto de modelos, e denotamos o conjunto expansão de um KMTS M por $K(M)$. Escrevemos $[M, s \models \varphi] = \top$ para indicar que a verificação de modelos de uma fórmula φ sobre um KMTS M em um estado s do mesmo resulta em $\top$; escrevemos $[M, s \models \varphi] = F$ para indicar que o valor da verificação de modelos resulta em falso; e $[M, s \models \varphi] = \bot$ para indicar que o valor resultante é indefinido.

Capítulo

2

ESTRUTURAS DE KRIPKE E LÓGICA DE ÁRVORE COMPUTACIONAL

A verificação de modelos é uma técnica formal que permite a verificação automática de propriedades no modelo de um sistema. Para isto, é preciso especificar tais propriedades em alguma lógica, normalmente lógica temporal, e descrever o comportamento do sistema em um modelo finito de transições de estados.

Neste capítulo, apresentamos dois tipos de modelos formais de sistemas de transição: Estruturas de Kripke e Estruturas Modais de Kripke (*Kripke Modal Transition System* – KMTS). Estes modelos podem ser utilizados para representar o comportamento de sistemas, de forma a realizar, automaticamente, a verificação de modelos. O método de verificação de modelos é tratado no capítulo 3.

2.1 ESTRUTURAS DE KRIPKE

Definição 2.1.1. *(HUTH; RYAN, 2008)* Uma estrutura de Kripke é uma tupla $M = (AP, S, R, L)$ onde S é um conjunto finito de estados, R é uma relação binária em S ($R \subseteq S^2$) tal que para todo $s \in S$, existe algum $s' \in S$ com $s \rightarrow s'$. L é uma função de rotulação $L : S \rightarrow 2^{AP}$, onde AP é um conjunto de fórmulas atômicas. A função L rotula os átomos que ocorrem em um determinado estado s.

Uma estrutura de Kripke pode ser expressa por um grafo direcionado, a fim de facilitar a visualização do modelo. Nesta representação, cada nó do grafo é rotulado com as proposições atômicas, que valem naquele estado, enquanto as transições são representadas por arestas. Tome, por exemplo, uma estrutura de Kripke com três estados, ou seja, $S = \{s_0, s_1, s_2\}$; um conjunto $P = \{p, q, r\}$ de átomos. Considere a função de rotulação como $L(s_0) = \{p, q\}$, $L(s_1) = \{q, r\}$ e $L(s_2) = \{r\}$; e ainda as transições desta estrutura: $s_0 \rightarrow s_2$, $s_0 \rightarrow s_1$, $s_1 \rightarrow s_0$, $s_1 \rightarrow s_2$ e $s_2 \rightarrow s_2$. Esta estrutura pode ser representada pelo grafo da figura 2.1.

Estruturas de Kripke podem ser utilizadas para representar o comportamento de sistemas. Cada estado do modelo representa uma situação do sistema, enquanto as transições representam ações no sistema que o levam de um estado a outro.

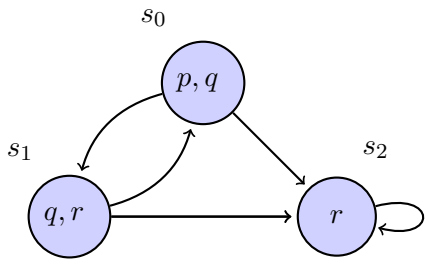

Figura 2.1: Estrutura de Kripke representado por um grafo direcionado. Fonte: Huth e Ryan (2008, p. 136)

Segundo a definição de estrutura de Kripke acima, a relação de transição deve ser total, contudo segundo (HUTH; RYAN, 2008) esta restrição tem apenas o intuito de garantir que o sistema representado nunca pare. No entanto, isto não reflete nenhuma restrição real sobre sistemas que podemos modelar. Se um sistema para, então é suficiente criar um estado de parada para representar tal situação e criar uma aresta que sai deste estado e incide sobre ele mesmo, ou seja, um laço. Desta forma, iremos considerar o relaxamento desta restrição, isto é, permitiremos que a relação de transições seja parcial. Este ponto é importante, pois mais a frente quando tratarmos de modelos KMTS como conjunto de estruturas de Kripke, iremos relaxar esta restrição.

2.2 KMTS-ESTRUTURAS DE TRANSIÇÕES MODAIS DE KRIPKE

Nos primeiros estágios de desenvolvimento de sistemas, é comum nos depararmos com informações parciais e/ou incompletas. Contudo, as estruturas de Kripke não são capazes de representar explicitamente este tipo de informação. Uma alternativa é considerar um conjunto de modelos que representem possíveis combinações destas informações. No entanto, esta abordagem não é eficiente na prática, pois quanto maior o número de indefinições, maior será o número de modelos. Nesse contexto, o ideal seria utilizar uma estrutura capaz de capturar estas informações parciais, como, por exemplo, o KMTS.

O KMTS é uma estrutura modal de Kripke capaz de especificar, de forma explícita, o comportamento do sistema frente a informações parciais. Esta estrutura contém dois tipos de transições: as que devem ocorrer e as que possivelmente ocorrem no sistema (*may* e *must*, respectivamente). Esta estrutura considera, também, a indefinição de propriedades nos estados do modelo, o que implica no desconhecimento da propriedade naquele estado, ou seja, a mesma pode ou não ocorrer no estado em questão do sistema final (HUTH; JAGADEESAN; SCHMIDT, 2001).

Definição 2.2.1. *(GUERRA; ANDRADE; WASSERMANN, 2013)* Um KMTS é uma tupla $M = (AP, S, R^+, R^-, L)$, onde S é um conjunto de estados finitos, $R^+ \subseteq S \times S$ e $R^- \subseteq S \times S$ são relações de transição, tal que $R^+ \subseteq R^-$; $Lit = AP \cup \{\neg p | p \in AP\}$ é o conjunto de literais sobre AP e $L : S \to 2^{Lit}$ é uma função de rotulação, tal que para todos os estados s e $p \in AP$, no máximo um entre p e $\neg p$ ocorre.

As transições R^+ e R^- de um KMTS correspondem às transições *must* e *may*, respectivamente.

A figura 2.2 apresenta um exemplo de um KMTS com conjunto $Lit = \{p, \neg p\}$. As linhas seccionadas indicam as transições *may* enquanto as cheias indicam as transições *must*. A transição *may* $s_0 \rightarrow s_2$ informa que esta transição possivelmente ocorre no sistema. Por outro lado, $s_1 \rightarrow s_2$ é uma transição *must* e como tal informa que a mesma deve estar presente no sistema. Observe que no estado s_1 não temos nenhum átomo, o que indica que o átomo p é indefinido neste estado, ou seja, não sabemos se ele é válido ou não em s_1. Vale ressaltar aqui a diferença entre a omissão dos termos de um estado em um grafo de uma estrutura de Kripke e de um KMTS. Nesta, a omissão indica a incompletude da informação, ou seja, não sabemos se tal propriedade é válida ou não no estado em questão; na estrutura de Kripke, a omissão indica que a propriedade não é válida no estado. Devemos observar, também, que esta omissão dos termos em um grafo de uma estrutura de Kripke comum, é apenas por uma questão de conveniência, evitando que escreva-se todas as propriedades em todos os estados (informando se ela é verdadeira ou não).

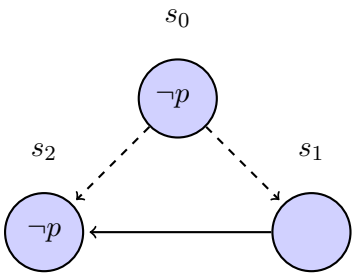

Figura 2.2: Exemplo de modelo KMTS.

2.2.1 KMTS como um Conjunto de Estruturas de Kripke

Um KMTS, como proposto em (GUERRA; ANDRADE; WASSERMANN, 2013), pode ser interpretado como um conjunto de estruturas de Kripke. Nesta interpretação, uma transição *may* em um KMTS representa dois modelos (um com a transição em questão e outro sem) e a indefinição de um literal l em um estado s de um KMTS representa outros dois modelos: um modelo onde l é válido no estado s e um outro modelo onde l não é válido neste mesmo estado. Sendo assim, um KMTS representa um conjunto com 2^n modelos, onde n é o número de indeterminações do KMTS (transições *may* e propriedades indefinidas nos estados).

Considere, como exemplo, o KMTS da figura 2.2, com duas transições *may* ($s_o \rightarrow s_1$ e $s_0 \rightarrow s_2$) e uma propriedade p indefinida no estado s_1 totalizando três indeterminações. A expansão deste modelo resulta em oito estruturas de Kripke (2^3) apresentadas na figuras 2.3.

Vale ressaltar, que o inverso não é necessariamente verdade. Isto é, dado um conjunto de estruturas de Kripke pode ser necessário considerar mais de um KMTS para representar

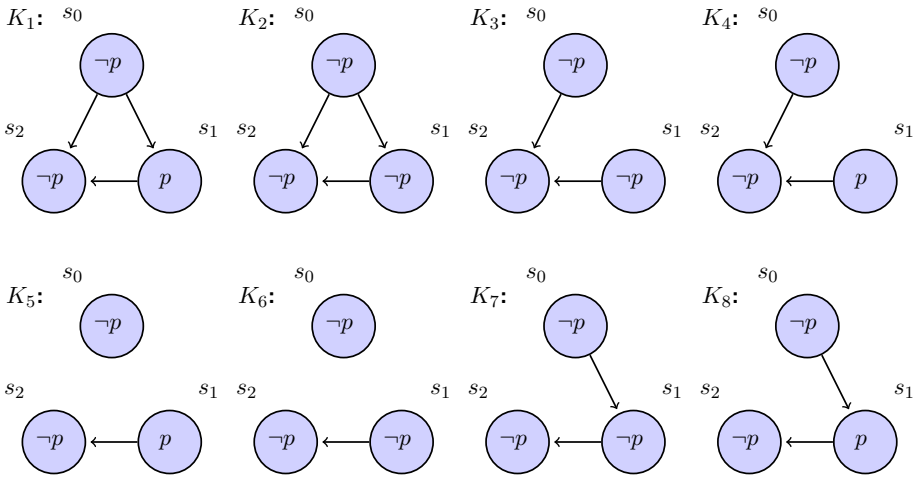

Figura 2.3: Expansão de um KMTS em um conjunto de estruturas de Kripke.

tal conjunto. Por exemplo, para o conjunto $\{K_1, K_2, K_5, K_6\}$ da figura 2.3, é necessário dois KMTSs (M_1 e M_2, apresentados na figura 2.4). O KMTS M_1 compacta os modelos K_1 e K_2, enquanto M_2 compacta os modelos K_5 e K_6.

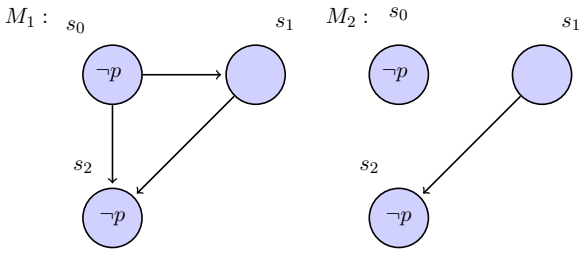

Figura 2.4: Conjunto KMTS para os modelos K_1, K_2, K_5, e K_6 da figura 2.3.

2.3 ESPECIFICAÇÃO E VERIFICAÇÃO DE PROPRIEDADES

Ao representarmos o comportamento do sistema por um modelo, como as estruturas de Kripke ou KMTS, podemos verificar a correção do mesmo frente a propriedades desejadas. Para isto, é preciso especificá-las em alguma lógica formal, normalmente, lógica temporal. Neste trabalho, estamos interessados em propriedades representadas pela lógica CTL.

2.3.1 Lógica CTL

A lógica CTL é uma lógica temporal que permite fazer referência ao futuro. Ela modela o tempo em uma sequência de estados (caminho computacional – veja definição 2.3.1) que se estende infinitamente para o futuro. Estruturas de Kripke (seção 2.1) são modelos desta lógica e a partir de agora, iremos nos referir às estruturas de Kripke como modelos

CTL.

Definição 2.3.1. (HUTH; RYAN, 2008) Um caminho computacional em uma estrutura de Kripke M é uma sequência infinita[1] de estados $s_0, s_1, \ldots$, em S, tal que, para cada $i \geq 0$, $s_i \rightarrow s_{i+1}$. Representamos este caminho por $s_0, s_1, \ldots$

Consideremos, por exemplo, a estrutura de Kripke representada na figura 2.1. Se considerarmos todos os possíveis caminhos a partir de s_0, obtemos a árvore computacional da figura 2.5.

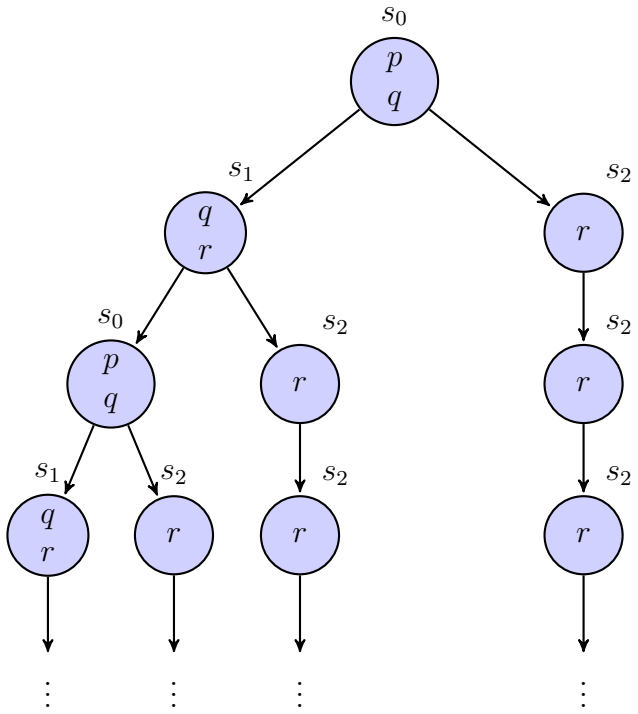

Figura 2.5: Caminho computacional da Estrutura de Kripke da figura 2.1.
Fonte: Huth e Ryan (2008, p. 137)

Em CTL, é possível quantificar os caminhos, ou seja, se existe uma sequência de estados do modelo que satisfaz uma fórmula Ψ ou se todos os caminhos do modelo, a partir de um estado s satisfazem Ψ. A definição 2.3.2 apresenta a sintaxe geral das fórmulas da CTL e a definição 2.3.4 a forma normal negativa da mesma.

Definição 2.3.2. *(HUTH; RYAN, 2008)* Seja p um símbolo proposicional, define-se Ψ uma fórmula bem formada da CTL por:
$\Psi ::= p \mid (\neg\Psi) \mid \Psi \ \wedge \ \Psi \mid \Psi \ \vee \ \Psi \mid \Psi \ \rightarrow \ \Psi \mid AX\ \Psi \mid EX\ \Psi \mid \ AF\ \Psi \mid EF\ \Psi$
$\Psi ::= EG\ \Psi \mid AG\ \Psi \mid A(\ \Psi\ U\ \Psi\ ) \mid E\ (\ \Psi\ U\ \Psi\ )\ A(\ \Psi\ R\ \Psi\ ) \mid E\ (\ \Psi\ R\ \Psi\ )$

[1]No KMTS a relação de transição pode ser parcial. Desta forma, caminhos computacionais nos modelos CTL, expandidos por estas estruturas, podem ser finitos.

Em CTL todos os conectivos (com exceção de $\neg, \vee, \wedge$ e $\rightarrow$) são formados por um par de símbolos: A ou E seguido de outro símbolo (X, F, G ou U). O símbolo A significa "para todos os caminhos" e E "existe um caminho". Os demais símbolos são: X (próximo estado), F(existe um estado no futuro), G(futuramente é sempre verdade), U(verdade até), e R (Solta).

$AX\Psi$ – Em todos os próximos estados de todos os caminhos Ψ é verdade.

$EX\Psi$ – Existe um caminho a partir do estado atual onde no próximo estado Ψ é verdadeiro.

$AF\Psi$ – Para todos os caminhos, existe um estado no futuro em que Ψ será verdade.

$EF\Psi$ – Existe um caminho, onde futuramente Ψ será verdadeira em algum estado.

$AG\Psi$ – Para todos os caminhos possíveis, Ψ é verdade em todos os estados.

$EG\Psi$ – Existe um caminho, onde Ψ é verdade em todos os estados.

$A(\Psi\ U\ \Psi')$ – Para todos os caminhos possíveis, Ψ é verdade, a partir do estado atual, e em todos os próximos estados até Ψ'passar a ser verdadeira em algum estado.

$E(\Psi\ U\ \Psi')$ – Existe um caminho, onde Ψ é verdade em todos os próximos estados até Ψ' passar a ser verdadeira em algum estado.

$A(\Psi\ R\ \Psi')$ – Para todos os caminhos possíveis, Ψ' é verdadeira, a partir do estado atual, e em todos os próximos estados até Ψ passe a ser verdadeira em algum estado. Ou ainda, para todos os caminhos possíveis, Ψ' é verdadeira, a partir do estado atual e em todos os próximos estados e Ψ nunca é satisfeita.

$E(\Psi\ R\ \Psi')$ – Existe um caminho onde Ψ' é verdadeira, a partir do estado atual, e em todos os próximos estados até Ψ passar a ser verdadeira em algum estado. Ou ainda, existe um caminho onde Ψ' é verdadeira, a partir do estado atual e em todos os próximos estados e Ψ nunca é satisfeita.

A semântica de uma fórmula CTL é definida em relação a uma estrutura de Kripke. Escrevemos $M, s \models \varphi$ para denotar que uma fórmula CTL φ é satisfeita por uma estrutura de Kripke M em um estado s, e escrevemos $M, s \not\models \varphi$ para denotar que φ não é satisfeita por M em um estado s.

Definição 2.3.3. *(SHOHAM; GRUMBERG, 2007) (HUTH; RYAN, 2008) Sejam φ uma fórmula CTL bem formada, $M = (AP, S, R, L)$ uma estrutura de Kripke, e $s_0 \in S$ um estado de M. A relação $M, s_0 \models \varphi$ é definida indutivamente da seguinte forma:*

1. *$M, s_0 \models p$ sse $p \in L(s_0)$;*
2. *$M, s_0 \models \neg\varphi$ sse $M, s_0 \not\models \varphi$;*
3. *$M, s_0 \models \varphi_1 \vee \varphi_2$ sse $M, s_0 \models \varphi_1$ e $M, s_0 \models \varphi_2$;*
4. *$M, s_0 \models \varphi_1 \wedge \varphi_2$ sse $M, s_0 \models \varphi_1$ ou $M, s_0 \models \varphi_2$;*
5. *$M, s_0 \models \varphi_1 \rightarrow \varphi_2$ sse $M, s_0 \not\models \varphi_1$ ou $M, s_0 \models \varphi_2$;*
6. *$M, s_0 \models AX\varphi$ sse para todo $s_0 \rightarrow s_i \in R$, tem-se que $M, s_i \models \varphi$;*
7. *$M, s_0 \models EX\varphi$ sse existe $s_0 \rightarrow s_i \in R$ tal que $M, s_i \models \varphi$;*

8. $M, s_0 \models AG\varphi$ *sse para todos os caminhos* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$, *e para todos os estados* $s_i \in \pi$, $M, s_i \models \varphi$;

9. $M, s_0 \models EG\varphi$ *sse existe um caminho* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$, *e para todos os estados* $s_i \in \pi$, $M, s_i \models \varphi$;

10. $M, s_0 \models AF\varphi$ *sse para todos os caminhos* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$, *existe um estado* $s_i \in \pi$ *tal que* $M, s_i \models \varphi$;

11. $M, s_0 \models EF\varphi$ *sse existe um caminho* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$ *e um estado* $s_i \in \pi$ *tal que* $M, s_i \models \varphi$;

12. $M, s_0 \models A(\varphi_1 U \varphi_2)$ *sse para todos os caminhos* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$, *existe um estado* $s_i \in \pi$ *tal que* $M, s_i \models \varphi_2$ *e para qualquer* $j < i$, $M, s_j \models \varphi_1$;

13. $M, s_0 \models E(\varphi_1 U \varphi_2)$ *sse existe um caminho* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$ *e um estado* $s_i \in \pi$ *tal que* $M, s_i \models \varphi_2$ *e para qualquer* $j < i$, $M, s_j \models \varphi_1$;

14. $M, s_0 \models A(\varphi_1 R \varphi_2)$ *sse para todos os caminhos* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$ *e para qualquer* $k \geq 0$, $M, s_j \not\models \varphi_1 \Rightarrow M, s_k \models \varphi_2$;

15. $M, s_0 \models E(\varphi_1 R \varphi_2)$ *sse existe um caminho* $\pi = s_0 \rightarrow s_1 \rightarrow \ldots$ *e para qualquer* $k \geq 0$, $M, s_j \not\models \varphi_1 \Rightarrow M, s_k \models \varphi_2$;

Neste trabalho, consideramos a lógica CTL em sua forma normal negativa.

Definição 2.3.4. *(SHOHAM; GRUMBERG, 2007)* Seja AP um conjunto de átomos e Lit o conjunto de literais de AP, isto é, $Lit = AP \cup \{\neg p | p \in AP\}$; e $l \in Lit$ um literal qualquer. A CTL em sua forma normal negativa é definida como:
$\Psi ::= l \mid \Psi \wedge \Psi \mid \Psi \vee \Psi \mid AX\ \Psi \mid EX\ \Psi$
$\Psi ::= A(\Psi\ U\ \Psi) \mid E(\Psi\ U\ \Psi) \mid A(\Psi\ R\ \Psi) \mid E(\Psi\ R\ \Psi)$

Como podemos perceber, A definição 2.3.4 da lógica CTL em sua forma normal negativa desconsidera alguns conectivos apresentados na definição 2.3.2 e restringe a negação somente aos símbolos proposicionais. De fato, não são necessários todos os conectivos apresentados na definição 2.3.2 para expressar toda a lógica CTL, pois existem subconjuntos próprios destes conectivos que formam uma base de conectivos na lógica CTL, como enunciado no teorema a seguir.

Teorema 2.3.1. *(HUTH; RYAN, 2008) Um conjunto de conectivos temporais é adequado se, e somente se, contém pelo menos um dos conectivos em* $\{AX, EX\}$, *pelo menos um dos conectivos em* $\{EG, AF, AU\}$ *e também contém o conectivo EU.*

Segue do teorema 2.3.1 que o conjunto de conectivos $\beta = \{\neg, \vee, \wedge, AX, AU, EU\}$ forma uma base de conectivos da CTL. Suponhamos agora, que a negação só possa ocorrer junto aos símbolos proposicionais. Desta forma, para que β seja uma base de conectivos, precisamos incluir os duais dos conectivos AX, AU e EU, isto é, precismos incluir os conectivos EX, AR e ER. Portanto, o conjunto $\beta' = \{\neg, \vee, \wedge, AX, EX, AU, EU, AR, ER\}$ é uma base de conectivos.

2.3.2 Exemplo de Verificação Utilizando KMTS

Podemos modelar o comportamento do sistema por um KMTS, quando precisarmos lidar com informação parcial, e especificar as propriedades em CTL, a fim de verificar se o modelo satisfaz as mesmas. Os autores em (GUERRA; ANDRADE; WASSERMANN, 2013) consideram um KMTS como uma estrutura que compacta um conjunto de modelos CTL. Por isso, ao verificarmos uma propriedade, três resultados podem ocorrer: verdadeiro ($\top$), se todos os modelos representados pelo KMTS satisfazem a propriedade, falso (F) se todos os modelos representando pelo KMTS não satisfazem a propriedade; e indefinido ($\bot$) se existem modelos, no conjunto expandido pelo KMTS, que satisfazem a propriedade e outros que não a satisfazem. Tomamos como exemplo o KMTS da figura 2.2 e as seguintes propriedades em CTL:

1. $EF\neg p$
2. EFp

Consideremos que a verificação será feita tomando s_0 como estado inicial. Dessa forma, o modelo satisfaz a propriedade 1, pois o próprio estado s_0 satisfaz $\neg p$. Observe, também, que todos os modelos da figura 2.3 satisfazem a propriedade 1. No entanto, a propriedade 2 é indefinida no KMTS, pois p ocorre de forma indefinida no estado s_1 e para alcançá-lo é preciso passar pela transição *may* $s_0 \rightarrow s_1$. Ou seja, nos modelos em que a transição $s_0 \rightarrow s_1$ não ocorre e/ou a propriedade em s_0 é $\neg p$ a propriedade 2 não é satisfeita. Logo temos um modelo que a satisfaz e outros que não a satisfazem. Podemos observar na figura 2.3 que os K_1 e K_8 são os únicos que satisfazem a propriedade, enquanto todos os outros não a satisfazem.

Capítulo 3

VERIFICAÇÃO DE MODELOS

Existem diversos algoritmos de verificação de modelos, mais detalhes em (HUTH; RYAN, 2008) e (CLARKE; GRUMBERG; PELED, 1999). Neste trabalho, no entanto, estamos interessados em realizar o refinamento do modelo KMTS quando a verificação deste resulta em indefinido. O algoritmo de verificação de modelos, baseado em jogos, proposto em (SHOHAM; GRUMBERG, 2007) informa as possíveis causas de falha para este caso. Estas informações podem ser usadas para realizar o refinamento do modelo (capítulo 4). Na seção 3.1 tratamos sobre o algoritmo baseado em jogos para lógica CTL com dois valores enquanto na seção 3.2, tratamos do algoritmo baseado em jogos para lógica CTL com três valores. Na seção 3.2.3, apresentamos as limitações destes verificadores de modelos sobre um KMTS interpretado como um conjunto de estruturação de Kripke e sua relação com o refinamento destes modelos.

3.1 VERIFICAÇÃO DE MODELOS COM JOGOS

Dada uma estrutura de Kripke $M = (AP, S, R, L)$ e uma fórmula CTL φ em sua forma normal negativa (definição 2.3.4 no capítulo 2), o algoritmo para verificação de modelos proposto em (SHOHAM; GRUMBERG, 2007) consiste em um jogo entre dois jogadores: $\exists$ (Eva) e $\forall$ (Abelardo). Eva (jogador $\exists$) tenta provar a satisfazibilidade da fórmula, enquanto Abelardo (jogador $\forall$) tenta refutar a mesma. O jogo consiste em uma arena (grafo do jogo) e um conjunto de regras que determina os movimentos realizáveis e as configurações em que cada jogador pode jogar.

A arena é formada pelo produto cartesiano dos estados de M com as sub-fórmulas de φ, ou seja, $S \times sub(\varphi)$, onde $sub(\varphi)$ é o conjunto de sub-fórmulas de φ apresentado na definição 3.1.1 a seguir.

Definição 3.1.1. (SHOHAM; GRUMBERG, 2007) Dada uma fórmula φ qualquer da CTL da forma $A(\varphi_1 U \varphi_2)$, $E(\varphi_1 U \varphi_2)$, $A(\varphi_1 R \varphi_2)$ ou $E(\varphi_1 R \varphi_2)$, sua expansão, representada por $exp(\varphi)$ é definida da seguinte forma:

$$exp(A(\varphi_1 U \varphi_2)) = \{A(\varphi_1 U \varphi_2),\ \varphi_2 \vee (\varphi_1 \wedge AX\varphi),\ \varphi_1 \wedge AX\varphi,\ AX\varphi\}$$

$exp(E(\varphi_1 U \varphi_2)) = \{E(\varphi_1 U \varphi_2),\ \varphi_2 \vee (\varphi_1 \wedge EX\varphi),\ \varphi_1 \wedge EX\varphi,\ EX\varphi\}$
$exp(A(\varphi_1 R \varphi_2)) = \{A(\varphi_1 R \varphi_2),\ \varphi_2 \wedge (\varphi_1 \vee AX\varphi),\ \varphi_1 \vee AX\varphi,\ AX\varphi\}$
$exp(E(\varphi_1 R \varphi_2)) = \{E(\varphi_1 R \varphi_2),\ \varphi_2 \wedge (\varphi_1 \vee EX\varphi),\ \varphi_1 \vee EX\varphi,\ EX\varphi\}$

Definição 3.1.2. (SHOHAM; GRUMBERG, 2007) O conjunto de sub-fórmulas $sub(\varphi)$ é definida como segue:

Se $\varphi = \top, F$ *ou* l, *onde* $l \in Lit$, *então* $sub(\varphi) = \{\varphi\}$
Se $\varphi = \varphi_1 \wedge \varphi_2$ *ou* $\varphi_1 \vee \varphi_2$, *então* $sub(\varphi) = \{\varphi\} \cup sub(\varphi_1) \cup sub(\varphi_2)$
Se $\varphi = AX\varphi_1$ *ou* $EX\varphi_1$, *então* $sub(\varphi) = \{\varphi\} \cup sub(\varphi_1)$
Se $\varphi = A(\varphi_1 U \varphi_2)$, $E(\varphi_1 U \varphi_2)$, $A(\varphi_1 R \varphi_2)$ *ou* $E(\varphi_1 R \varphi_2)$, *então* $sub(\varphi) = exp(\varphi) \cup sub(\varphi_1) \cup sub(\varphi_2)$

As configurações são elementos da arena e denotadas por $C_i = (s_j, \psi)$, onde $C_i \in S \times sub(\varphi)$, $s_j \in S$ e $\psi \in sub(\varphi)$. Em cada configuração somente um dos jogadores pode realizar o movimento. O jogo começa na configuração $C_0 = (s_0, \phi)$ e o jogador da vez escolhe, de acordo com as regras do jogo e sua estratégia, a próxima configuração, onde então, o jogador desta fará o seu movimento e assim sucessivamente até que o jogo acabe. A esta sequência de movimentos, dá-se o nome de jogada.

Definição 3.1.3. Uma jogada é uma sequência, possivelmente infinita, de configurações denotada por: $C_0 \to C_1 \to \ldots$.

Os movimentos continuam a ser executados até que o jogo acabe. O jogo termina somente em dois casos: ao alcançar uma configuração final (configuração sem transições para outras configurações) ou se os participantes do jogo, Eva ($\exists$) e Abelardo ($\forall$), jogam infinitamente frequentemente, isto é, eles jogam em um ciclo da arena. Se uma jogada é infinita, então uma das seguintes sub-fórmulas ocorre: AU, EU, AR e ER. À sub-fórmula que ocorre na jogada, dá-se o nome de testemunha (SHOHAM; GRUMBERG, 2007).

As regras do jogo determinam as configurações de cada jogador e os movimentos que podem ser realizados em cada configuração. Elas são definidas de acordo com a sub-fórmula φ que ocorre na configuração C_i. As regras são definidas como segue:

(1) $C_i = (s, false)$, $C_i = (s, true)$, $C_i = (s, l)$, onde l é um literal. Estas são as configurações terminais. Nestas configurações, não há escolha de um movimento, o jogo simplesmente acaba, desta forma não faz diferença qual o jogador da configuração. Por isso, podemos atribuir Abelardo ou Eva para a tarefa. Sem perda de generalidade, Eva foi escolhida.
(2) $C_i = (s, AX\varphi)$, Abelardo ($\forall$) escolhe uma transição $s \to s'$ em M e $C_{i+1} = (s', \varphi)$.
(3) $C_i = (s, EX\varphi)$, Eva ($\exists$) escolhe uma transição $s \to s'$ em M e $C_{i+1} = (s', \varphi)$.
(4) $C_i = (s, \varphi_1 \wedge \varphi_2)$, Abelardo($\forall$) escolhe $j \in \{1, 2\}$ e $C_{i+1} = (s, \varphi_j)$.
(5) $C_i = (s, \varphi_1 \vee \varphi_2)$, Eva ($\exists$) escolhe $j \in \{1, 2\}$ e $C_{i+1} = (s, \varphi_j)$.
(6) $C_i = (s, A(\varphi_1 U \varphi_2))$, Eva ($\exists$) $C_{i+1} = (s, \varphi_2 \vee (\varphi_1 \wedge AXA(\varphi_1 U \varphi_2)))$.
(7) $C_i = (s, E(\varphi_1 U \varphi_2))$, Eva ($\exists$) $C_{i+1} = (s, \varphi_2 \vee (\varphi_1 \wedge EXE(\varphi_1 U \varphi_2)))$.
(8) $C_i = (s, A(\varphi_1 R \varphi_2))$, Eva ($\exists$) $C_{i+1} = (s, \varphi_2 \wedge (\varphi_1 \vee AXA(\varphi_1 R \varphi_2)))$.
(9) $C_i = (s, E(\varphi_1 R \varphi_2))$, Eva ($\exists$) $C_{i+1} = (s, \varphi_2 \wedge (\varphi_1 \vee EXE(\varphi_1 R \varphi_2)))$.

Observe que as regras (6) a (9) são deterministas, ou seja, somente um movimento é possível. Por esta razão, não faz diferença qual o jogador da configuração. Assim, sem perda de generalidade, Eva ($\exists$) foi escolhida como a jogadora destas configurações.

Cada configuração da arena é colorida a depender do resultado obtido pelos movimentos realizados a partir daquela configuração: F se Abelardo ($\forall$) tem estratégia de ganhar independente do outro jogador, $\top$ caso Eva ($\exists$) tenha estratégia de ganhar independente das escolhas do outro jogador.

A estratégia é uma função parcial $F : C \to C$, onde C é o conjunto das configurações da arena do jogo. Um jogador tem uma estratégia de ganhar se ele vence em todas as configurações da jogada de acordo com a sua estratégia.

Abelardo ($\forall$) tem estratégia de ganhar, ou seja, vence o jogo, se e somente se uma das seguintes condições ocorre:

(1) a jogada é finita e termina em uma configuração final do tipo $C_i = (s, false)$ ou $C_i = (s, l)$, onde l é um literal e $l \notin L(s)$;
(2) a jogada é infinita e a testemunha tem uma das seguintes formas: AU ou EU.

Eva ($\exists$) tem estratégia de ganhar, ou seja, vence o jogo, se e somente se uma das seguintes condições ocorre:

(1) a jogada é finita e termina em uma configuração final do tipo $C_i = (s, true)$ ou $C_i = (s, l)$, onde l é um literal e $l \in L(s)$;
(2) a jogada é infinita e a testemunha tem uma das seguintes formas: AR ou ER.

O algoritmo de verificação de modelos baseado em jogos consiste em colorir a arena. Ele decide se $M, s_0 \models \varphi$ de acordo com a cor ($\top$ ou F) atribuída à configuração inicial $C_0 = (s_0, \varphi)$ da arena. Apresentamos na seção 3.1.1 como construir a arena e na seção 3.1.2 como colorir a mesma.

3.1.1 Arena do Jogo

A arena do jogo consiste em um conjunto de configurações e de arestas. Estas representam os possíveis movimentos dos jogadores em cada configuração. Como estamos interessados em saber o resultado da jogada iniciada na configuração $C_0 = (s_0, \varphi)$, onde $s_0 \in S$ é o estado inicial de um modelo M, a arena será construída a partir dela. A cada configuração alcançada, aplica-se as regras do jogo a fim de obter as arestas e estados alcançáveis a partir da mesma. Este processo se propaga recursivamente visitando-se os nós filhos desta configuração, até alcançar um estado final ou fechar um ciclo. O resultado é um grafo $G = (V, E)$, onde $V \subseteq S \times sub(\varphi)$ é o conjunto de configurações do jogo e $E \subseteq V \times V$ o conjunto de arestas.

As arestas que partem das configurações do tipo AX e EX refletem as transições de uma estrutura de Kripke M utilizada na verificação de modelos. A figura 3.1 apresenta uma arena para uma fórmula $\varphi = A(\text{p R q})$ e uma estrutura de Kripke M presente na mesma figura. A fim de evitar uma quantidade desnecessária de parênteses e garantir uma melhor legibilidade das configurações, adotaremos nas figuras a notação utilizada pelos autores em (GUERRA; ANDRADE; WASSERMANN, 2013). Cada configuração

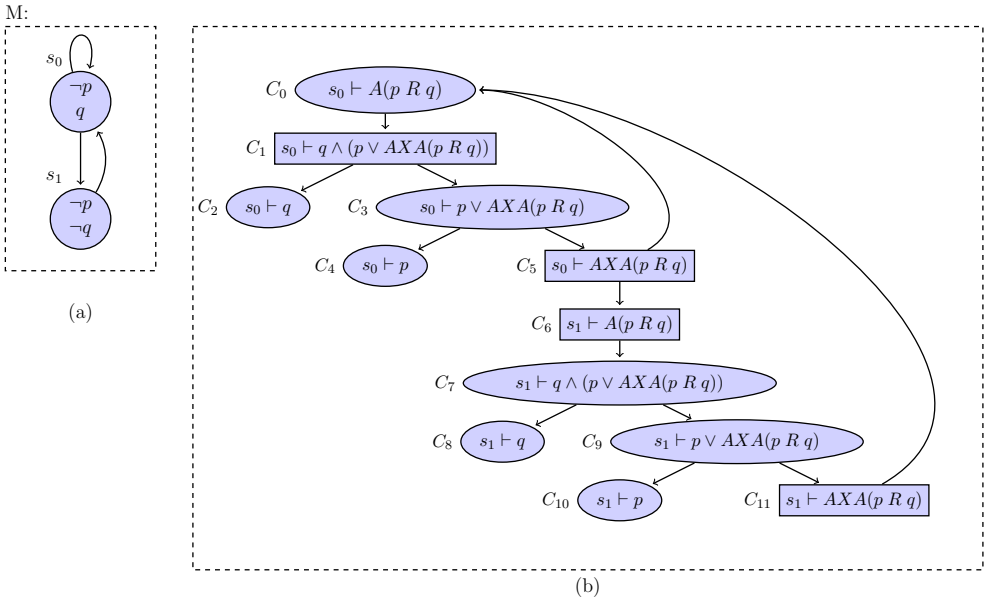

Figura 3.1: Arena do jogo (b) para uma estrutura de Kripke M (a) e fórmula $\varphi = A(p\ R\ q)$. As configurações elípticas indica as configurações em que Eva joga, enquanto as retangulares indicam as configurações pertencentes a Abelardo. Fonte: Adaptada de Shoham e Grumberg (2007, p. 11)

$C_i = (s_j, \psi)$, onde $i, j \in \mathbb{N}$ é representada na figura como $s_j \vdash \psi$. Por exemplo, a configuração $C_0 = (s_0 \vdash A(p\ R\ q))$ é apresentada na figura 3.1 como $s_0 \vdash A(pRq)$. O modelo M apresenta a estrutura de Krikpe utilizada na construção do grafo do jogo. A configuração C_0 é do tipo A(p R q), e por isso aplica-se à regra (8) gerando a configuração C_1. Esta última é do tipo $\varphi_1 \wedge \varphi_2$, assim aplica-se a regra (4), gerando duas configurações (C_2 e C_3). C_2 é uma configuração terminal, logo, mais nenhuma configuração pode ser gerada. Tomemos, agora, a configuração C_5, do tipo $AXA\varphi_1$. Como dito anteriormente, neste tipo de configuração as arestas devem refletir as transições do modelo. Olhando para o mesmo, podemos observar que existem duas transições que partem do estado s_0 ($s_0 \rightarrow s_0$ e $s_0 \rightarrow s_1$). Assim, duas transições partem de C_5: $C_5 \rightarrow C_0$, refletindo a transição $s_0 \rightarrow s_0$; e $C_5 \rightarrow C_6$, refletindo a transição $s_0 \rightarrow s_1$.

Uma arena pode ser dividida em suas componentes fortemente conexas maximais.

Definição 3.1.4. *Um subgrafo G' de G é dito maximal com respeito a uma propriedade τ sse G' detém a propriedade τ e não é um subgrafo próprio de nenhum outro subgrafo de G que detenha τ.*

Definição 3.1.5. Uma componente fortemente conexa maximal (CFCM) de um grafo direcionado $G = (V, E)$, é um subgrafo $G' = (V', E')$, tal que

(1) para todo par de vértices u e v em V, existe um caminho de u para v e vice-versa;

(2) G' é maximal em relação à condição (1).

Lema 3.1.1. (SHOHAM; GRUMBERG, 2007) Seja β o conjunto das configurações de uma CFCM, que tenham pelo menos uma aresta da arena do jogo, então o conjunto de fórmulas associadas às configurações em β é exatamente um dos conjuntos $exp(\varphi)$ (definição 3.1.1), onde $\varphi \in \{A(\varphi_1 U \varphi_2), E(\varphi_1 U \varphi_2), A(\varphi_1 R \varphi_2), E(\varphi_1 R \varphi_2)\}$.

A fórmula φ da qual trata o lema 3.1.1 é a testemunha da CFCM que pode levar a uma jogada infinita. Esta testemunha é única para cada CFCM. A figura 3.1 tem exatamente cinco CFCMs: $F_1 = \{C_2\}, F_2 = \{C_4\}, F_3 = \{C_8\}, F_4 = \{C_{10}\}$ e $F_5 = \{C_0, C_1, C_3, C_5, C_6, C_7, C_9, C_{11}\}$. Esta informação é utilizada pelo algoritmo de verificação de modelos com jogos para colorir o grafo do jogo e decidir o vencedor.

3.1.2 Algoritmo para a Verificação de Modelos com Jogos

O algoritmo de verificação de modelos com jogos consiste em colorir cada configuração da arena do jogo com uma das seguintes cores: $\top$ se Eva ($\exists$) tem estratégia de ganhar ou F se Abelardo ($\forall$) tem estratégia de ganhar. O algoritmo particiona o grafo do jogo em suas CFCMs e as colore, de forma ascendente, baseado na relação de ordem parcial $\leq$ definida sobre as CFCMs do grafo do jogo.

Definição 3.1.6. (SHOHAM; GRUMBERG, 2007) Sejam Q_i e Q_j CFCMs quaisquer do grafo do jogo, $\leq$ é uma relação de ordem parcial definida sobre as CFCMs do grafo do jogo, tal que um caminho $\pi = n \rightarrow \cdots \rightarrow n'$ existe se e somente se $Q_i \leq Q_j$; onde $n \in Q_j$ e $n' \in Q_i$.

Cada CFCM Q_i é colorida da seguinte forma:

(1) Configurações terminais C_j em Q_i são coloridas com $\top$ se $C_j = (s, true)$, ou $C_j = (s, l)$ e $l \in L(s)$; F caso $C_j = (s, false)$, ou $C_j = (s, l)$ e $l \notin L(s)$.
(2) Configurações $C_j = (s, \varphi)$ em Q_i, onde φ tem forma $\varphi_1 \vee \varphi_2$ ou $EX\varphi_1$, são coloridas com $\top$ se existe um filho colorido com $\top$; F caso contrário.
(3) Configurações $C_j = (s, \varphi)$ em Q_i, onde φ tem forma $\varphi_1 \wedge \varphi_2$ ou $AX\varphi_1$, são coloridas com F se existe um filho colorido com F; $\top$ caso contrário.
(4) Configurações em Q_i, que após a aplicação sucessiva das regras (1) a (3) ainda permanecem sem ser coloridas, serão coloridas, de acordo com a testemunha em Q_i: F se a testemunha for da forma AU ou EU; $\top$ se a testemunha for da forma AR ou ER.

Segundo (SHOHAM; GRUMBERG, 2007), a aplicação do algoritmo sobre uma arena G resulta em uma função $\delta : N \rightarrow \{T, F\}$, onde N é o conjunto das configurações em G. δ informa a cor atribuída a uma configuração $C_i \in N$ do grafo do jogo. Os teoremas a seguir podem ser encontrados em (STIRLING, 2001) e (SHOHAM; GRUMBERG, 2007).

Teorema 3.1.1. Seja δ a função de coloração resultante após a aplicação do algoritmo sobre uma arena e C_i uma configuração qualquer da mesma. Então:

(1) $\delta(C_i) = \top$ se e somente se Eva ($\exists$) tem estratégia de ganhar para todas as jogadas a partir da configuração C_i;

(2) $\delta(C_i) = F$ se e somente se Abelardo ($\forall$) tem estratégia de ganhar para todas as jogadas a partir da configuração C_i.

Teorema 3.1.2. Seja M uma estrutura de Kripke e ψ uma fórmula CTL. Então, para toda configuração $C_i = (s, \varphi)$ da arena deste jogo, onde $\varphi \in sub(\psi)$, tem-se que:

(1) $M, s \models \varphi$ se e somente se C_i for colorido com $\top$, ou seja, se e somente se Eva ($\exists$) tem estratégia de ganhar começando em C_i;
(2) $M, s \not\models \varphi$ se e somente se C_i for colorido com F, ou seja, se e somente se Abelardo ($\forall$) tem estratégia de ganhar começando em C_i.

De acordo com os teoremas 3.1.1 e 3.1.2, pode-se concluir que dada uma estrutura de Kripke M, um estado inicial s_0 desta estrutura e uma fórmula CTL ψ, $M, s_0 \models \psi$ se e somente se $\delta(C_i) = \top$, com $C_i = (s_0, \psi)$; e $M, s_0 \not\models \psi$, caso contrário. (SHOHAM; GRUMBERG, 2007).

3.1.3 Exemplo de Verificação de Modelos com Jogos

O algoritmo de verificação de modelos com jogos pode ser aplicado sobre a arena do jogo da figura 3.1, obtendo-se como resultado a arena colorida da figura 3.2.

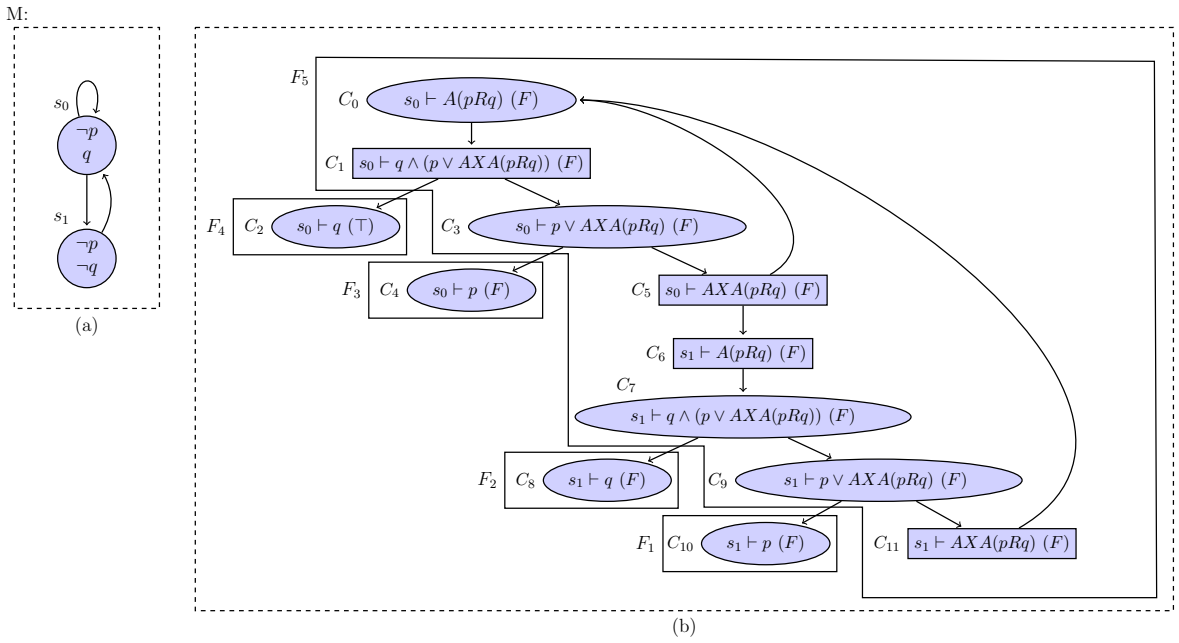

Figura 3.2: Arena do jogo (b) para uma estrutura de Kripke M (a) e fórmula $\varphi = A(pRq)$. Fonte. Adaptada de Shoham e Grumberg (2007, p. 11).

Na figura 3.2 cada configuração é colorida com (F) ou $(\top)$. A arena é particionada em suas CFCMs e rotuladas por F_i, com $i \in \mathbb{N}^*$, que indica a ordem de coloração de cada uma. Assim, o algoritmo colore primeiro a CFCM F_1 que tem uma configuração

terminal $C_{10} = (s_1, p)$, o átomo p não vale no estado s_1 de M, assim, de acordo com a regra (1) do algoritmo esta configuração deve ser colorida com (F). O mesmo ocorre para as CFCMs F_2, F_3 e F_4, esta última diferente das anteriores é colorida com $(\top)$, visto que o átomo q vale no estado s_0 do modelo M. A CFCM F_5 é a última a ser colorida. Nesta CFCM, aplica-se as regras (2) e (3) do algoritmo. C_7 é colorida com (F), pois ela é da forma $\varphi_1 \wedge \varphi_2$ e tem um filho (C_8) colorido com (F), obedecendo a regra de coloração (3) do algoritmo. Esta coloração se propaga para cima colorindo todos os outros nós com (F). Ao final, C_0 é colorido com F, logo $M, s_0 \not\models A(pRq)$. As colorações de C_9 e C_{11} ocorrem após a coloração de C_0. Visto que C_0 está colorido com F e é uma configuração filha de C_{11}, aplica-se a regra (3) do algoritmo. Desta forma, C_11 é colorido com F. Visto que ambas configurações filhas de C_9 estão coloridas com F, aplica-se a regra (2) do algoritmo, e C_9 é colorida com F.

3.2 VERIFICAÇÃO DE MODELOS COM JOGOS PARA LÓGICA DE 3 VALORES

O trabalho (SHOHAM; GRUMBERG, 2007) abstrai uma estrutura de Kripke em um KMTS e propõe um algoritmo de verificação de modelos com a lógica de três valores. A verificação de modelos com jogos proposta recebe como entrada um modelo KMTS M, um estado inicial s_0 deste modelo, uma fórmula ψ e decide se $M, s_0 \models \psi$. O jogo continua a ser disputado por Eva $(\exists)$ e Abelardo $(\forall)$ que tentam, respectivamente, provar a validade da fórmula e refutá-la. Três resultados são possíveis:

(1) $\top$, então a estrutura de Kripke abstraída pelo KMTS M satisfaz ψ;
(2) F, então a estrutura de Kripke abstraída pelo KMTS M não satisfaz ψ;
(3) $\bot$, nada se pode afirmar.

A verificação de modelos considera fórmulas CTL escritas na sintaxe de μ-calculus. A definição abaixo apresenta a sintaxe de μ-calculus em sua forma normal negativa.

Definição 3.2.1. (CLARKE; GRUMBERG; PELED, 1999) Seja AP um conjunto de proposições atômicas e V um conjunto de nomes de variáveis. O conjunto de literais sobre AP é definido como $Lit = AP \cup \{\neg p | p \in AP\}$. O μ-calculus em sua forma normal negativa sobre AP é definida por $\varphi ::= l \mid Z \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \langle a \rangle X\varphi \mid [a]X\varphi \mid \mu Z.\varphi \mid \nu Z.\varphi$ onde $l \in Lit$ e $Z \in V$. $[a]X$ significa para todos os sucessores alcançáveis através de uma aresta com rótulo "a" e $\langle a \rangle X$ significa existe um sucessor alcançável através de uma aresta com rótulo "a".

Os termos $\mu Z.\varphi$ e $\nu Z.\varphi$ denotam respectivamente o menor e maior ponto fixo de certas funções recursivas, que capturam a semântica dos operadores temporais da lógica CTL. Para o μ-calculus completo, as arestas do KMTS representam ações que ocorrem no sistema. No entanto, como estamos interessados somente em propriedades especificadas em CTL e, também, por tratarmos um KMTS como uma estrutura que compacta diversos modelos CTL, não precisamos considerar esses rótulos. Desta forma, é suficiente considerar os conectivos $\langle a \rangle X$ e $[a]X$ por EX e AX, respectivamente.

Fórmulas CTL podem ser especificadas em μ-calculus pelas seguintes equivalências: $EF\phi \equiv \mu Z.\phi \vee EXZ$; $AF\phi \equiv \mu Z.\phi \vee AXZ$; $EG\phi \equiv \upsilon Z.\phi \wedge EXZ$; $AG\phi \equiv \upsilon Z.\phi \wedge AXZ$;

$E[\phi U \phi] \equiv \mu Z.\phi \vee (\phi \wedge EXZ)$; e $A[\phi U \phi] \equiv \mu Z.\phi \vee (\phi \wedge AXZ)$. (CLARKE; GRUMBERG; PELED, 1999)

Assim como no algoritmo da seção anterior, a verificação de modelos para lógica de três valores com jogos constrói primeiramente uma arena, que será colorida posteriormente, baseado nas regras do jogo e nas estratégias dos jogadores. Dado um KMTS M e uma fórmula ψ, a arena para este jogo é construída decompondo-se a fórmula ψ em suas subfórmulas e combinando-as com os estados do modelo M, considerando-se as regras do jogo apresentadas na figura 3.3.

Regras para Eva($\exists ve$)

(1) $\frac{s \vdash \psi_0 \vee \psi_1}{s \vdash \psi_i} : i \in \{0,1\}$ (2) $\frac{s \vdash EX\psi}{t \vdash \psi} : R^+(s,t)$ ou $R^-(s,t)$

(3) $\frac{s \vdash \eta Z.\psi}{s \vdash Z} : \eta \in \{\upsilon, \mu\}$ (4) $\frac{s \vdash Z}{s \vdash \psi}$

Regras para Abelardo

(5) $\frac{s \vdash \psi_0 \wedge \psi_1}{s \vdash \psi_i} : i \in \{0,1\}$ (6) $\frac{s \vdash AX\psi}{t \vdash \psi} : R^+(s,t)$ ou $R^-(s,t)$

Figura 3.3: Regras do jogo com lógica de três valores. Fonte. Adaptado de (GRUMBERG et al., 2007).

As regras do jogo definem os movimentos possíveis em cada configuração, e qual o jogador responsável por realizar o movimento na mesma. Na figura 3.3, cada configuração $C_i = (s, \psi)$ do grafo do jogo é representada por $s \vdash \psi$. A parte superior de cada regra indica a forma da configuração do jogador a qual a regra deve ser aplicada, enquanto a parte inferior indica as configurações atingíveis a partir dela. As regras (2) e (6) refletem as transições do KMTS. As regras definidas nesta figura foram adaptadas do original, para lidar, somente, com fórmulas CTL, uma vez que estamos interessados somente em propriedades especificadas nesta lógica. A proposta de verificação de modelos para o μ-calculus completo pode ser encontrada em (GRUMBERG et al., 2007).

O jogo inicia-se na configuração inicial $C_0 = (s_0, \psi)$ e cada jogador faz o seu movimento de acordo com a sua estratégia, até que o jogo acabe (alcançar configuração terminal ou jogar infinitamente frequentemente). Uma jogada é infinita, somente se uma variável Z ligada a um ponto fixo ocorrer. Esta variável é chamada de testemunha.

As estratégias dos jogadores são diferentes das apresentadas na seção anterior. Considerando as regras (2) e (6) da figura 3.3 que refletem as transições do modelo KMTS, os jogadores podem optar por jogar por transições *must* e *may*. Eva e Abelardo devem escolher jogar por transições *must* se desejarem ganhar o jogo, ou seja, provar a validade da fórmula ou refutar a mesma, respectivamente. No entanto, podem jogar por uma transição *may* se quiserem evitar perder o jogo. Assim, se um jogador deseja vencer, deverá jogar consistentemente, isto é, apenas por transições *must*. No entanto, poderá jogar por uma transição *may* se quiser evitar perder.

Definição 3.2.2. (SHOHAM; GRUMBERG, 2007) Uma jogada é consistente se em cada configuração da forma $C_i = (s, EX\varphi)$ e $C_i = (s, AX\varphi)$, respectivamente, Eva ($\exists$) e Abelardo ($\forall$) jogarem por transições *must*.

As estratégias de ganhar de acordo com (SHOHAM; GRUMBERG, 2007) e (GRUMBERG et al., 2007) são apresentadas a seguir.

Abelardo ($\forall$) tem estratégia de ganhar, se jogar sempre consistentemente e uma das seguintes condições ocorrer:

(1) A jogada é finita e termina em uma configuração terminal da forma $C_i = (s, false)$ ou $C_i = (s, l)$; onde $\neg l \in L(s)$.
(2) joga infinitamente frequentemente e a testemunha é uma variável de menor ponto fixo.

Eva ($\exists$) tem estratégia de ganhar, se jogar sempre consistentemente e uma das seguintes condições ocorrer:

(1) A jogada é finita e termina em uma configuração terminal da forma $C_i = (s, true)$ ou $C_i = (s, l)$; onde $l \in L(s)$.
(2) joga infinitamente frequentemente e a testemunha é uma variável de maior ponto fixo.

Para qualquer outro caso, o jogo termina em empate, ou seja, a configuração inicial C_0 é colorida com $\perp$. O que indica que ambos os jogadores tem estratégia de não perder.

Teorema 3.2.1. Adaptado de (SHOHAM; GRUMBERG, 2007). Seja M um KMTS qualquer e ψ uma fórmula CTL qualquer expressa em μ-calculus. Então, para cada configuração $s \in S$, da arena do jogo, onde S é o conjunto de estados de M:

(1) $[M, s \models \psi] = \top$ se e somente se Eva ($\exists$) tem estratégia de ganhar começando em s;
(2) $[M, s \models \psi] = F$ se e somente se Abelardo ($\forall$) tem estratégia de ganhar começando em s;
(3) $[M, s \models \psi] = \perp$ se e somente se nenhum dos dois jogadores tem estratégia de ganhar começando em s, ou seja, ambos tem estratégia de não perder começando em s.

Teorema 3.2.2. (SHOHAM; GRUMBERG, 2007). Seja δ a função de coloração resultante após a aplicação do algoritmo sobre uma arena e C_i uma configuração qualquer da mesma. Então:

(1) $\delta(C_i) = \top$ se e somente se Eva ($\exists$) tem estratégia de ganhar começando em C_i;
(2) $\delta(C_i) = F$ se e somente se Abelardo ($\forall$) tem estratégia de ganhar começando em C_i;
(3) $\delta(C_i) = \perp$ se e somente se nenhum dos jogadores tem estratégia de ganhar começando em C_i, ou seja, ambos os jogadores tem estratégia de não perder a partir de C_i.

3.2.1 Algoritmo de Coloração

Apresentamos nesta seção o algoritmo de coloração para a verificação de modelos com jogos para lógica de três valores proposto em (SHOHAM; GRUMBERG, 2007). O algoritmo aqui apresentado é uma adaptação do original, para trabalhar sobre a arena gerada sobre as regras do jogo da figura 3.3. A adaptação consiste basicamente em colorir a arena do jogo considerando uma fórmula CTL escrita em μ-calculus. As regras de coloração para configurações que não ocorrem em CFCM com apenas uma configuração permanecem as mesmas. Para as demais CFCM, cada configuração é colorida de acordo com a variável de ponto fixo que ocorre no ciclo: variável de menor ou maior ponto fixo. Estas variáveis dizem respeitos as fórmulas ligas pelos conectivos: AU, EU, AR e ER.

Para o algoritmo de coloração, consideramos dois tipos configurações. Uma configuração é dita configuração *may* ou nó *may* se for filha de uma outra configuração e a aresta que os conecta for do tipo *may*. Todas as outras configurações serão chamadas de configuração *must* ou nó *must*.

O algoritmo particiona a arena em suas CFCMs (definição 3.1.5) e as colore de forma ascendente com respeito a relação de ordem parcial $\leq$ (definição 3.1.6). Cada CFCM F_i é colorida em duas fases, como segue:

1ª Fase: Coloração dos nós filhos
As regras a seguir são aplicadas a todas configurações em F_i até que mais nenhuma possa ser aplicada.

(1) Configurações terminais são coloridas com $\top$ se Eva ($\exists$) tem estratégia de ganhar, com F se Abelardo ($\forall$) tem estratégia de ganhar e com $\bot$ caso contrário.

(2) Configurações da forma $AX\varphi$ são coloridas da seguinte forma:

$\top$ se todos os nós filhos (*may* ou *must*) estiverem coloridos com $\top$;

F se existe um nó filho *may* colorido com F;

$\bot$ se todos os filhos *must* estiverem coloridos com $\top$ ou $\bot$; e existe um filho *may* colorido com F ou $\bot$.

(3) Configurações da forma $EX\varphi$ são coloridas da seguinte forma:

$\top$ se existe um filho *must* colorido com $\top$;

F se todos os filhos (*may* ou *must*) estiverem coloridos com F;

$\bot$ se existe um filho *may* colorido com $\bot$ ou $\top$ e todos os filhos *must* estiverem coloridos com F ou $\bot$.

(4) Configurações da forma $\varphi_1 \wedge \varphi_2$ são coloridas da seguinte forma:

$\top$ se ambos os filhos estiverem coloridos com $\top$;

F se um dos filhos estiver colorido com F;

$\bot$ se um dos filhos estiver colorido com $\top$ ou $\bot$ e o outro estiver colorido com $\bot$.

(5) Configurações da forma $\varphi_1 \vee \varphi_2$ são coloridas da seguinte forma:

$\top$ se existe um filho colorido com $\top$;

F se ambos os filhos estiverem coloridos com F;

$\bot$ se um dos filhos estiver colorido com $\bot$ ou F e o outro estiver colorido com $\bot$.

$2^{\underline{a}}$ *Fase:* Coloração das testemunhas

Se após a propagação das regras da $1^{\underline{a}}$ fase, ainda existirem nós na CFCM F_i sem estarem coloridos, então os jogadores jogam infinitamente frequentemente em F_i e a testemunha ou é uma variável de maior ponto fixo ou de menor ponto fixo. Desta forma, os nós remanescentes serão coloridos, em duas fases, de acordo com a testemunha, como segue:

(1) Testemunha de menor ponto fixo μ:
   (a) Colorir sucessivamente com $\bot$ cada configuração em F_i que obedeça uma das seguintes condições até que nenhuma delas possa mais ser aplicada:
      Nós da forma $AX\varphi$ com todos os filhos *must* coloridos com $\top$ ou $\bot$;
      Nós da forma $\varphi_1 \wedge \varphi_2$ com ambos os filhos coloridos com $\top$ ou $\bot$;
      Nós da forma $EX\varphi$ que têm um filho *may* colorido com $\top$ ou $\bot$;
      Nós da forma $\varphi_1 \vee \varphi_2$ com um filho colorido com $\bot$;
   (b) Colorir todos os nós remanescentes com F.

(2) Testemunha de maior ponto fixo ν:
   (a) Colorir sucessivamente com $\bot$ cada configuração em F_i que obedeça uma das seguintes condições até que nenhuma delas possa mais ser aplicada:
      Nós da forma $AX\varphi$ com pelo menos um filho *may* colorido com F ou $\bot$;
      Nós da forma $\varphi_1 \wedge \varphi_2$ com um dos filhos coloridos com $\bot$;
      Nós da forma $EX\varphi$ com todos os filhos *must* coloridos com F ou $\bot$;
      Nós da forma $\varphi_1 \vee \varphi_2$ com ambos os filhos coloridos com F ou $\bot$.
   (b) Colorir todos os nós remanescentes com $\top$.

3.2.2 Exemplo de Verificação de Modelos com Jogos para Lógica de 3 Valores

Consideremos, como exemplo, a fórmula CTL $AX((AG\neg m) \vee (AFm))$ equivalente a fórmula $AX((\nu Z.\neg m \wedge AXZ) \vee (\mu Y.m \vee AXY))$ em μ-calculus. A figura 3.4 apresenta a arena do jogo relativa a esta fórmula e o KMTS M apresentado na figura.

Na arena da figura 3.4 as configurações são coloridas com (F), $(\top)$ ou $(\bot)$. A arena está particionada em suas CFCMs identificadas por F_i, com $i \in \mathbb{N}^*$. Arestas *may* são identificadas por setas seccionadas e as *must* por setas sólidas. A arena está particionada em suas CFCMs F_i, onde i informa a ordem de coloração, com respeito a relação de ordem parcial $\leq$. Assim, colore-se primeiramente as CFCMs F_1, F_2, F_3 e F_4. Como todas elas contêm apenas uma configuração terminal, aplica-se somente as regras da primeira fase do algoritmo de coloração. As configurações terminais C_{18} e C_5 são coloridas com $\bot$, pois os literais m e $\neg m$ são indeterminados no estado s_1. As próximas CFCMs a serem coloridas são F_5 e F_6. Em F_5 colore-se primeiro C_{17} com $\top$, visto que Eva $(\exists)$

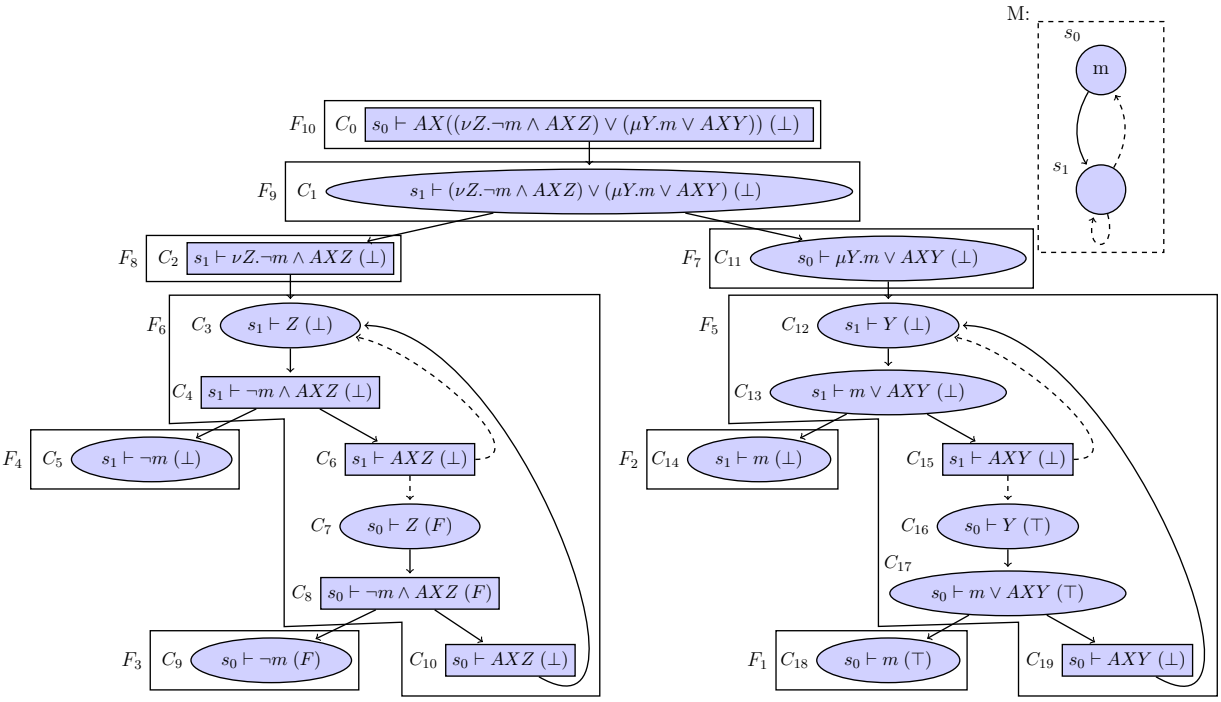

Figura 3.4: Exemplo de arena para jogo de três valores. Fonte. Adaptada de Guerra, Andrade e Wassermann (2013, p. 11).

tem estratégia de ganhar neste nó. A coloração é, então, propagada para C_{16}. Como mais nenhuma regra da primeira fase pode ser aplicada, passa-se para a segunda fase e colore-se, neste caso, todos os nós remanescentes com $\bot$. Ao final da coloração de F_5, passa-se a colorir as configurações em F_6. Finalmente, as CFCMs F_7, F_8, F_9 e F_{10} são coloridas; sendo aplicadas somente as regras da primeira fase. Ao final, a configuração C_0 é colorida com $\bot$. Logo, a verificação de modelos para o KMTS M da figura 3.4 resulta em indefinido.

3.2.3 Verificação de Modelos com 3-Valores e KMTS como um Conjunto de Estruturas de Kripke

A verificação de modelos proposta em (SHOHAM; GRUMBERG, 2007) quando aplicada sobre um KMTS interpretado como um conjunto de modelos (GUERRA; ANDRADE; WASSERMANN, 2013) fornece informações capazes de guiar o refinamento do modelo KMTS. Contudo, devido a diferença de semântica, as informações fornecidas pelo algoritmo de verificação não são suficientes para o refinamento completo de um KMTS interpretado como conjunto de modelos de Kripke. Mais precisamente, dado um KMTS M, interpretado como um conjunto de modelos, e uma fórmula CTL φ, verificar se M satisfaz φ a partir de um estado s_0 tem os seguintes resultados ao aplicarmos a verificação

de modelos proposta em (SHOHAM; GRUMBERG, 2007):

- $\top$, então todas as estruturas de Kripke expandidas de M satisfazem φ em s_0;
- F, então todas as estruturas de Kripke expandidas de M não satisfazem φ em s_0;
- $\bot$, nada se pode afirmar.

O resultado indefinido pode ocorrer, por exemplo, na verificação de certas propriedades, mesmo quando todos os modelos expandidos satisfazem a mesma. Então, o refinamento é realizado, e seria esperado que o algoritmo retornasse o próprio modelo de entrada, o que não ocorre. Ilustramos na figura 3.5 a arena de um jogo de verificação de modelos para um KMTS M e fórmula CTL $\varphi = (m \vee EX\neg m) \vee (\neg m \wedge AXp)$. Visto que a configuração raiz da arena está colorida com $\bot$, concluímos que a verificação de modelos resulta em indefinido. Contudo, se observamos os modelos do conjunto expansão de M, ilustrados na figura 3.6, verificamos que todos eles satisfazem a fórmula φ. Em outras palavras, o algoritmo de verificação de modelos apresentado não atende à semântica de um KMTS interpretado como conjunto de modelos.

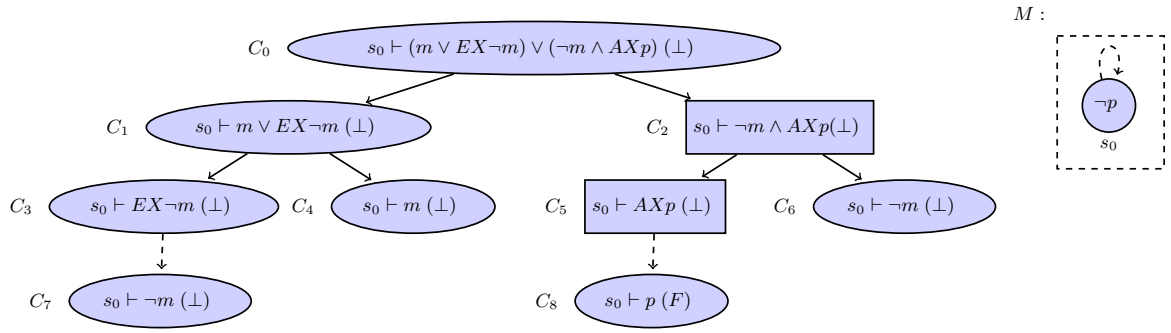

Figura 3.5: Arena do jogo de três valores colorida para a verificação de modelos um KMTS M e fórmula CTL $\varphi = (m \vee EX\neg m) \vee (\neg m \wedge AXp)$.

As informações fornecidas pelo verificador guiam o refinamento de modelos para um conjunto com mais de um modelo KMTS que se expandem nos mesmos modelos de Kripke expandidos do modelo original. Portanto, embora consistente, o resultado não

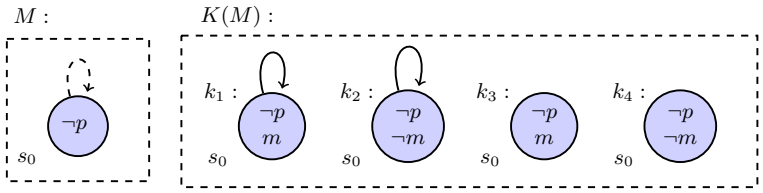

Figura 3.6: Exemplo de um KMTS M e conjunto $K(M)$ das estruturas de Kripke expandidas a partir de M.

corresponde ao definido pelo refinamento, uma vez que a solução não atende ao critério de minimalidade estabelecido. Nesse cenário, um verificador de modelos que atenda completamente à semântica adotada torna-se essencial para assistir integralmente o refinamento de modelos KMTSs. Nesse sentido, propomos no capítulo 5 um verificador de modelos para um KMTS interpretado como um conjunto de estruturas de Kripke. Este verificador de modelos atende à semântica adotada e fornece um suporte completo para o refinamento de modelos KMTSs.

Capítulo 4

REFINAMENTO DE MODELOS KMTS

O refinamento de modelos KMTSs é uma etapa da revisão de modelos (GUERRA; WASSERMANN, 2010) e ocorre quando a verificação de modelos resulta em indefinido. Neste caso, o resultado do refinamento é um conjunto de KMTSs que satisfazem a especificação e são obtidos através de mudanças minimais segundo uma relação de ordem parcial definida na seção 4.2.

Na seção 4.1, discorremos sobre operações de mudanças primitivas em relação a um KMTS e mudanças minimais. Na seção 4.3, apresentamos o conceito de refinamento de modelos KMTS e provamos que o refinamento admite solução única.

4.1 OPERAÇÕES DE MUDANÇAS

O trabalho (GUERRA; ANDRADE; WASSERMANN, 2013) apresenta três operações primitivas de mudanças que transformam as indeterminações de um KMTS em determinações, as quais apresentamos aqui como:

$P_1(s, s')$: remove o par (s, s') da relação R^-;
$P_2(s, s')$: transforma o par (s, s') em R^- para um par (s, s') em R^+;
$P_3(s, l)$: atribui o literal l ao estado s, se $l, \neg l \notin L(s)$.

A aplicação da operação $P_2(s_0, s_2)$, por exemplo, sobre o KMTS M da figura 4.1 gera o KMTS M_1 ilustrado na figura 4.2 que representa as estruturas de Kripke k_1, k_2, k_3 e k_4 presentes no conjunto expansão de M, denotado por $K(M)$. A aplicação da operação $P_3(s_2, p)$ atribui o literal p ao estado s_2 de M, e o modelo M_2 (figura 4.1) gerado por esta operação representa as estruturas de Kripke k_2, k_4, k_6 e k_8 presentes no conjunto expansão de M.

Os modelos do conjunto expansão de um KMTS são obtidos pela transformação das indeterminações deste em determinações, enquanto as determinações do mesmo são preservadas. Por isso, a aplicação das operações P_1, P_2 e P_3 sobre um KMTS gera um modelo cuja expansão é um subconjunto próprio daquele expandido do KMTS original, como podemos observar facilmente através dos exemplos apresentados.

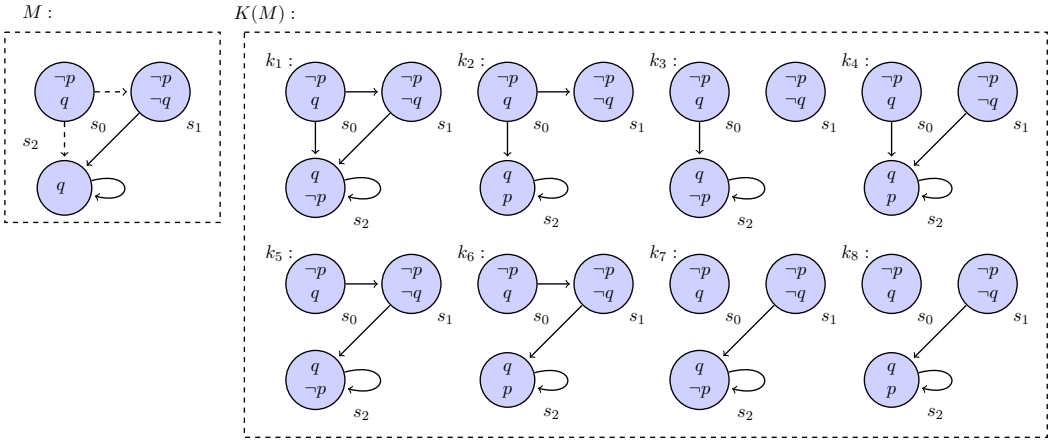

Figura 4.1: KMTS M e conjunto expansão $K(M)$.

Usamos o termo mudança para nos referir a um conjunto finito de operações primitivas do tipo P_1, P_2 e P_3 e escrevemos $M(X)$ para representar o modelo gerado pela aplicação de uma mudança X sobre um KMTS M. O modelo M_1 da figura 4.2, por exemplo, é gerado pela aplicação da mudança $X = \{P_2(s_0, s_2)\}$ sobre M, ou seja, $M_1 = M(X)$.

Definição 4.1.1. *Dados dois KMTSs M e M', dizemos que M' é instância de M, $M \sqsubseteq M'$ sse existir uma mudança X tal que $M' = M(X)$.*

Os modelos M_1 e M_2, gerados respectivamente pelas mudanças $X_1 = \{P_2(s_0, s_2)\}$ e $X_2 = \{P_2(s_0, s_1)\}$, são exemplos de instâncias de M.

Lema 4.1.1. *A função de aplicação de uma mudança X sobre um KMTS M, denotada por $M(X)$, é injetora.*

Prova. A prova segue por contradição. Suponhamos que a função de aplicação de mudanças sobre KMTSs não seja injetora. Logo, para algum KMTS $M = (AP, S, R^+, R^-, L)$, existem mudanças distintas X_1 e X_2 tais que $M(X_1) = M(X_2)$. As mudanças X_1 e X_2 transformam indeterminações de M em determinações. Por hipótese, $X_1 \neq X_2$, portanto existe pelo menos uma operação $\varrho \in X_1$ tal que $\varrho \notin X_2$ (ou vice versa). Sejam $s, u \in S$ estados quaisquer de M, e $l \in Lit$ um literal definido a partir do conjunto AP de átomos. Seja $\varrho \in X_1$ e $\varrho \notin X_2$:

(i) se $\varrho = P_1(s, u)$, então ϱ remove a transição $s \to u$ de M, ou seja, a instância $M(X_1)$ não tem a transição *may* $s \to u$. Contudo, como $\varrho \notin X_2$, X_2 não remove $s \to u$ e, portanto, ou $M(X_2)$ tem a transição *may* $s \to u$ (pois X_2 não tem ϱ) ou $M(X_2)$ tem a transição *must* $s \to u$ (neste caso X_2 tem a operação $P_2(s, u)$, que não está em X_1; pois se $P_2(s, u)$ ocorresse em X_1, este seria inconsistente). Concluímos que $M(X_1) \neq M(X_2)$ que contradiz a nossa hipótese.

(ii) se $\varrho = P_2(s, u)$, então ϱ transforma a transição $s \to u$ de M em uma transição *must*, ou seja, a transição $s \to u$ em $M(X_1)$ é uma transição *must*. Contudo, como $\varrho \notin X_2$,

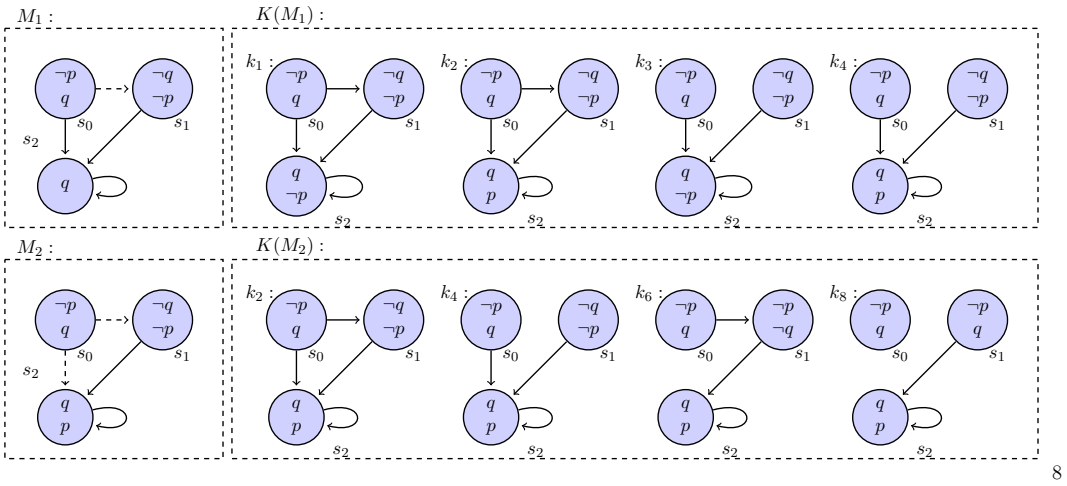

Figura 4.2: KMTSs M_1 e M_2 geradas respectivamente pela aplicação das operações de mudanças $P_2(s_0, s_1)$ e $P_3(s_1, p)$ sobre o KMTS M da figura 4.1, e os conjuntos expansão de M_1 e M_2.

a mudança X_2 não transforma $s \to u$ em transição *must* e, portanto, $M(X_2)$ ou tem a transição *may* $s \to u$, ou $M(X_2)$ não tem tal transição (neste caso, X_2 remove a transição $s \to u$ do modelo). Desta forma, concluímos que $M(X_1) \neq M(X_2)$ que contradiz a nossa hipótese.

(iii) se $\varrho = P_1(s, l)$, então ϱ atribui o literal l ao estado s de M, ou seja, $l \in L(s)$. Contudo, como $\varrho \notin X_2$, a mudança X_2 não atribui o literal l ao estado s de M. Logo, $M(X_2)$ ou tem o literal $\neg l$ atribuído ao estado s, ou nem l nem $\neg l$ estão atribuídos ao estado s. Nos dois casos, $l \notin L(s)$. Concluímos, portanto, que $M(X_1) \neq M(X_2)$ que contradiz a nossa hipótese.

Segue de *(i)* a *(iii)* que $M(X_1) \neq M(X_2)$ que contradiz a hipótese $M(X_1) = M(X_2)$. Concluímos, portanto, que $M(X)$ é injetora.

∎

Corolário 4.1.1. *Dados três KMTSs quaisquer M, M_1 e M_2, tal que $M_1 = M(X_1)$ e $M_2 = M(X_2)$; onde X_1 e X_2 são mudanças, $M_2 \sqsubseteq M_1$ sse $X_2 \subseteq X_1$.*

Prova. Tomemos $M_1 = M(X_1)$ e $M_2 = M(X_2)$ (I).

"⇒" Suponhamos $M_2 \sqsubseteq M_1$, ou seja, $\exists X'; M_1 = M_2(X')$.

M_2 é obtido aplicando-se a operação X_2 sobre M, e a aplicação de X' sobre M_2 gera o modelo M_1, ou seja, M_1 é gerado pela aplicação das mudanças $X_2 \cup X'$ sobre M. Logo, $M_1 = M(X' \cup X_2)$. Como $M_1 = M(X_1)$, temos que $M(X_1) = M(X' \cup X_2)$. Portanto, $X_1 = X' \cup X_2$, uma vez que $M(X)$ é injetora (lema 4.1.1). Concluímos assim que $X_2 \subseteq X_1$.

"$\Leftarrow$" Suponhamos $X_2 \subseteq X_1$. Logo, para algum X', $X_1 = X_2 \cup X'$.

Temos que $M_1 = M(X_1)$, portanto $M_1 = M(X_2 \cup X')$. Sabemos que $M_2 = M(X_2)$, assim $M_1 = M_2(X')$. Logo, existe uma mudança X' que aplicada a M_2 gera M_1. Concluímos que $M_2 \sqsubseteq M_1$.

∎

A aplicação de uma mudança sobre um KMTS, algumas vezes não é recomendada. Por exemplo, não podemos aplicar simultaneamente as operações $P_3(s, l)$ e $P_3(s, \neg l)$ sobre um KMTS, nem aplicar $P_1(s, s')$ e $P_2(s, s')$, pois não podemos definir um KMTS com uma transição $s \rightarrow s'$ pertencente a R^+, mas que não pertença a R^-. Chamamos estas operações de complementares.

Definição 4.1.2. *Seja ϱ uma operação primitiva qualquer, a operação complementar de ϱ, denotada por $\neg\varrho$ é definida como:*

(i) $\varrho = P_2(s, l)$ *sse* $\neg\varrho = P_2(s, \neg l)$

(ii) $\varrho = P_1(s, s')$ *sse* $\neg\varrho = P_2(s, s')$

Dizemos que duas mudanças são incompatíveis se ambas têm pelo menos uma operação complementar da outra.

Definição 4.1.3. *Uma mudança X é incompatível com uma mudança Y, denotado por $X \not\simeq Y$, sse existe uma operação $\varrho \in X$ tal que $\neg\varrho \in Y$.*

Se duas mudanças X e Y não são incompatíveis, então dizemos que X e Y são compatíveis e escrevemos $X \simeq Y$.

Corolário 4.1.2. *Seja M um KMTS qualquer, $M_1 = M(X_1)$ e $M_2 = M(X_2)$ instâncias de M. Se $X_1 \not\simeq X_2$, então $K(M_1) \cap K(M_2) = \emptyset$.*

Prova. Suponhamos que $X_1 \not\simeq X_2$. Logo, existe uma operação $\varrho \in X_1$ tal que $\neg\varrho \in Y$.

1. $\varrho = P_3(s, l)$: $\forall k \in K(M_1)$, $l \in L(s)$ e $\forall k \in K(M_2), \neg l \in L(s)$. Logo, $K(M_1) \cap K(M_2) = \emptyset$;

2. $\varrho = P_1(s, s')$: $\forall k \in K(M_1), (s, s') \notin R^-$ e $\forall k \in K(M_2), (s, s') \in R^- \setminus R^+$. Logo, $K(M_1) \cap K(M_2) = \emptyset$;

3. $\varrho = P_1(s, s')$: análogo ao item anterior.

Concluímos, portanto, que $K(M_1) \cap K(M_2) = \emptyset$.

∎

Lembramos que as indefinições de um KMTS são caracterizadas por transições *may* genuínas (transições *may* que não são *must*) ou por um par de literais complementares não definidos em algum estado s do modelo, i.e, literais $l, \neg l \notin L(s)$. As indeterminações de um KMTS são dadas pelo conjunto $Ind(M)$ (definição 4.1.4).

Definição 4.1.4. *Sejam $M = (AP, S, R^+, R^-, L)$ um KMTS qualquer e Lit o conjunto dos literais de AP, o conjunto das indeterminações de M é definido como*

$$Ind(M) = (R^- \setminus R^+) \cup \{(s, w) \mid s \in S \text{ e } w, \neg w \in Lit \text{ e } w, \neg w \notin L(s)\}.$$

Temos duas opções para transformar uma indeterminação de um KMTS em uma determinação: se for uma transição *may* genuína ($R^- \setminus R^+$), podemos ou removê-la ou transformá-la em uma transição *must*; caso seja um literal l indeterminado em um estado do modelo ($l, \neg l \notin L(s)$, onde s é um estado do modelo), podemos definir ou l ou $\neg l$ no respectivo estado. Logo, para cada indeterminação, temos duas operações primitivas que podem ser aplicadas para transformá-la em uma determinação. A função *chg*, definição 4.1.5 a seguir, mapeia cada indeterminação ao conjunto das operações primitivas que as removem. Vale ressaltar que entre as duas operações primitivas, apenas uma entre elas pode ser escolhida para remover a respectiva indeterminação, visto que são operações complementares.

Definição 4.1.5. *Sejam $M = (AP, S, R^+, R^-)$ um KMTS qualquer, Lit o conjunto dos literais de AP e Ω o conjunto de todas as operações primitivas aplicáveis sobre M. Definimos a função $chg : Ind(M) \to 2^{\Omega}$ que mapeia cada indeterminação de M a um conjunto de operações primitivas:*

$$chg(s,u) = \begin{cases} \{P_1(s,u), P_2(s,u)\} & sse\ (s,u) \in R^- \setminus R^+; \\ \{P_3(s,u), P_3(s,\neg u)\} & sse\ u, \neg u \in Lit \text{ e } u, \neg u \notin L(s). \end{cases}$$

A proposição a seguir associa o conjunto expansão de dois KMTSs ao conjunto de indeterminações destes KMTSs com respeito à relação de inclusão.

Proposição 4.1.1. *Sejam M_1 e M_2 dois KMTSs quaisquer, se $K(M_1) \subseteq K(M_2)$ então $Ind(M_1) \subseteq Ind(M_2)$.*

Prova. Consideramos $M_w \in \{M_1, M_2\}$. Todo modelo em $K(M_w)$ é uma instância de M_w, ou seja, $\forall k_i \in K(M_w)\ \exists X_i; k_i = M_w(X_i)$. Seja $k_i \in K(M_w)$ uma estrutura de Kripke qualquer e X_i a mudança que gera k_i, ou seja, $k_i = M_w(X_i)$. Para toda indeterminação $t \in Ind(M_w)$, existem operações ϱ e $\neg\varrho$ aplicáveis sobre M_w. Logo,

$$\forall k_i \in K(M_w),\ \text{ou } \varrho \in X_i \text{ ou } \neg\varrho \in X_i \tag{4.1}$$

$$\forall (s,u) \in Ind(M_w)\ \exists k_i, k_j \in K(M_w); \varrho, \neg\varrho \in chg(s,u) \to \varrho \in X_i \text{ e } \neg\varrho \in X_j. \tag{4.2}$$

Façamos $\xi(\varrho) = \{k \in K(M_w) \mid k = M_w(X) \text{ e } \varrho \in X\}$ o conjunto de todas as estruturas de Kripke em $K(M_w)$ que foram obtidas pela aplicação de pelo menos uma operação primitiva ϱ. De (4.1), temos que

$$\forall (s,u) \in Ind(M_w);\ \varrho, \neg\varrho \in chg(s,u) \to K(M_w) = \xi(\varrho) \cup \xi(\neg\varrho) \tag{4.3}$$

Do enunciado, $K(M_1) \subseteq K(M_2)$, ou seja, $\forall k_i \in M_1,\ k_i \in K(M_2)$. Portanto,

$$\forall (s,u) \in Ind(M_1) \exists k_i, k_j \in K(M_2);\ \varrho, \neg\varrho \in chg(s,u) \rightarrow k_i \in \xi(\varrho) \text{ e } k_j \in \xi(\neg\varrho).$$

Para toda indeterminação $t \in Ind(M)$, existem modelos k_i e k_j em $K(M_2)$ obtidos por operações complementares, portanto M_2 também tem a indeterminação t; caso contrário, apenas k_i ou k_j ocorreria em $K(M_2)$. Logo, $\forall t \in Ind(M_1),\ t \in Ind(M_2)$. Ou seja,

$$Ind(M_1) \subseteq Ind(M_2)$$

∎

Proposição 4.1.2. *Um KMTS M_1 é instância de um KMTS M_2 sse $K(M_1) \subseteq K(M_2)$*

Prova. "⇒" Suponhamos $M_2 \sqsubseteq M_1$, logo $\exists X'; M_1 = M_2(X')$.

Temos que $\forall k_i \in K(M_1)\ \exists X_i;\ k_i = M_1(X_i)$. Fixemos X_i como a mudança que gera k_i a partir de M_1. Por hipótese, $M_1 = M_2(X')$. Logo,

$$\forall k_i \in K(M_1),\ k_i = M_2(X_i \cup X').$$

Concluímos que $\forall k_i \in K(M_1),\ k_i \in K(M_2)$, ou seja, $K(M_1) \subseteq K(M_2)$.

"⇐" Suponhamos $K(M_1) \subseteq K(M_2)$.

Toda instância k_i em $K(M_1)$ é obtida pela aplicação de uma mudança X_i sobre M_1 que transforma todas as indeterminações de M_1 em determinações. Seja $k_i \in K(M_1)$ uma estrutura de Kripke, e X_i a mudança que gera k_i, ou seja, $k_i = M_1(X_i)$. Segue da proposição 4.1.1 que $Ind(M_1) \subseteq Ind(M_2)$. Portanto, toda mudança aplicável a M_1 é aplicável a M_2.

Façamos $M_i' = M_2(X_i)$, a instância obtida pela aplicação de X_i sobre M_2. A aplicação de X_i remove todas as indeterminações de M_1 visto que gera a estrutura de Kripke $k_i \in K(M_1)$, porém a aplicação X_i sobre M_2 pode não remover todas as indeterminações de M_2, pois $Ind(M_1) \subseteq Ind(M_2)$. Por hipótese, segue que $\forall k_i \in K(M_1),\ k_i \in K(M_2)$, existe uma mudança X' que aplicada a M_i' gera k_i, ou seja, $k_i = M_i'(X')$. Desta forma, $k_i = M_2(X_i \cup X')$. Apliquemos X' sobre M_2 a fim de gerar um modelo $M_3 = M_2(X')$. Logo, $k_i = M_3(X_i)$. Apresentamos na figura 4.3 os modelos gerados pelas aplicações das mudanças X_i e X' sobre os modelos aqui citados.

Lembramos que $k_i = M_1(X_i)$. Assim, $M_3 = M_1 = M_2(X')$. Concluímos, portanto, que $M_2 \sqsubseteq M_1$.

∎

A proposição a seguir associa as mudanças de dois KMTSs ao conjunto expansão destes KMTSs com respeito à relação de inclusão.

Proposição 4.1.3. *Seja M um KMTS qualquer, e $M_1 = M(X_1)$ e $M_2 = M(X_2)$ instâncias de M. $X_1 \subseteq X_2$ sse $K(M_2) \subseteq K(M_1)$.*

Prova. Segue direto do corolário 4.1.1 e da proposição 4.1.2. ∎

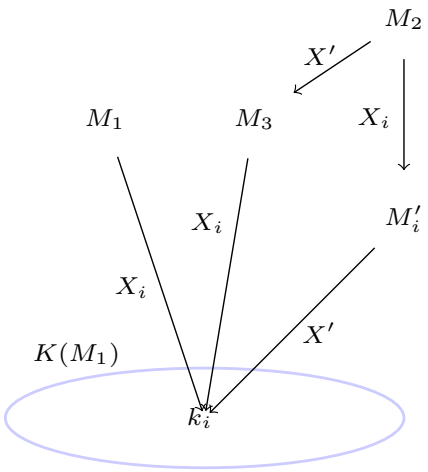

Figura 4.3: Diagrama de aplicações de mudanças sobre os modelos M_1 e M_2 da prova 4.1.2 que geram uma instância k_i qualquer em $K(M_1)$.

4.2 MUDANÇA MINIMAL E INSTÂNCIA MAXIMAL

O refinamento de modelos KMTSs é definido sobre um critério de mudança minimal, no sentido que os modelos do refinamento conservam o maior número possível de informações do KMTS original. Em (GUERRA; ANDRADE; WASSERMANN, 2013), o critério de mudança minimal é definido segundo a relação de ordem parcial de inclusão $\subseteq$.

Definição 4.2.1. *Seja β um conjunto qualquer de mudanças, uma mudança $X_1 \in \beta$ é minimal em relação a β sse não existir uma mudança $X_2 \in \beta$ tal que $X_2 \subset X_1$.*

Visto que um KMTS é um modelo finito de transições de estados, um conjunto β de mudanças aplicáveis sobre um KMTS sempre é finito e, por ser um conjunto parcialmente ordenado com respeito à relação de ordem parcial $\subseteq$, β tem pelo menos um elemento minimal se for não vazio. Usaremos $Min(\beta)$ para denotar o conjunto de todos os elementos minimais de um conjunto β de mudanças.

Definição 4.2.2. *Seja β um conjunto de instâncias de um KMTS M qualquer, uma instância $M_1 \in \beta$ é maximal em relação a β sse não existir uma instância $M_2 \in \beta$ tal que $M_2 \sqsubseteq M_1$ e $M_2 \neq M_1$.*

Denotamos o conjunto de todas as instâncias maximais de um conjunto β por $Max(\beta)$.

O lema 4.2.1 é utilizado na prova da proposição 4.2.1 e estabelece que dado um conjunto β de mudanças particionado em dois conjuntos, α e γ, se nenhuma mudança em α estiver contida em outra mudança também em α, e para cada mudança em γ existir uma mudança menor em α, então α é o conjunto das mudanças minimais de β com respeito à relação de inclusão ($\subset$).

Lema 4.2.1. *Dado um conjunto β de mudanças e um subconjunto α de β. Se $\forall X_j \in \beta \setminus \alpha, \exists X_i \in \alpha; X_i \subseteq X_j$ e $\forall X_i, X_j \in \alpha, X_i \not\subseteq X_j$, então $\forall X_i \in \alpha, \nexists X_j \in \beta; X_j \subset X_i$.*

Prova. A prova segue por contradição. Suponhamos que

$$\forall X_j \in \beta \setminus \alpha, \exists X_i \in \alpha; X_i \subseteq X_j \tag{4.4}$$
$$\forall X_i, X_j \in \alpha, X_i \not\subseteq X_j \tag{4.5}$$
$$\exists X_i \in \alpha, \exists X_j \in \beta; X_j \subset X_i \tag{4.6}$$

Da última expressão acima, temos que existe uma mudança $X_k \in \alpha$ e uma mudança $X'_k \in \beta$ tal que $X'_k \subset X_k$. De (4.5), $X'_k \notin \alpha$, logo $X'_k \in \beta \setminus \alpha$. De (4.4), segue que

$$\exists X_j \in \alpha; X_j \subseteq X'_k \tag{4.7}$$

Da expressão acima e visto que $X'_k \subseteq X_k$, temos que $\exists X_j \in \alpha; X_j \subseteq X_k$ que contradiz a nossa hipótese em (4.5). Concluímos, portanto, que $\forall X_i \in \alpha$, $\nexists X_j \in \beta; X_j \subset X_i$.

∎

Mudanças minimais produzem instâncias maximais e vice-versa, no sentido que quanto menor a mudança maior o modelo produzido por ela.

Proposição 4.2.1. *Seja β um conjunto de instâncias de um KMTS M qualquer e $\chi = \{X_i \mid M(X_i) \in \beta\}$ o conjunto das mudanças que geram as instâncias em β. Um KMTS $M_i = M(X_i) \in \beta$ é maximal sse $X_i \in \chi$ é minimal.*

Prova. Segue do enunciado que

$$\forall M_j \in \beta \setminus Max(\beta),\ \exists M_i \in Max(\beta);\ M_i \sqsubseteq M_j \tag{4.8}$$

$$M(X_i) \in \beta \Leftrightarrow X_i \in \chi \tag{4.9}$$

Tomemos o conjunto $\alpha = \{X_i \in \chi \mid M(X_i) \in Max(\beta)\}$ das mudanças que geram os modelos maximais em β. Portanto,

$$M(X_i) \in Max(\beta) \Leftrightarrow X_i \in \alpha \tag{4.10}$$

$$M(X_j) \in \beta \setminus Max(\beta) \Leftrightarrow X_j \in \chi \setminus \alpha \tag{4.11}$$

De (4.8) e do corolário (4.1.1),

$$\forall M(X_j) \in \beta \setminus Max(\beta),\ \exists M(X_i) \in Max(\beta);\ X_i \subseteq X_j \tag{4.12}$$

pois, $M_i \sqsubseteq M_j$, e M_i e M_j são instâncias de M.
Aplicando (4.10) e (4.11) em (4.12), temos

$$\forall X_j \in \chi \setminus \alpha, \exists X_i \in \alpha;\ X_i \subseteq X_j \tag{4.13}$$

De acordo com a definição de $Max(\beta)$, $\forall M_i, M_j \in Max(\beta),\ M_j \not\sqsubseteq M_i$; e do corolário 4.1.1 segue que $\forall M(X_i), M(X_j) \in Max(\beta),\ X_j \not\subseteq X_i$. Logo, de (4.10), temos

$$\forall X_i, X_j \in \alpha,\ X_i \not\subseteq X_j$$

Temos do resultado acima, de (4.13) e do lema 4.2.1 que

$$\forall X_i \in \alpha,\ \nexists X_j \in \chi;\ X_j \subset X_i \tag{4.14}$$

Em outras palavras, toda mudança em α é minimal, ou seja, $\alpha = Min(\chi)$. Segue de (4.10) que $M(X_i) \in Max(\beta) \Leftrightarrow X_i \in Min(\chi)$. Concluímos, portanto, que $M(X_i) \in \beta$ é maximal sse $X_i \in \chi$ for minimal.

∎

4.3 REFINAMENTO DE MODELOS KMTS

Refinar um KMTS M em relação a uma fórmula CTL φ consiste em selecionar o conjunto das estruturas de Kripke expandidas de M que satisfazem φ. Este conjunto deve ser representado por instâncias de M geradas por mudanças minimais, ou seja, pelas instâncias mais próximas de M.

Para as definições a seguir, denotamos por $I(M, \alpha) = \{M \sqsubseteq M' \mid K(M') \subseteq \alpha\}$, o conjunto de todas as instâncias de M que representam um conjunto $\alpha \subseteq K(M)$; e $\chi(M, \alpha) = \{X_i \mid M(X_i) \in I(M, \alpha)\}$ o conjunto das mudanças que geram as instâncias em $I(M, \alpha)$.

Problema 4.3.1 (Refinamento KMTS)**.** *Dados um KMTS M qualquer e $\alpha \subseteq K(M)$ um subconjunto dos modelos expandidos de M, determinar um conjunto β de instâncias de M que represente o conjunto das estruturas de Kripke em α, tal que toda instância em β é gerada por uma mudança minimal. Ou seja,*

$$\beta \subseteq I(M, \alpha) \qquad \forall M(X_i) \in \beta,\ X_i \in Min(\chi(M, \alpha))$$
$$\alpha = \bigcup_{M_i \in \beta} K(M_i)$$

As estruturas de Kripke k_1, k_4, k_5 e k_6 representadas pelo KMTS M da figura 4.1 são as únicas que satisfazem a fórmula CTL $\varphi = EX(p \wedge q) \vee EX(\neg q)$. O refinamento de M em relação ao conjunto $\alpha = \{k_1, k_4, k_5, k_6\}$ resulta no conjunto de instâncias $\beta = \{M_3, M_4\}$ ilustradas na figura 4.4. As mudanças $X_3 = \{P_2(s_0, s_1)\}$ e $X_4 = \{P_2(s_0, s_2), P_3(s_2, p)\}$ aplicadas a M geram respectivamente os modelos M_3 e M_4. Claramente, não existe mudança menor que X_3 que aplicada a M gere um modelo cujo conjunto expansão seja subconjunto de α. Quanto a X_4, as únicas mudanças menores são $X_1 = \{P_2(s_0, s_2)\}$ e $X_2 = \{P_3(s_2, p)\}$ que aplicadas a M geram respectivamente os modelos M_1 e M_2 da figura 4.2. Porém, a estrutura de Kripke k_1 expandida a partir de M_1 não pertence a α, enquanto M_2 representa o modelo k_8 que também não pertence a α. Portanto, β é uma solução do refinamento de M em relação a α. De fato, não existe nenhuma outra solução para este refinamento; pois, como enunciamos no teorema 4.3.2, a solução do refinamento é única.

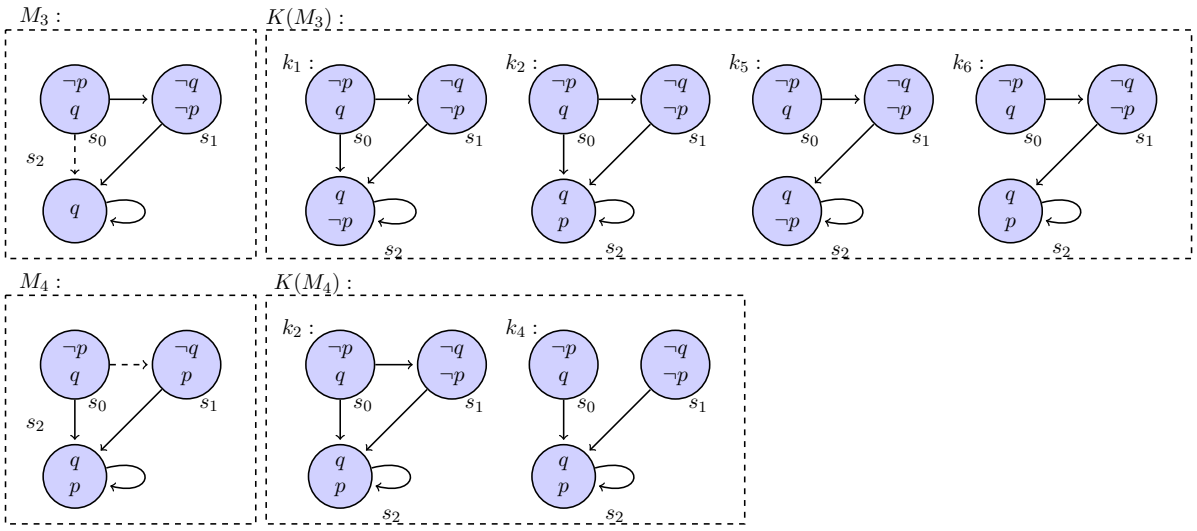

Figura 4.4: Instâncias M_3 e M_4, geradas respectivamente pela aplicação das mudanças $X_3 = \{P_2(s_0, s_1)\}$ e $X_4 = \{P_2(s_0, s_2), P_3(s_2, p)\}$ sobre o KMTS M da figura 4.1, e respectivos conjuntos expansão.

Teorema 4.3.1. *Seja M um KMTS qualquer e $\alpha \subseteq K(M)$ um conjunto não vazio. O conjunto $Max(I(M, \alpha))$ é solução do refinamento de M em relação a α.*

Prova. Suponhamos $\beta = Max(I(M, \alpha))$. O conjunto β é solução do refinamento de M em relação a α sse

(1) $\beta \subseteq I(M, \alpha)$:

Temos que $\beta = Max(I(M, \alpha))$ e $Max(I(M, \alpha)) \subseteq I(M, \alpha)$, logo $\beta \subseteq I(M, \alpha)$.

(2) $\forall M(X_i) \in I(M, \alpha),\ X_i \in Min(\chi(M, \alpha))$:

Pela definição de $\chi(M, \alpha)$, $M(X_i) \in I(M, \alpha))$ sse $X_i \in \chi(M, \alpha)$. Conforme a proposição 4.2.1, $M(X_i) \in Max(I(M, \alpha))$ sse $X_i \in Min(\chi(M, \alpha))$. Visto que $\beta = Max(I(M, \alpha))$,

$$M(X_i) \in \beta \Leftrightarrow X_i \in Min(\chi(M, \alpha))$$

$\therefore\ \forall M(X_i) \in \beta,\ X_i \in Min(\chi(M, \alpha))$.

(3) $\alpha = \bigcup_{M_i \in \beta} K(M_i)$:

A prova segue por contradição. Suponhamos que $\alpha \neq \bigcup_{M_i \in \beta} K(M_i)$, ou seja

$$\exists k \in \alpha \text{ tal que } \forall M_i \in \beta, k \notin K(M_i) \tag{4.15}$$

Como $\beta = Max(I(M,\alpha))$,

$$\forall M_i \in I(M,\alpha) \setminus \beta,\ \exists M_j \in \beta;\ M_j \sqsubseteq M_i \tag{4.16}$$

Façamos $\gamma = \{\beta' \subseteq I(M,\alpha) \mid \alpha = \bigcup_{M_i \in \beta'} K(M_i)\}$ o conjunto dos subconjuntos de $I(M,\alpha)$ que representa α. O conjunto γ é não vazio, visto que α é um conjunto de estruturas de Kripke. Tomemos um conjunto $\beta' \in \gamma$. Claramente $\beta' \neq \beta$, pois β não representa α, por hipótese. Logo,

$$\forall k \in \alpha, \exists M_i \in \beta' \text{ tal que } k \in K(M_i) \tag{4.17}$$

De (4.15), existe pelo menos um k em α que não é representado por nenhum M_j em β. Portanto, a partir de (4.15) e (4.17) concluímos que

$$\exists k \in \alpha, \exists M_i \in \beta' \text{ tal que } \forall M_j \in \beta,\ k \in K(M_i) \text{ e } k \notin K(M_j) \tag{4.18}$$

Logo,

$$\exists M_i \in \beta', \forall M_j \in \beta \text{ tal que } K(M_i) \not\subseteq K(M_j) \tag{4.19}$$

Do resultado acima e da proposição 4.1.2,

$$\exists M_i \in \beta', \forall M_j \in \beta \text{ tal que } M_j \not\sqsubseteq M_i \tag{4.20}$$

Temos, portanto, um elemento M_i em β' que não ocorre em β, ou seja, $M_i \in I(M,\alpha) \setminus \beta$. Logo,

$$\exists M_i \in I(M,\alpha) \setminus \beta, \forall M_j \in \beta \text{ tal que } M_j \not\sqsubseteq M_i \tag{4.21}$$

que contradiz o enunciado em (4.16). Concluímos, portanto, que $\alpha = \bigcup_{M_i \in \beta} K(M_i)$

∎

Antes de enunciarmos e provarmos que a solução do refinamento de um modelo KMTS é única, consideramos as operações e lemas apresentadas a seguir as quais serão utilizadas na prova do teorema 4.3.2.

Definição 4.3.1. *Seja Γ um conjunto de mudanças, e X e Y mudanças quaisquer. Definimos as seguintes operações:*

$$\overline{X} = \{\ \{\neg\varrho\} \mid \varrho \in X\}$$

$$X - Y = \begin{cases} \{X\} & \textit{sse } X \not\simeq Y \\ \{X \cup \{\neg\varrho\} \mid \varrho \in Y \textit{ e } \varrho \notin X\} & \textit{caso contrário} \end{cases}$$

$$\Gamma \setminus\setminus Y = \bigcup_{X_i \in \Gamma} X_i - Y.$$

A operação $X - Y$ expande minimamente[1] X em um conjunto de mudanças compatíveis com X e incompatíveis com Y, enquanto a operação $\Gamma \between Y$ expande minimamente as mudanças em Γ em um conjunto de mudanças incompatíveis com Y. Ilustramos as operações através de um exemplo:

$$X_1 = \{a, b\} \Rightarrow \overline{X_1} = \{\ \{\neg a\}, \{\neg b\}\ \} \qquad X_2 = \{c, d\} \Rightarrow \overline{X_2} = \{\ \{\neg c\}, \{\neg d\}\ \}$$
$$X_3 = \{d, e\} \Rightarrow \overline{X_3} = \{\ \{\neg d\}, \{\neg e\}\ \}$$

onde a, b, c, d, e são operações primitivais quaisquer.
Como $X_1 \simeq X_3$ e $X_2 \simeq X_3$, segue que

$$X_1 - X_3 = \{\ \{a, b, \neg d\}, \{a, b, \neg e\}\ \} \qquad X_2 - X_3 = \{\ \{c, d, \neg e\}\ \}$$
$$\{X_1, X_2\} \between X_3 = (X_1 - X_3) \cup (X_2 - X_3)$$
$$= \{\ \{a, b, \neg d\}, \{a, b, \neg e\}, \{c, d, \neg e\}\ \}$$

Definição 4.3.2. *A função complemento de um conjunto não vazio de mudanças é definida como:*

$$Comp(\{X\}) = \overline{X}$$
$$Comp(\{X_0, \ldots, X_n, X_{n+1}\}) = Comp(\{X_0, \ldots, X_n\}) \between X_{n+1}.$$

A função complemento de um conjunto Γ de mudanças, $Comp(\Gamma)$, calcula um conjunto de mudanças complementares às mudanças em Γ. $Comp(\Gamma)$ expande minimamente as mudanças em Γ de forma a obter um conjunto Γ' cujas mudanças são incompatíveis com aquelas em Γ. Para o conjunto $\Gamma = \{X_1, X_2\}$ das mudanças do exemplo anterior, temos que:

$$\begin{aligned} Comp(\Gamma) &= Comp(X_1) \between X_2 \\ &= \overline{X_1} \between \{c, d\} \\ &= \{\ \{\neg a\}, \{\neg b\}\ \} \between \{c, d\} \\ &= (\{\neg a\} - \{c, d\}) \cup (\{\neg b\} - \{c, d\}) \\ &= \{\ \{\neg a, \neg c\}, \{\neg a, \neg d\}\ \} \cup \{\ \{\neg b, \neg c\}, \{\neg b, \neg d\}\ \} \\ &= \{\ \{\neg a, \neg c\}, \{\neg a, \neg d\}, \{\neg b, \neg c\}, \{\neg b, \neg d\}\ \} \end{aligned}$$

Lema 4.3.1. *Seja Γ um conjunto não vazio de mudanças, $\forall X \in \Gamma$, $\forall Y \in Comp(\Gamma)$ $X \not\simeq Y$.*

[1] Dizemos que um conjunto de mudanças é expandido minimamente com relação a uma propriedade P sse o conjunto γ de mudanças, obtido após a expansão, detém a propriedade P e γ for subconjunto de qualquer outro conjunto de mudanças que detém a propriedade P.

Prova. Apresentada no Apêndice A. ∎

Lema 4.3.2. *Seja Γ um conjunto não vazio de mudanças e Y uma mudança qualquer. Se $\forall X \in \Gamma$, $X \cap Y = \emptyset$ e $Y \simeq X$, então $\forall X \in Comp(\Gamma), X \simeq Y$.*

Prova. Apresentada no Apêndice A. ∎

Teorema 4.3.2. *Seja M um KMTS qualquer e $\alpha \subseteq K(M)$ um conjunto não vazio. A solução do refinamento de M em relação a α é única.*

Prova. Do teorema 4.3.1, $\beta = Max(I(M, \alpha))$ é solução do refinamento de M em relação a α. Suponhamos, por contradição, que β não seja a única solução deste refinamento, ou seja, existe um conjunto β' diferente de β que é solução do refinamento. Logo, $\exists M_i \in \beta';\ M_i \notin \beta$ ou $\exists M_i \in \beta;\ M_i \notin \beta'$

(1) $\exists M_i \in \beta';\ M_i \notin \beta$

Como β' é solução do refinamento, $\forall M(X_i) \in \beta',\ X_i \in Min(\chi(M, \alpha))$. Da proposição 4.2.1, $M(X_i) \in \beta$ sse $X_i \in Min(\chi(M, \alpha))$. Assim, $\forall M_i \in \beta',\ M_i \in \beta$ que contradiz nossa hipótese.

(2) $\exists M_i \in \beta;\ M_i \notin \beta'$

Da prova em (1), temos que $\forall M_i \in \beta',\ M_i \in \beta$. Portanto, β' é subconjunto próprio de β. Fixemos um $M_i = M(X_i)$ pertencente a β que não ocorre em β'. Como $M_i \in \beta$ e β é solução do refinamento, então $K(M_i) \subseteq \alpha$ e $K(M_i) \subseteq \bigcup_{X_j \in \beta'} K(M_j)$.

Temos assim que

$$\alpha = \bigcup_{M_j \in \beta'} K(M_j) \qquad\qquad \alpha = K(M_i) \cup \bigcup_{M_j \in \beta'} K(M_j)$$

Temos, portanto, que $\forall k \in K(M(X_i)),\ \exists M(X_j) \in \beta'$ tal que $k \in K(M(X_j))$. Fazendo-se $\gamma = \{X_j \in \chi(M, \alpha) \mid M(X_j) \in \beta' \text{ e } K(M(X_i)) \cap K(M(X_j)) \neq \emptyset\}$ o conjunto de mudanças que geram modelos em β' que tem interseção com a expansão do modelo M_i. Logo,

$$K(M(X_i)) \subseteq \bigcup_{X_j \in \gamma} K(M(X_j)) \tag{4.22}$$

Mais especificamente,

$$\forall k \in K(M(X_i)),\ \exists X_j \in \gamma;\ k \in K(M(X_j)). \tag{4.23}$$

Segue da definição de γ que

$$X_j \in \gamma \Leftrightarrow K(M(X_i)) \cap K(M(X_j)) \neq \emptyset \tag{4.24}$$

Da expressão acima e pela contrapositiva do corolário 4.1.2,

$$\forall X_j \in \gamma, X_j \simeq X_i \tag{4.25}$$

Tomemos o conjunto $\omega = \{X_j \setminus X_i \mid X_j \in \gamma\}$. Logo,

$$\forall X_j \in \omega, \exists X_k \in \gamma; X_j \subset X_k \tag{4.26}$$

Da expressão acima e de (4.25), segue que

$$\forall X_j \in \omega, X_i \simeq X_j \tag{4.27}$$

Tomemos uma mudança $Y \in Comp(\omega)$ qualquer. Como $\forall X_j \in \omega, X_i \cap X_j = \emptyset$ e $X_i \simeq X_j$, segue respectivamente dos lemas 4.3.1 e 4.3.2 que

$$\forall X_j \in \omega, X_j \not\simeq Y \tag{4.28}$$
$$X_i \simeq Y \tag{4.29}$$

Consideremos a mudança $Z = X_i \cup Y$. Segue das duas expressões acima que

$$\forall X_j \in \omega, Z \not\simeq X_j \tag{4.30}$$

Como $X_j \setminus X_i = X'_j$, segue da diferença de conjuntos que

$$X_j \in \gamma \text{ sse } \exists X'_j \in \omega \text{ e } X_j = X'_j \cup (X_j \cap X_i).$$

Portanto,

$$\forall X_j \in \gamma, \exists X_k \in \omega; X_k \subset X_j.$$

Da expressão acima e de (4.28), temos

$$\forall X_j \in \gamma, X_j \not\simeq Y \tag{4.31}$$

da expressão acima e de (4.30), segue que

$$\forall X_j \in \gamma, X_j \not\supset Z. \tag{4.32}$$

Logo, $\forall X_j \in \gamma, K(M(X_j)) \cap K(M(Z)) = \emptyset$ que implica em

$$\exists k \in K(M(Z)); \forall X_j \in \gamma, k \notin K(M(X_j)) \tag{4.33}$$

Como $X_i \subset Z$, segue da proposição 4.1.3

$$K(M(Z)) \subset K(M(X_i)). \tag{4.34}$$

Da expressão acima e de (4.33), segue que

$$\exists k \in K(M(X_i)); \forall X_j \in \gamma, k \notin K(M(X_j))$$

que contradiz o exposto em 4.23.

Concluímos de (1) e (2), portanto, que β é a única solução do refinamento de um KMTS M em relação a um conjunto $\alpha \subseteq K(M)$.

∎

Em nosso processo, especificamente, o refinamento de um KMTS diz respeito a um conjunto α de estruturas de Kripke que satisfazem uma fórmula φ dada. A princípio, α é desconhecido e não vazio, uma vez que o refinamento só ocorre quando a verificação de modelos resulta em indefinido. Determinar α previamente a fim de encontrar a solução do refinamento é inviável, uma vez que realizamos a verificação de modelos sobre um KMTS, e não sobre as estruturas de Kripke por ele representadas. Por esta razão, lidamos diretamente com o KMTS e encontramos o resultado do refinamento sem precisar determinar α. Em outras palavras, deseja-se encontrar a solução do problema sem confrontá-la com o conjunto das estruturas de Kripke que satisfazem a propriedade especificada. Para tanto, podemos considerar a verificação de modelos com jogos de três valores (capítulo 3) para encontrar as mudanças que devem ser aplicadas sobre o KMTS original, gerando o conjunto solução do refinamento. Esta abordagem apresenta alguns desafios e apresentamos no capítulo 6 a nossa proposta para encontrar os modelos do refinamento.

Capítulo

5

VERIFICAÇÃO DE MODELOS POR CONTRAÇÃO

Não consta na literatura um verificador de modelos para lógica CTL cujos modelos KMTS são interpretados como um conjunto de estruturas de Kripke. Neste sentido, propomos um verificador de modelos adequado a esta interpretação.

Definimos inicialmente na seção 5.1 a semântica da lógica CTL em função dos modelos KMTSs interpretados como conjunto de estruturas de Kripke. Nas seções subsequentes, definimos operações sobre KMTSs e conjuntos partições de KMTS. Na seção 5.2, definimos a operação de contração sobre KMTSs que quando aplicada a conjuntos partições de KMTSs detém propriedades úteis para a definição da Verificação de Modelos por Contração que propomos neste capítulo. Os resultados apresentados neste capítulo foram publicados em (RIBEIRO; ANDRADE, 2015).

5.1 SEMÂNTICA DE CTL EM RELAÇÃO A UM KMTS

Consideramos as seguintes definições e notações, as quais serão úteis na definição e demonstração de propriedades relativas à semântica da lógica CTL apresentada ao final desta seção.

Definição 5.1.1. *Seja $M = (AP, S, R^+, R^-, L)$ um KMTS qualquer. O conjunto dos estados de M alcançáveis a partir de um estado $s \in S$ de M é o conjunto $\overrightarrow{S}(s) = \{s' \in S \mid s \to s' \in R^-\}$.*

Definição 5.1.2. *Seja $M = (AP, S, R^+, R^-, L)$ um KMTS qualquer, $s \to s' \in R^-$ uma transição may em M. O subconjunto de estruturas de Kripke de $K(M)$ que não tem a transição $s \to s'$ é o conjunto $\left[R/\right]_M(s, s') = \{k \in K(M) \mid s \to s' \notin k\}$.*

Definição 5.1.3. *Seja $M = (AP, S, R^+, R^-, L)$ um KMTS e φ uma fórmula CTL. O subconjunto de estruturas de Kripke representadas por M que satisfazem φ a partir de um estado $s \in S$ é dado pelo conjunto $\left[\varphi\right]_M^s = \{k \in K(M) \mid k, s \models \varphi\}$.*

Definição 5.1.4. *Dados um KMTS $M = (AP, S, R^+, R^-, L)$ e uma fórmula CTL φ, definimos os conjuntos união sucessiva($\left[\bigcup\right]^s_M(\varphi)$) e interseção sucessiva ($\left[\bigcap\right]^s_M(\varphi)$):*

$$\left[\bigcup\right]^s_M(\varphi) = \bigcup_{s' \in \overrightarrow{S}(s)} \left(\left[\varphi\right]^{s'}_M \setminus \left[R/\right]_M(s, s')\right);$$
$$\left[\bigcap\right]^s_M(\varphi) = \bigcap_{s' \in \overrightarrow{S}(s)} \left(\left[\varphi\right]^{s'}_M \cup \left[R/\right]_M(s, s')\right).$$

O conjunto união sucessiva captura todas as estruturas de Kripke representadas por M que satisfazem φ a partir de algum estado s' alcançável por uma transição $s \to s'$. Por outro lado, o conjunto interseção sucessiva captura todas as estruturas de Kripke que satisfazem φ a partir de todos os estados s' alcançáveis por todas as transições $s \to s'$ em M. De fato, o conjunto união sucessiva e interseção sucessiva capturam os modelos que satisfazem as fórmulas CTL $EX\varphi$ e $AX\varphi$, respectivamente.

Proposição 5.1.1. *Seja M um KMTS, s um estado de M e φ uma fórmula CTL. Uma estrutura de Kripke k pertence a $\left[\bigcup\right]^s_M(\varphi)$ sse $k \in K(M)$ e $k, s \models EX\varphi$.*

Prova. $k, s \models EX\varphi$ sse $\exists s \to s' \in k$ s.t $k, s' \models \varphi$ (I).

"⇒" Da definição 5.1.3, temos $\forall k \in \left[\varphi\right]^s_M$; $k, s \models \varphi$ e $k \in K(M)$. Logo, $\forall k \in \left(\left[\varphi\right]^{s'}_M \setminus \left[R/\right]_M(s, s')\right)$; $k, s' \models \varphi$ e $s \to s' \in k$ (II). Assim, de (II) e (I), temos $\forall k \in \left(\left[\varphi\right]^{s'}_M \setminus \left[R/\right]_M(s, s')\right)$; $k, s \models EX\varphi$. Dessa forma, $\forall k \in \bigcup_{s' \in \overrightarrow{S}(s)} \left(\left[\varphi\right]^{s'}_M \setminus \left[R/\right]_M(s, s')\right)$; $k, s \models EX\varphi$ e $k \in K(M)$. Portanto, $\forall k \in \left[\bigcup\right]^s_M(\varphi)$; $k, s \models EX\varphi$ e $k \in K(M)$.

"⇐" Suponhamos $k, s \models EX\varphi$ e $k \in K(M)$. De (I), $\exists s \to s' \in k$, $k, s' \models \varphi$. Assim, $k \in \left[\varphi\right]^{s'}_M$ e $k \notin \left[R/\right]_M(s, s')$. Logo, $k \in \left[\varphi\right]^{s'}_M \setminus \left[R/\right]_M(s, s')$. Portanto, $k \in \bigcup_{s' \in \overrightarrow{S}(s)} \left[\varphi\right]^{s'}_M \setminus \left[R/\right]_M(s, s')$ e $k \in \left[\bigcup\right]^s_M(\varphi)$.

∎

Proposição 5.1.2. *Seja M um KMTS, φ uma fórmula CTL e s um estado de M. Uma estrutura de Kripke k pertence a $\left[\bigcap\right]^s_M(\varphi)$ sse $k \in K(M)$ e $k, s \models AX\varphi$.*

Prova. A prova segue por contradição.

"⇒" Suponhamos que $\exists k \in \left[\bigcap\right]^s_M(\varphi)$ tal que $k \notin K(M)$ ou $K, s \not\models AX\varphi$.

Caso 1. $k \notin K(M)$. No entanto, pela definição 5.1.4, $\forall k \in \left[\bigcap\right]^s_M(\varphi)$; $k \in K(M)$ que contradiz a suposição.

Caso 2. $k \not\models AX\varphi$. Segue direto que $\exists s \to s' \in k$ s.t $k, s' \not\models \varphi$ (I). Por hipótese $k \in \left[\bigcap\right]^s_M(\varphi)$, logo

$$k \in \bigcap_{s' \in \overrightarrow{S}(s)} \left[\varphi\right]^s_M \cup \left[R/\right]_M(s, s').$$

Dessa forma, $\forall s' \in \overrightarrow{S}(s)$, $k \in \left[\varphi\right]^s_M \cup \left[R/\right]_M(s, s')$ que significa $\forall s' \in \overrightarrow{S}(s)$ $k \in \left[\varphi\right]^{s'}_M$ or $k \in \left[R/\right]_M(s, s')$ (II). Contudo, de (I) $\exists s \to s' \in k$ s.t $k, s' \not\models \varphi$; que que implica em $k \notin \left[\varphi\right]^{s'}_M$ e $k \notin \left[R/\right]_M(s, s')$. Que contradiz o enunciado em (II).

"$\Leftarrow$" Suponhamos que $\exists k \in K(M)$ s.t $k, s \models AX\varphi$ e $k \notin \left[\bigcap\right]^s_M(\varphi)$. Segue direto que $\forall s \to s' \in k$; $k, s' \models \varphi$ (I). Se $k \notin \left[\bigcap\right]^s_M(\varphi)$, então $\exists s' \in \overrightarrow{S}(s)$ tal que $k \notin \left(\left[\varphi\right]^{s'}_M \cup \left[R/\right]_M(s, s')\right)$. Logo, $k \notin \left[\varphi\right]^{s'}_M$ e $k \notin \left[R/\right]_M(s, s')(II)$. Portanto, $s \to s' \in k$ e $k, s' \not\models \varphi$. Derivamos uma contradição; visto que a partir de (I), $k, s' \models \varphi$.

∎

Definição 5.1.5. *A semântica de uma fórmula CTL φ em sua forma normal negativa com respeito a um KMTS é apresentada na tabela 5.1.*

Tabela 5.1: Semântica de uma fórmula CTL φ em relação a um KMTS M.

Fórmula	$\top$	F	$\bot$
$\|l\|_M(s)$	$l \in L(s)$	$\neg l \in L(s)$	caso contrário
$\|\varphi_1 \wedge \varphi_2\|_M(s)$	$\left[\varphi_1\right]^s_M \cap \left[\varphi_2\right]^s_M = K(M)$	$\left[\varphi_1\right]^s_M \cap \left[\varphi_2\right]^s_M = \emptyset$	caso contrário
$\|\varphi_1 \vee \varphi_2\|_M(s)$	$\left[\varphi_1\right]^s_M \cup \left[\varphi_2\right]^s_M = K(M)$	$\left[\varphi_1\right]^s_M \cup \left[\varphi_2\right]^s_M = \emptyset$	caso contrário
$\|EX\varphi\|_M(s)$	$\left[\bigcup\right]^s_M(\varphi) = K(M)$	$\left[\bigcup\right]^s_M(\varphi) = \emptyset$	caso contrário
$\|AX\varphi\|_M(s)$	$\left[\bigcap\right]^s_M(\varphi) = K(M)$	$\left[\bigcap\right]^s_M(\varphi) = \emptyset$	caso contrário
$\|E(\varphi_1 U \varphi_2)\|_M(s)$	$\left[\varphi_2\right]^s_M \cup \left(\left[\varphi_1\right]^s_M \cap \left[\bigcup\right]^s_M(E(\varphi_1 U \varphi_2))\right) = K(M)$	$\left[\varphi_2\right]^s_M \cup \left(\left[\varphi_1\right]^s_M \cap \left[\bigcup\right]^s_M(E(\varphi_1 U \varphi_2))\right) = \emptyset$	caso contrário
$\|A(\varphi_1 U \varphi_2)\|_M(s)$	$\left[\varphi_2\right]^s_M \cup \left(\left[\varphi_1\right]^s_M \cap \left[\bigcup\right]^s_M(A(\varphi_1 U \varphi_2))\right) = K(M)$	$\left[\varphi_2\right]^s_M \cup \left(\left[\varphi_1\right]^s_M \cap \left[\bigcup\right]^s_M(A(\varphi_1 U \varphi_2))\right) = \emptyset$	caso contrário
$\|E(\varphi_1 R \varphi_2)\|_M(s)$	$\left[\varphi_2\right]^s_M \cap \left(\left[\varphi_1\right]^s_M \cup \left[\bigcup\right]^s_M(E(\varphi_1 U \varphi_2))\right) = K(M)$	$\left[\varphi_2\right]^s_M \cap \left(\left[\varphi_1\right]^s_M \cup \left[\bigcup\right]^s_M(E(\varphi_1 U \varphi_2))\right) = \emptyset$	caso contrário
$\|A(\varphi_1 R \varphi_2)\|_M(s)$	$\left[\varphi_2\right]^s_M \cap \left(\left[\varphi_1\right]^s_M \cup \left[\bigcup\right]^s_M(A(\varphi_1 U \varphi_2))\right) = K(M)$	$\left[\varphi_2\right]^s_M \cap \left(\left[\varphi_1\right]^s_M \cup \left[\bigcup\right]^s_M(A(\varphi_1 U \varphi_2))\right) = \emptyset$	caso contrário

Proposição 5.1.3. *Seja M um KMTS e φ uma fórmula CTL, então*

$$\|\varphi\|_M(s) = \begin{cases} \top & \textit{sse } \forall k \in K(M); k, s \models \varphi = T; \\ F & \textit{sse } \forall k \in K(M); k, s \models \varphi = F; \\ \bot, & \textit{caso contrário} \end{cases}$$

Prova. Segue direto da semântica. ∎

5.2 OPERAÇÕES SOBRE KMTS

A semântica de CTL sobre um KMTS interpretado como um conjunto de estruturas de Kripke lida diretamente com operações de conjuntos. Desta forma, a fim de decidir se um conjunto de estruturas de Kripke representadas por um KMTS M satisfaz uma fórmula CTL, deveríamos, em princípio, verificar se cada modelo CTL em $K(M)$ satisfaz tal propriedade. Porém, podemos lidar diretamente com M em vez de considerar cada modelo expandido de M.

Nesse sentido, definimos nesta seção operações de conjuntos sobre KMTSs e provamos algumas propriedades destas operações.

As operações definidas a seguir consideram as definições e propriedades sobre instâncias e compatibilidade entre instâncias apresentadas na seção 4.1 do capítulo 4.

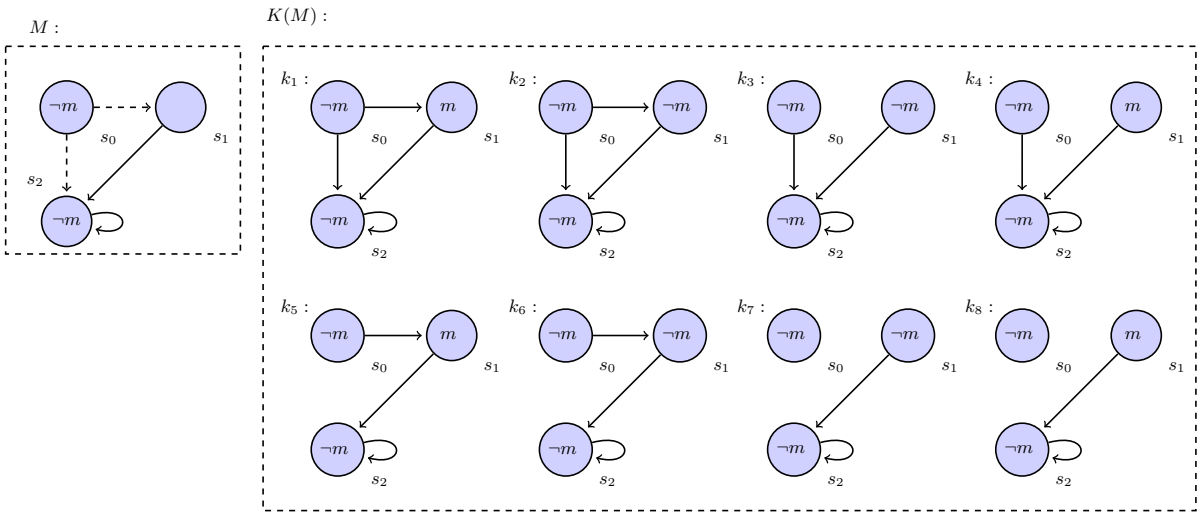

Figura 5.1: Exemplo de um KMTS e respectivo conjunto expansão

Definição 5.2.1. *Sejam M, M_1, M_2 KMTSs, X_1 e X_2 duas operações de mudanças, tais que $M \sqsubseteq M_1, M \sqsubseteq M_2$, $M_1 = M(X_1)$ e $M_2 = M(X_2)$. Definimos as operações de interseção, união e diferença em relação a dois KMTSs.*

$$
\begin{aligned}
&\textbf{\textit{União:}} && M_1 \sqcup M_2 = \{M_1, M_2\} \\
&\textbf{\textit{Interseção:}} && M_1 \sqcap M_2 = \begin{cases} \emptyset & \textit{sse } X_1 \not\simeq X_2 \\ \{M(X_1 \cup X_2)\} & \textit{caso contrário} \end{cases} \\
&\textbf{\textit{Diferença:}} && M_1 \setminus M_2 = \begin{cases} \{M_1\} & \textit{sse } X_1 \not\simeq X_2 \\ \bigcup\limits_{\varrho_i \in (X_2 \setminus X_1)} \{M\big(X_1 \cup \{\neg \varrho_i\}\big)\} & \textit{caso contrário} \end{cases}
\end{aligned}
$$

A operação de diferença $M_1 \setminus M_2$ gera um conjunto de KMTSs de tal forma que os modelos CTL representados por cada um deles estão presentes em $K(M_1)$, mas não ocorrem em $K(M_2)$. Como M_1 e M_2 são instâncias de M, os modelos resultantes da

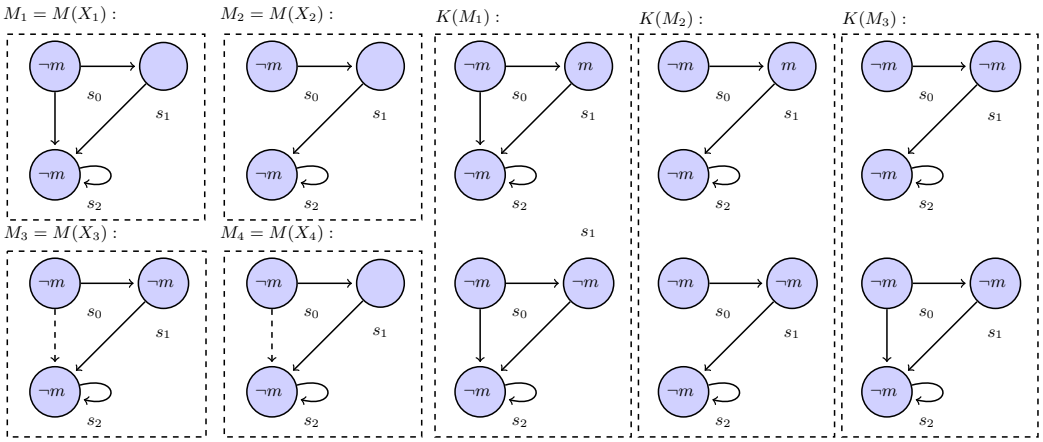

Figura 5.2: Instâncias M_1, M_2, M_3 and M_4 do KMTS M ilustrado na figura 5.1 e conjuntos expansão $K(M_1), K(M_2)$ e $K(M_3)$.

diferença entre M_1 e M_2 podem ser definidos a partir das mudanças que geram M_1 e M_2. Consequentemente, se X_1 e X_2 não são compatíveis, então a interseção entre os modelos é vazia e a diferença é o próprio M_1. Por outro lado, se eles são compatíveis, então os modelos resultantes da operação de diferença são obtidos a partir de M pela mudança X_1 (X_1 gera M_1 quando aplicada a M) junto às operações complementares e às operações primitivas em X_2 que não ocorrem em X_1 ($X_1 \cup \{\neg p_i\}$) a fim de eliminar a interseção entre $K(M_1)$ e $K(M_2)$. Por exemplo, para os KMTSs M_2 e M_3 ilustrados na figura 5.2, $M_2 \setminus M_3$ gera um conjunto contendo apenas a estrutura de Kripke da parte superior de $K(M_2)$. Este modelo é gerado pela mudança $X' = \{P_1(s_0, s_2),\ P_2(s_0, s_1)\} \cup \{P_3(s_1, m)\}$, visto que $M_2 = M(\{P_1(s_0, s_2),\ P_2(s_0, s_1)\})$ e $M_3 = M(\{P_2(s_0, s_1), P_3(s_1, \neg m)\})$, e $P_3(s_1, \neg m)$ não pertence a X_1.

A operação de interseção $M_1 \sqcap M_2$ gera um conjunto com um único KMTS cuja expansão é equivalente a $K(M_1) \cap K(M_2)$. A operação de união é simples e não carece de nenhuma explicação adicional.

Às vezes os conjuntos expansão de dois KMTSs podem ser representados por um único KMTS. Por exemplo, o conjunto das estruturas de Kripke representadas pelo KMTS M_4 na figura 5.2 é equivalente à união dos conjuntos expansão de M_1 e M_2, i.e., $K(M_4) = K(M_1) \cup K(M_2)$. Como consequência, $\{M_1, M_2\}$ pode ser expresso por um único KMTS, neste caso M_4. Neste sentido, definimos a operação de contração que é um caso específico da operação de união para tais casos. Se dois KMTSs não podem ser contraídos em um único, então a operação de contração será equivalente à operação de união previamente definida.

Definição 5.2.2. *Seja M um KMTS, M_1 e M_2 instâncias de M geradas, respectivamente, pelos conjuntos de mudanças X_1 e X_2. A operação de contração, denotada por $M_1 \sqcup^+ M_2$,*

é definida como:

$$M_1 \sqcup^+ M_2 = \begin{cases} \{M(X_1 \cap X_2)\} & \textit{sse } X_1 \subseteq X_2 \textit{ or } X_2 \subseteq X_1 \textit{ ou} \\ & \exists \varrho \in X_1 \textit{ t.q } \neg\varrho \in X_2 \textit{ e} \\ & X_1 \setminus \{\varrho\} = X_2 \setminus \{\neg\varrho\} \\ \{M_1,\ M_2\} & \textit{caso contrário} \end{cases}$$

A fim de contrair dois modelos em um único, é necessário (mas não é suficiente) que ambos tenham exatamente uma operação complementar em relação ao outro, caso contrário a operação de contração resulta em um conjunto que contém exatamente estes dois modelos. Explicamos a contração através de um exemplo. Suponhamos dois KMTSs $M_1 = M(X_1)$ e $M_2 = M(X_2)$ tal que $X_1 \subseteq X_2$. Logo, $X_1 \cap X_2 = X_1$ e $K(M_2) \subseteq K(M_1)$, consequentemente o resultado da contração é M_1, i.e., $M_1 \sqcup^+ M_2 = M(X_1 \cap X_2) = M_1$. Consideremos agora que X_1 tenha dentre todas as suas operações apenas uma operação p complementar com alguma operação em X_2 e que X_1 seja igual a X_2 a menos desta operação. As operações ϱ e $\neg\varrho$ podem ser aplicadas sobre M e geram respectivamente as instâncias $M_1 = M(X_1)$ e $M_2 = M(X_2)$. Este caso se reduz ao anterior, ou seja, a contração resulta em $M(X_1 \cap X_2)$, uma vez que $K(M(X_1 \cap X_2)) = K(M_1) \cup K(M_2)$.

Na figura 5.2, o modelo M_1 pode ser contraído com M_2, i.e., $M_1 \sqcup^+ M_2 = M_4$, pois $X_1 \setminus \{P_2(s_0, s_2)\} = X_2 \setminus \{P_1(s_0, s_2)\}$ e $P_1(s_0, s_2)$ é complementar com $P_2(s_0, s_2)$. Além disso, o modelo M_1 pode ser contraído com M_4, e o modelo M_2 pode ser contraído com M_4, ambas as contrações resultam em M_4. Todavia, M_3 não pode ser contraído com nenhum outro modelo da figura 5.2.

5.2.1 Lidando com Conjuntos de KMTSs

Para lidar com um conjunto de modelos CTL, podemos considerar um KMTS cuja expansão represente tais modelos. Porém, às vezes um único KMTS pode não ser suficiente para representar um conjunto específico de estruturas de Kripke. Uma alternativa é considerar um conjunto de KMTSs que cubra o conjunto de modelos CTL requerido.

Visto que o conjunto expansão de um KMTS é exponencial em relação ao número indeterminações, lidar com um conjunto de KMTSs é computacionalmente mais conveniente, pois é preferível lidar diretamente com um KMTS em vez de seu conjunto expansão. Definimos, neta seção, operações sobre conjuntos partições de KMTSs que detém propriedades úteis para a definição da verificação de modelos por contração.

Seja M um KMTS, sempre é possível construir um conjunto Γ de instâncias de tal forma que cada elemento deste conjunto é uma instância de M e a interseção entre dois modelos quaisquer em Γ é sempre vazia.

Definição 5.2.3. *Um conjunto Γ de instâncias é um Conjunto Partição (CP) de um KMTS M sse cada modelo em Γ é uma instância de M e $\forall M_1, M_2 \in \Gamma, M_1 \sqcap M_2 = \emptyset$.*

Definição 5.2.4. *Sejam Γ e Γ' dois conjuntos de instâncias de um KMTS M. Γ e Γ' são equivalentes, $\Gamma \equiv \Gamma'$, sse $\bigcup_{M_i \in \Gamma} K(M_i) = \bigcup_{M_i \in \Gamma'} K(M_i)$.*

Definição 5.2.5. *Sejam Γ_1 e Γ_2 dois conjuntos de instâncias de um KMTS M. Definimos a operação de diferença ($\setminus\!\!\setminus$):*

$$\Gamma_1 \setminus\!\!\setminus \Gamma_2 = \bigcup_{\substack{M_i \in \Gamma_1, \\ M_k \in \Gamma_2}} M_i \setminus M_k$$

A operação de diferença apresentada na definição 5.2.5 calcula a diferença dos conjuntos expansão dos KMTS de um conjunto Γ de KMTSs. A operação de diferença entre dois conjuntos de KMTSs Γ_1 e Γ_2 resulta em um conjunto Γ' tal que $K(\Gamma') = K(\Gamma_1) \setminus K(\Gamma_2)$.

Teorema 5.2.1. *Para qualquer conjunto Γ de instâncias de um KMTS M há sempre um CP Γ' tal que $\Gamma \equiv \Gamma'$.*

Prova. Seja M um KMTS, Γ um conjunto de instâncias de M e M_1 uma instância de M em Γ. Criemos o conjunto $\Gamma_1 = \Gamma \setminus\!\!\setminus \{M_1\}$. Logo, $\forall M_i \in \Gamma_1, M_i \sqcap M_1 = \emptyset$ pela operação de diferença, e $\Gamma_1 \cup \{M_1\} \equiv \Gamma$. Escolhamos um modelo $M_2 \in \Gamma_1$, logo o conjunto $\Gamma_1' = \{M_1, M_2\}$ é um CP. Criemos agora o conjunto $\Gamma_2 = \Gamma_1 \setminus\!\!\setminus \{M_2\}$, assim $\forall M_i \in \Gamma_2,\ M_i \sqcap M_2 = \emptyset$ e $M_i \sqcap M_1 = \emptyset$ e $\Gamma_2 \cup \{M_1, M_2\} \equiv \Gamma$. Escolhamos um modelo $M_3 \in \Gamma_2$, logo $\{M_1, M_2, M_3\}$ é um CP e $\Gamma_3 \cup \{M_1, M_2, M_3\} \equiv \Gamma$. Visto que Γ é um conjunto finito, se continuarmos a seguir esta construção, em algum momento não será mais possível tomar nenhum elemento, e o conjunto obtido neste último passo será um CP equivalente a Γ. ∎

Definição 5.2.6. *Sejam Γ_1 e Γ_2 dois conjuntos de instâncias de um KMTS M. Definimos as operações de interseção ($\sqcap\!\!\cdot$) e união ($\sqcup\!\!\cdot$).*

$$\Gamma_1 \sqcap\!\!\cdot\, \Gamma_2 = \bigcup_{\substack{M_i \in \Gamma_1, \\ M_k \in \Gamma_2}} M_i \sqcap M_k \qquad \Gamma_1 \sqcup\!\!\cdot\, \Gamma_2 = \Gamma_1 \cup CP(\Gamma_2 \setminus\!\!\setminus (\Gamma_1 \sqcap\!\!\cdot\, \Gamma_2))$$

onde $CP(\Gamma)$ é um Conjunto Partição equivalente a um conjunto Γ de instâncias.

Seja $K(\Gamma)$ o conjunto de todas as estruturas de Kripke representadas por todos os KMTSs em Γ. As operações de interseção e união apresentadas na definição 5.2.6 calculam respectivamente a interseção e união dos conjuntos expansão dos KMTS em Γ. A operação de interseção entre dois conjuntos de KMTSs Γ_1 e Γ_2 resulta em um conjunto Γ' tal que $K(\Gamma') = K(\Gamma_1) \cap K(\Gamma_2)$. A união é interpretada de forma similar.

Proposição 5.2.1. *Se Γ_1 e Γ_2 são dois CPs de um KMTS M, então $\Gamma_1 \sqcap\!\!\cdot\, \Gamma_2$ e $\Gamma_1 \sqcup\!\!\cdot\, \Gamma_2$ são CPs.*

Prova. Segue direto da definição 5.2.6. ∎

Se um CP de um KMTS M representa todos os modelos CTL de $K(M)$, então dizemos que tal CP é um Conjunto Partição Completo.

Definição 5.2.7. *Seja M um KMTS e Γ um conjunto de instâncias de M. Dizemos que Γ é um Conjunto Partição Completo (CPC) de M sse Γ é um CP e $K(M) = \bigcup_{M_i \in \Gamma} K(M_i)$.*

Corolário 5.2.1. *Se Γ é um conjunto de instâncias de um KMTS M e $K(M) = \bigcup_{M_i \in \Gamma} K(M_i)$, então há um CPC Γ' tal que $\Gamma \equiv \Gamma'$.*

Prova. Segue direto do teorema 5.2.1. Suponhamos um conjunto Γ de instâncias de M como enunciado acima. Pelo teorema 5.2.1, existe um CP Γ' equivalente à Γ e visto que $K(M) = \bigcup_{M_i \in \Gamma} K(M_i)$, Γ' é um CPC. ∎

Qualquer KMTS M pode ser obtido por um número finito de operações de contrações aplicadas sobre um CPC de M. Para provarmos esta afirmação, definimos primeiramente o conceito de Árvore Conjunto Partição e mostramos como representar um CP sobre estas estruturas.

5.2.2 Árvore Conjunto Partição

Uma Árvore Conjunto Partição (ACP) é um árvore binária que representa um conjunto partição Γ de um KMTS M. Cada nó v de uma ACP é rotulado por uma operação primitiva de mudança $L_T(v)$ aplicável sobre M.

Definição 5.2.8. *Uma Árvore Conjunto Partição (ACP) de um KMTS $M = (AP, S, R^+, R^-, L)$ é uma tupla $T_M = (N, v_0, E, L_T, Lf, Rg)$, onde N é um conjunto finito de nós, $v_0 \in N$ é a raiz, $E \subseteq N \times N$ é o conjunto de arestas, L_T é uma função parcial de rotulação que mapeia cada nó em N a uma operação primitiva aplicável sobre M tal que v_0 é o único nó de uma ACP que não está definido em L_T; e Lf, Rg são funções parciais que mapeiam respectivamente um nó ao seu filho da esquerda e da direita. Além disso, para cada nó não terminal $v \in N$, existem nós $v_1, v_2 \in N$ tal que $Lf(v) = v_1$ e $Rg(v) = v_2$ sse existe um literal $p \in Lit$ tal que $L_T(v_1) = p$ e $L_T(v_2) = \neg p$; onde Lit é o conjunto dos literais de AP.*

Seja $\pi = v_0 \to v_1 \to \cdots \to v_n$ um caminho entre dois nós v_0 e v_n em uma ACP, denotamos por $\pi \setminus v_0$ o subcaminho $v_1 \to \cdots \to v_n$ de π, e $v \in \pi$ denota que um nó v pertence a um caminho π. Ressaltamos que cada operação rotulada aos vértices ao longo de um caminho de v_0 a v_n segue de uma indeterminação do respectivo KMTS.

Definição 5.2.9. *Seja M um KMTS, $T_M = (N, v_0, E, L_T, Lf, Rg)$ uma ACP. Definimos a operação $Change(v_n)$ que mapeia o único caminho $\pi = v_0 \to v_1 \to \cdots \to v_n$ entre v_0 e qualquer nó $v_n \in T_M$ a um conjunto de mudanças aplicáveis a M:*

$$Change(v_n) = \bigcup_{v_k \in \pi \setminus v_0} \{L_T(v_k)\}$$

Uma mudança em um nó v como definida acima considera todas as operações primitivas que ocorrem ao longo do único caminho que conecta o nó raiz v_0 da ACP ao vértice v excluindo a raiz que não define nenhuma operação de mudança.

A mudança $Change(v)$ de um nó em uma ACP T_M quando aplicada a um KMTS M gera uma instância deste KMTS e dizemos que um nó representa uma instância de M. O conjunto das instâncias representadas por todos os nós terminais de uma ACP é um CP.

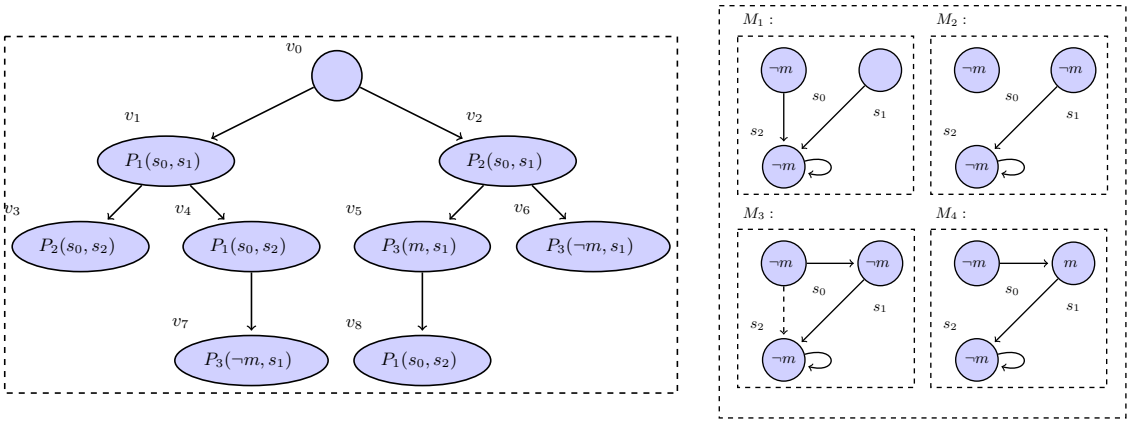

Figura 5.3: Uma Árvore Conjunto Partição que representa o CP $\{M_1, M_2, M_3, M_4\}$.

A figura 5.3 ilustra um CP Γ do KMTS M apresentado na figura 5.1 e uma ACP T_M que representa Γ. M_1, M_2, M_3 e M_4 são gerados a partir das mudanças definidas pelos nós folhas da ACP, i.e, pela aplicação das mudanças $Change(v_3), Change(v_7)$, $Change(v_6)$ e $Change(v_8)$ sobre M, respectivamente.

Lema 5.2.1. *Seja Γ um CP, em relação a um KMTS M, representado por uma ACP T_M e M' uma instância de M. Se $\Gamma \cup \{M'\}$ é um CP, então há um CP Γ' equivalente a Γ que também pode ser representado por uma ACP.*

Prova. Da operação de contração, temos que para qualquer mudança X e operação primitiva ϱ, $M(X \cup \{\varrho\}) \sqcup^{+} M(X \cup \{\neg\varrho\}) = M(X)$ (I). Tomemos uma ACP T'_M igual T_M. Se M' é uma instância de M, então M' é gerado por uma mudança X', i.e., $M' = M(X')$. Selecionemos em T'_M o seguinte subcaminho $\pi = v_0 \rightarrow \cdots \rightarrow v_i$ de tal forma que $Change(v_i) \subset X'$ e v_i tenha somente um filho v_{i+1} tal que $L_T(v_{i+1}) \notin X'$ ou v_i tem dois filhos; onde$L_T(Lf(v_i)) \notin X'$ e $L_T(Rg(v_i)) \notin X'$.
Caso 1. v_i tem apenas um nó filho v_{i+1} e $L_T(v_{i+1}) \notin X'$. Logo, temos dois casos: $\neg L_T(v_{i+1}) \in X'$ ou $\neg L_T(v_{i+1}) \notin X'$.

(a) $\neg L_T(v_{i+1}) \in X'$. Criemos um nó v_{i+2} para ser o outro filho de v_i e adicionemos a T'_M o seguinte caminho $\pi_1 = v_0 \rightarrow \cdots \rightarrow v_i \rightarrow v_{i+2} \rightarrow \cdots \rightarrow v'$ em que $Change(v') = X'$ e $L_T(v_{i+2}) = \neg L_T(v_{i+1})$. Assim, a ACP T'_M resultante representa o CP $\Gamma' = \Gamma \cup \{M'\}$.

(b) $\neg L_T(v_{i+1}) \notin X'$. Seja $X_i = Change(v_i)$, logo $X_i \subset X'$ e $X' = X_i \cup (X' \setminus X_i)$. Tomemos $M'_1 = M(X' \cup \{\varrho\})$ e $M'_2 = M(X' \cup \{\neg\varrho\})$, onde $\varrho = L_T(v_{i+1})$. De (I), temos que $M' = M(X') = M(X' \cup \{\varrho\}) \sqcup^{+} M(X' \cup \{\neg\varrho\})$ que implica em $M' = M'_1 \sqcup^{+} M'_2$. Desta forma, o conjunto $\Gamma' = \Gamma \cup \{M'_1, M'_2\}$ é equivalente a $\Gamma \cup \{M'\}$ e por hipótese é um CP. Logo, se representarmos M'_1 e M'_2 em T'_M, alcançamos uma representação de M' em T'_M. Para fazê-lo, tomemos um v_{i+2} para ser filho de v_i e adicionemos a T'_M o caminho $\pi_1 = v_0 \rightarrow \ldots v_i \rightarrow v_{i+2} \rightarrow \cdots \rightarrow v'$ em que $L_T(v_{i+2}) = \neg L_T(v_{i+1})$ e $Change(v') = X' \cup \{L_T(v_{i+2})\}$. Portanto, T'_M representa M'_2. Seja $X'_1 = X' \cup \{L_T(v_{i+1})\}$, para representarmos M'_1, devemos criar um caminho $\pi_2 = v_0 \rightarrow \cdots \rightarrow v_i \rightarrow v_{i+1} \rightarrow$

$\cdots \to v'$ tal que $Change(v') = X'_1$. Para tanto, selecionemos em T'_M um subcaminho $\pi'_2 = v_0 \to \cdots \to v_i \to v_{i+1} \to \cdots \to v_k$ de tal forma que $Change(v_k) \subset X'_1$ e apenas uma das seguintes condições seja verdade: ou v_k tem apenas um nó filho v_{k+1} tal que $L_T(v_{k+1}) \notin X'_1$; ou v_k tem dois nós filhos em que $L_T(Lf(v_k)) \notin X'_1$ e $L_T(Rg(v_k)) \notin X'_1$. Ambas as condições se reduzem aos casos (1) e (2).

Caso 2. v_i tem dois nós filhos, onde $L_T(Lf(v_i)) \notin X'$ e $L_T(Rg(v_i)) \notin X'$. Seja $X_i = Change(v_i)$, assim $X_i \subset X'$ e $X' = X_i \cup (X' \setminus X_i)$. Tomemos $M'_1 = M(X' \cup \{\varrho\})$ e $M'_2 = M(X' \cup \{\neg\varrho\})$, onde $\varrho = L_T(Lf(v_1))$. De (I), temos que $M' = M(X') = M(X' \cup \{\varrho\}) \sqcup^+ M(X' \cup \{\neg\varrho\})$ que implica em $M' = M'_1 \sqcup^+ M'_2$. Assim, o conjunto $\Gamma' = \Gamma \cup \{M'_1, M'_2\}$ é equivalente a $\Gamma \cup \{M'\}$ e por hipótese é um CP. Logo, devemos representar em T'_M as instâncias M'_1 e M'_2. Seja $X'_1 = X' \cup \{L_T(Lf(v_i))\}$ e $X'_2 = X' \cup \{L_T(Rg(v_i))\}$; para representarmos M'_1, devemos criar um caminho $\pi_1 = v_0 \to \cdots \to v_i \to Lf(v_i) \to \cdots \to v_j$ onde $Change(v_j) = X'_1$. Tomemos em T'_M um subcaminho $\pi'_1 = v_0 \to \cdots \to v_i \to Lf(v_i) \to \cdots \to v_k$ de tal forma que $Change(v_k) \subset X'_1$ e apenas uma das seguintes condições seja verdade: or v_k tem apenas um nó filho v_{k+1} em que $L_T(v_{k+1}) \notin X'_1$; ou v_k tem dois nós filhos, onde $L_T(Lf(v_k)) \notin X'_1$ e $L_T(Rg(v_k)) \notin X'_1$. Ambas condições se reduzem aos casos (1) e (2). Para representarmos M_2, procedemos de forma similar ao que fizemos com M_1. Visto que um caminho em uma ACP é finito, certamente em algum momento no futuro os casos $1-(b)$ e 2 alcançarão o caso $1-(a)$, desta forma a ACP T'_M representará o CP Γ'. ∎

Segue do lemma 5.2.1 que para todo CP Γ há sempre um CP Γ' equivalente a Γ que é representado por uma ACP.

Teorema 5.2.2. *Se Γ é um CP com relação a um KMTS M, então há sempre um CP Γ' equivalente a Γ que pode ser representado por uma ACP T_M.*

Prova. Seja $\Gamma = \{M_1, \ldots, M_n\}$, cada $\{M_i\}$ é também um CP por definição. Do lemma 5.2.1, há um CP $\Gamma_2 \equiv \{M_1\} \cup \{M_2\}$ que pode ser representado por uma ACP. Além disso, existe um CP Γ_3 equivalente a $\Gamma_2 \cup \{M_3\}$ que pode ser representado por uma ACP. Sucessivamente, existe um CP $\Gamma_n \equiv \Gamma_{n-1} \cup \{M_n\}$ por uma ACP que é equivalente a $\{M_1, \ldots, M_n\} = \Gamma$. ∎

Teorema 5.2.3. *Seja M um e $T_M = (N, v_0, E, L_T, Lf, Rg)$ uma ACP que representa um CP Γ definido a partir de M. Se Γ é um CPC, então para cada nó não terminal $v_k \in N$ e $X_k = Change(v_k)$, $M_k = M(X_k)$ pode ser obtido por um número finito de operações de contração aplicadas em Γ.*

Prova. A prova segue por indução na estrutura de T_M. Seja $\pi_k = v_0 \to \cdots \to v_k$ um caminho de T_M, se M tem m indefinições, então $1 \leq k \leq m$.

Caso base: seja $\pi_k = v_0 \to \cdots \to v_{k-1} \to v_k$ um dos caminhos mais longos de T_M. Então, v_k é um nó não terminal. Como π_k é um dos caminhos mais longos de Γ que é um CPC, o caminho $\pi'_k = v_0 \to \cdots \to v_{k-1} \to v'_k$ pertence a T_M, onde $L_T(v_k) = \neg L_T(v'_k)$. Seja $\varrho = L_T(v_k)$, $X_{k-1} = Change(v_{k-1})$. $Change(v_k) = X_{k-1} \cup \{\varrho\}$ e $Change(v'_k) =$

$X_{k-1} \cup \{\neg\varrho\}$. Logo, temos que $M(X_{k-1}) = M(X_{k-1} \cup \{\varrho\}) \sqcup^{+} M(X_{k-1} \cup \{\neg\varrho\})$. Portanto, $M(X_{k-1})$ é obtido por uma operação de contração em Γ.

Passo da Indução: suponhamos que qualquer $M(X_k)$ possa ser gerado por um número finito de contrações. Mostramos que $\forall k-1$, $M(X_{k-1})$ pode ser gerado por um número finito de operações de contração. O caminho $\pi_{k-1} = v_0 \rightarrow \cdots \rightarrow v_{k-1}$ tem dois nós filhos v_k e v'_k, onde $L_T(v_K) = \neg L_T(v'_k)$ em virtude de Γ ser um CPC. Seja $\varrho = L_T(v_k)$, $X_{k-1} = Change(v_{k-1})$, $X_k = Change(v_k)$ e $X'_k = Change(v'_k)$. Assim, $X_k = X_{k-1} \cup \{\varrho\}$ e $X'_k = X_{k-1} \cup \{\neg\varrho\}$. Logo, $M(X_{k-1}) = M(X_k) \sqcup^{+} M(X'_k)$. Por hipótese da indução, $M(X_k)$ e $M(X'_k)$ podem ser obtidos por um número finito de operações de contração, portanto $M(X_{k-1})$ também é obtido por um número finito de operações de contração. ∎

O corolário 5.2.2 apresentado a seguir garante que dado um CPC de um KMTS M, sempre é possível obter o próprio modelo M a partir de aplicações de operações de contração sobre tal CPC. O algoritmo de verificação de modelos que propomos considera a aplicação de operações de contração sobre CPs. Quando um CPC é alcançado, a aplicação das operações de contração sobre o mesmo geram um conjunto que detém apenas o modelo M significando que todas as estruturas de Kripke representadas por M satisfazem a propriedade requerida. Portando, o conjunto unitário $\{M\}$ representa o valor verdade $\top$. Os demais valores verdades são representados por CPs que não são CPC. Nestes casos, um CP não vazio representa o valor verdade $\bot$, enquanto que um CP vazio representa o valor verdade F.

Corolário 5.2.2. *Seja M um KMTS e Γ um CP de M representado por uma ACP T_M. Se Γ é um CPC, então M pode ser gerado pela aplicação de um número finito de operações de contração sobre Γ.*

Prova. Segue direto do teorema 5.2.3.

Seja $\pi = v_0 \rightarrow v_1$ um caminho de T_M. Em virtude de Γ ser um CPC, $\pi' = v_0 \rightarrow v'_1$ pertence a T_M. Seja $\varrho = L_T(v_1)$, então $X_1 = Change(v_1) = \{\varrho\})$ e $X'_1 = Change(v'_1) = \{\neg\varrho\}$. Segue do teorema 5.2.3 que $M(X_1)$ e $M(X'_1)$ são gerados pela aplicação de operações de contração sobre Γ. Portanto, $M(X_1) \sqcup^{+} M(X'_1) = M$. ∎

Seja Γ um CP e Γ' um CP obtido pela aplicação de operações de contração sobre Γ, dizemos que Γ' é Conjunto Partição Maximal (CPM) se não for mais possível aplicar operações de contração sobre Γ'. De acordo com o corolário 5.2.2, se um CP é de fato um CPC, então o respectivo CPM é exatamente o conjunto $\{M\}$. Escrevemos $\sqcup^{+}(\Gamma)$ para denotar o CPM resultante de um CP Γ.

Proposição 5.2.2. *Sejam A e B dois CPs de um KMTS M. A absorção é válida:*

1. $\sqcup^{+}(A \sqcup B) \equiv \sqcup^{+}\big(A \sqcup \sqcup^{+}(B) \big)$
2. $\sqcup^{+}(A \sqcap B) \equiv \sqcup^{+}\big(A \sqcap \sqcup^{+}(B) \big)$

Prova. A contração conserva o conjunto expansão dos modelos envolvidos na contração, ou seja, $K(M_1 \sqcup^+ M_2) = K(M_1) \cup K(M_2)$; onde M_1 e M_2 são instâncias quaisquer de M. Logo, $\sqcup^+(\Gamma) \equiv \Gamma$, onde Γ é um CP qualquer de um KMTS M. Suponhamos $\Gamma = B$, assim

$$\bigcup_{M_j \in B} K(M_j) = \bigcup_{M_j \in \sqcup^+(B)} K(M_j) \tag{5.1}$$

1. $\sqcup^+(A \sqcup B) \equiv \sqcup^+\big(A \sqcup \sqcup^+(B) \big)$

Da operação de união de CPs temos que $A \sqcup B \equiv A \cup B$, ou seja,

$$\bigcup_{M_j \in A \sqcup B} K(M_j) = \left(\bigcup_{M_j \in A} K(M_j) \right) \cup \left(\bigcup_{M_j \in B} K(M_j) \right)$$

De (5.1), temos que

$$\bigcup_{M_j \in A \sqcup B} K(M_j) = \left(\bigcup_{M_j \in A} K(M_j) \right) \cup \left(\bigcup_{M_j \in \sqcup^+(B)} K(M_j) \right)$$

Segue da equação acima e da definição 5.2.4 que $A \sqcup B \equiv A \sqcup \sqcup^+(B)$, e concluímos que $\sqcup^+(A \sqcup B) \equiv \sqcup^+(A \sqcup \sqcup^+(B))$.

2. $\sqcup^+(A \sqcap B) \equiv \sqcup^+\big(A \sqcap \sqcup^+(B) \big)$. A prova é similar a do item anterior.

∎

5.3 VERIFICAÇÃO DE MODELOS POR CONTRAÇÃO

No capítulo 3, apresentamos a abordagem de verificação de modelos com jogos proposta por (SHOHAM; GRUMBERG, 2007) e (GRUMBERG et al., 2007). A verificação de modelos com jogos combinada com a operação de contração aplicada a cada configuração da arena de um jogo é suficiente para determinar o resultado da verificação de modelos para KMTSs interpretados como um conjunto de modelos CTL. Chamamos esta verificação de modelos com jogos alinhada com a operação de contração de verificação de modelos por contração.

Seja M um KMTS, s_0 um estado de M e φ uma fórmula CTL. A verificação de modelos $M, s_0 \models \varphi$ é realizado sobre uma arena construída de acordo com as regras do jogo que definem os movimentos que cada jogador pode realizar nas configurações em que joga. A arena é um grafo de configurações construído através da decomposição de φ em suas subfórmulas seguindo as regras do jogo apresentadas na figura 5.4. Cada configuração da arena pertence a $S \times sub(\varphi)$, onde S é o conjunto de estados de M e

$(1)\frac{s \vdash \psi_0 \vee \psi_1}{s \vdash \psi_i} : i \in \{0,1\}$ $(\exists ve)$	$(2)\frac{s \vdash \psi_0 \wedge \psi_1}{s \vdash \psi_i} : i \in \{0,1\}$ $(\forall belard)$
$(3)\frac{s \vdash EX\,\varphi}{t \vdash \varphi} : (s,t) \in R^-$ $(\exists ve)$	$(4)\frac{s \vdash AX\,\varphi}{t \vdash \varphi} : (s,t) \in R^-$ $(\forall belard)$
$(5)\frac{A(\varphi_1 U \varphi_2)}{s \vdash \varphi_2 \vee (\varphi_1 \wedge AXA(\varphi_1 U \varphi_2))}$ $(\exists ve)$	$(6)\frac{E(\varphi_1 U \varphi_2)}{s \vdash \varphi_2 \vee (\varphi_1 \wedge EXE(\varphi_1 U \varphi_2))}$ $(\exists ve)$
$(7)\frac{A(\varphi_1 R \varphi_2)}{s \vdash \varphi_2 \wedge (\varphi_1 \vee AXA(\varphi_1 R \varphi_2))}$ $(\exists ve)$	$(8)\frac{E(\varphi_1 R \varphi_2)}{s \vdash \varphi_2 \wedge (\varphi_1 \vee EXE(\varphi_1 R \varphi_2))}$ $(\exists ve)$

Figura 5.4: Regras do jogo para verificação de modelos com Jogos

$sub(\varphi)$ é o conjunto de subfórmulas de φ. Denotamos uma configuração por $s \vdash \psi$, onde ψ é uma subfórmula de φ e s é um estado de M.

Apresentamos na figura 5.5 a arena da verificação de modelos com jogos para $M, s_0 \models E(mU\neg m)$. A configuração inicial C_0 é rotulada com $(EXm) \vee E(mU\neg m)$ e s_0, i.e, $C_0 = s_0 \vdash (EXm) \vee E(mU\neg m)$. Conforme a regra (1), C_0 tem duas configurações filhas $C_1 = s_0 \vdash EXm$ e $C_3 = s_0 \vdash E(mU\neg m)$. De acordo com a regra (6), C_3 tem uma configuração filha $C_4 = s_0 \vdash \neg m \vee (m \wedge EXE(mU\neg m))$ que, de acordo com a regra (1), é decomposta em duas configurações: $C_5 = s_0 \vdash \neg m$ e $C_6 = s_0 \vdash m \wedge EXE(mU\neg m)$. A arena é construída seguindo as demais regras do jogo até que mais nenhuma regra possa ser aplicada. Os operadores EX e AX estão relacionados a caminhos do KMTS sobre o qual a verificação de modelos é realizada, e as configurações rotuladas com estes operadores refletem transições do respectivo KMTS. Por exemplo, a configuração C_8 está rotulada com $s_0 \vdash EXE(mU\neg m)$ e de acordo com a regra (4) C_8 tem uma configuração filha $C_9 = s_1 \vdash E(mU\neg m)$ que representa a transição (*must*) $s_0 \to s_1$ de M.

Seja G a arena do jogo de um KMTS M e fórmula CTL φ. A verificação de modelos por contração é uma função de coloração $\chi : V \to \{\top, F, \bot\}$, onde V é o conjunto de vértices de G, que mapeia cada configuração em G a um valor verdade. A função de coloração é definida sobre uma função δ de contração maximal que mapeia cada configuração da arena a um CPM Γ. A expansão dos KMTSs neste CP equivale ao conjunto de estruturas de Kripke em $K(M)$ que satisfazem a fórmula CTL a partir do estado s rotulado na respectiva configuração. Visto que o CP Γ resultante é um CPM, $\Gamma = \{M\}$ sse M satisfaz a fórmula φ a partir do estado s rotulados na respectiva configuração, $\Gamma = \emptyset$ se nenhum modelo CTL em $K(M)$ satisfaz φ a partir desta configuração, e Γ é um CP diferente de $\{M\}$ e $\emptyset$, caso contrário.

Definição 5.3.1. *Seja M um KMTS, s e s' estados de M, φ uma fórmula CTL e G a arena do jogo $M, s_0 \models \varphi$. A função δ de contração maximal é definida recursivamente como:*

$$\delta(s \vdash l) = \begin{cases} \{M\} & \text{sse } l \in L(s); \\ \emptyset & \text{sse } \neg l \in L(s); \\ \{M(\{P_3(s,l)\})\} & \text{caso contrário} \end{cases}$$

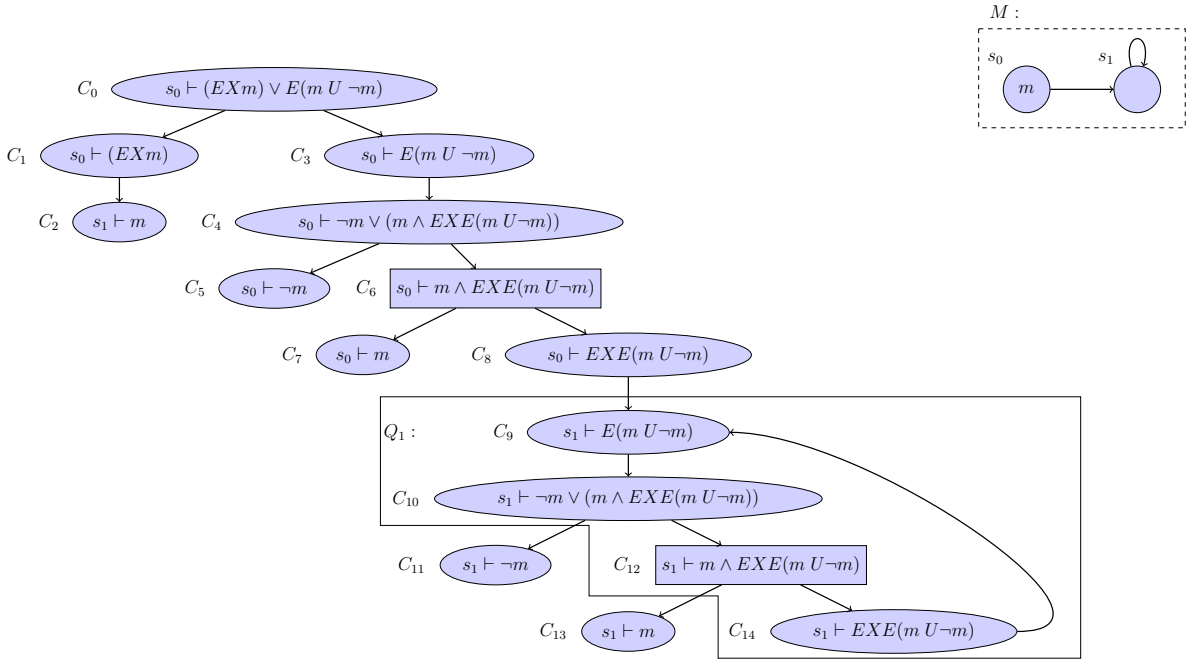

Figura 5.5: Arena de um jogo para a verificação de modelos de uma fórmula CTL $(EXm) \vee E(m\ U \neg m)$ e um KMTS M. Com exceção das configurações da CFCM Q_1, todas as configurações da arena são CFCMs.

$$\delta(s \vdash EX\ \varphi) = \sqcup^{+} \Big(\Big\lfloor \cdot \Big\rfloor_{s' \in \overrightarrow{S}(s)} \delta(s' \vdash \varphi) \sqcap \{M(\{P_2(s,s')\})\} \Big)$$

$$\delta(s \vdash AX\ \varphi) = \sqcup^{+} \Big(\Big\lceil \cdot \Big\rceil_{s' \in \overrightarrow{S}(s)} \delta(s' \vdash \varphi) \sqcup \{M(\{P_1(s,s')\})\} \Big)$$

$$\delta(s \vdash \varphi_1 \vee \varphi_2) = \sqcup^{+} \big(\delta(s \vdash \varphi_1) \sqcup \delta(s \vdash \varphi_2)\big)$$

$$\delta(s \vdash \varphi_1 \wedge \varphi_2) = \sqcup^{+} \big(\delta(s \vdash \varphi_1) \sqcap \delta(s \vdash \varphi_2)\big)$$

$$\delta(s \vdash A(\varphi_1 U \varphi_2)) = \delta(s \vdash \varphi_2 \vee (\varphi_1 \wedge AX\ A(\varphi_1 U \varphi_2)))$$

$$\delta(s \vdash E(\varphi_1 U \varphi_2)) = \delta(s \vdash \varphi_2 \vee (\varphi_1 \wedge EX\ E(\varphi_1 U \varphi_2)))$$

$$\delta(s \vdash A(\varphi_1 R \varphi_2)) = \delta(s \vdash \varphi_2 \wedge (\varphi_1 \vee AX\ A(\varphi_1 R \varphi_2)))$$

$$\delta(s \vdash E(\varphi_1 R \varphi_2)) = \delta(s \vdash \varphi_2 \wedge (\varphi_1 \vee EX\ E(\varphi_1 U \varphi_2)))$$

As configurações geradas pelas regras (5) até (8) têm apenas uma configuração filha, portanto o valor da função δ nestas configurações é equivalente ao valor definido na respectiva configuração filha. Desta forma, se $s \vdash \varphi$ é uma configuração definida a partir de um das regras entre (5) e (8), e $s \vdash \psi$ é a única configuração alcançável, então $\delta(s \vdash \varphi) = \delta(s \vdash \psi)$.

Definição 5.3.2. *Seja M um KMTS, s e s' estados de M, e φ uma fórmula CTL. A verificação de modelos por contração é uma função de coloração χ definida como:*

$$\chi(s \vdash \varphi) = \begin{cases} \top & \textit{sse } \delta(s \vdash \varphi) = \{M\} \\ F & \textit{sse } \delta(s \vdash \varphi) = \emptyset \\ \bot & \textit{caso contrário} \end{cases}$$

Para realizar a verificação de modelos por contração, particionamos a arena do jogo em suas CFCM (definição 3.1.5, capítulo 3) e aplicamos δ sobre as configurações destas componentes baseado na relação de ordem parcial $\leq$ (definição 3.1.6) sobre as CFCM do grafo do jogo. Mais especificamente, aplicamos a função δ sobre as configurações da arena do jogo a partir dos nós terminais para os ascendentes. Uma vez que o valor de $\delta(C_i)$ de uma configuração foi calculada, aplicamos δ à configuração C_w ascendente de C_i. Se C_w pertencer a uma CFCM com mais de uma configuração, isto é, a um ciclo então calculamos o valor de δ para cada uma destas configurações.

Ilustramos a verificação de modelos por contração através de um exemplo. Tomemos a arena da figura 5.5 e os modelos $M_1 = M(\{P_3(s_1, \neg m)\})$ e $M_2 = M(\{P_3(s_1, m)\})$. Inicialmente, calculamos o valor de δ para as configurações terminais C_2, C_5, C_7, C_{11} e C_{13}. $\delta(C_5) = \emptyset$, uma vez que $\neg m \in L(s_0)$. $\delta(C_2)$ e $\delta(C_{13})$ resultam em $\{M_2\}$, pois $m \in L(s_1)$ e, de acordo com a definição 5.3.1, $\delta(C_{12}) = \{M(P_3(s_1, m)\}) = \{M_2\}$. Os valores para as demais configurações é análogo e seguem direto da definição 5.3.1. A tabela 5.2 apresenta os valores de δ para estas configurações.

	C_2	C_5	C_7	C_{11}	C_{13}
$\delta(C_i)$	$\{M_2\}$	$\emptyset$	$\{M\}$	$\{M_1\}$	$\{M_2\}$

Tabela 5.2: Valores de δ para as configurações terminais da figura 5.5.

C_i	$\delta(C_i)$	Iteração			
		1ª	2ª	3ª	4ª
C_9	$\delta(C_{10})$	–	$\delta(C_{10})$	$\delta(C_{10})$	$\{M_1\}$
C_{10}	$\sqcup^+(\delta(C_{11}) \sqcup \delta(C_{12}))$	–	$\sqcup^+\big(\{M_1\} \sqcup \delta(C_{12})\big)$	$\{M_1\}$	$\{M_1\}$
C_{11}	$\{M(P_3(s_1, \neg m))\}$	$\{M_1\}$	$\{M_1\}$	$\{M_1\}$	$\{M_1\}$
C_{12}	$\sqcup^+(\delta(C_{13}) \sqcup \delta(C_{14}))$	–	$\sqcup^+\big(\{M_2\} \sqcap \delta(C_{14})\big)$	$\sqcup^+\big(\{M_2\} \sqcap \delta(C_{14})\big)$	$\emptyset$
C_{13}	$\{M(P_3(s_1, m))\}$	$\{M_2\}$	$\{M_2\}$	$\{M_2\}$	$\{M_2\}$
C_{14}	$\delta(C_9)$	–	$\delta(C_{10})$	$\{M_1\}$	$\{M_1\}$

Tabela 5.3: Valores das iterações de δ para o jogo da figura 5.5

Após calcularmos os valores das configurações terminais, devemos calcular o valor de δ para as configurações ascendentes destas configurações. Calculamos, agora, o valor de

δ para as configurações da CFCM Q_1. Calculamos o valor de δ de forma iterativa. Os valores calculados a cada iteração estão expostos na tabela 5.3. Dados os valores da primeira iteração (valores de δ para os vértices terminais), calculamos o valor parcial de δ para cada uma destas configurações até encontrarmos o valor para o qual δ converge. Da segunda iteração, temos

$$
\begin{aligned}
\delta(C_9) = \delta(C_{10}) &= \sqcup^{+}\Big(\{M_1\} \sqcup \delta(C_{12})\Big) \\
&= \sqcup^{+}\Big(\{M_1\} \sqcup \sqcup^{+}(\{M_2\} \sqcap \delta(C_{14}))\Big) && (5.2)\\
&= \sqcup^{+}\Big(\{M_1\} \sqcup (\{M_2\} \sqcap \delta(C_{14}))\Big) && \textit{(Absorção)} \quad (5.3)\\
&= \sqcup^{+}\Big((\{M_1\} \sqcup \{M_2\}) \sqcap (\{M_1\} \sqcup \delta(C_{14}))\Big) && (5.4)\\
&= \sqcup^{+}\Big((\{M_1, M_2\}) \sqcap (\{M_1\} \sqcup \delta(C_{14}))\Big) && (5.5)\\
&= \sqcup^{+}\Big(\sqcup^{+}(\{M_1, M_2\}) \sqcap (\{M_1\} \sqcup \delta(C_{14}))\Big) && \textit{(Absorção)} \quad (5.6)\\
&= \sqcup^{+}\Big(\{M\} \sqcap (\{M_1\} \sqcup \delta(C_{14}))\Big) && (5.7)\\
&= \sqcup^{+}\Big(\{M_1\} \sqcup \delta(C_{14})\Big) && (5.8)\\
&= \sqcup^{+}\Big(\{M_1\} \sqcup \sqcup^{+}(\{M_1\} \sqcup \delta(C_{14}))\Big) && \textit{(Absorção)} \quad (5.9)\\
&= \sqcup^{+}\Big(\{M_1\} \sqcup \{M_1\} \sqcup \delta(C_{14})\Big) && (\delta(C_{14}) = \delta(C_{10})) \quad (5.10)\\
&= \sqcup^{+}\Big(\{M_1\} \sqcup \delta(C_{10})\Big) && (5.11)\\
&= \sqcup^{+}\Big(\{M_1\}\Big) && (5.12)\\
&= \{M_1\} && (5.13)
\end{aligned}
$$

As fórmulas nos passos (5.3), (5.6) e (5.9) resultam da absorção. Ao analisarmos a parte mais interna da fórmula do passo (5.10), temos $\{M_1\} \sqcup \{M_1\} \sqcup \delta(C_{14})$. Se observamos os valores da segunda iteração da tabela, temos que $\delta(C_{14}) = \delta(C_{10})$. Logo, da equação (5.10), segue que $\delta(C_{10}) = \{M_1\} \sqcup \delta(C_{10})$, equação (5.11), cujo resultado é o próprio conjunto $\{M_1\}$ e $\sqcup^{+}(\{M_1\}) = \{M_1\}$. Logo, $\delta(C_{14}) = \delta(C_{10}) = \{M_1\}$. A tabela 5.3 apresenta na terceira iteração o resultado de $\delta(C_{14})$ e na quarta iteração os valores atualizados de δ para as demais configurações alcançando assim o valor de $\delta(C_9)$ e das demais configurações em Q_1.

Temos que $\delta(C_8) = \delta(C_9)$, logo $\delta(C_8) = \{M_1\}$. Os valores de δ para as demais configurações da arena podem ser calculados de forma ascendente e os apresentamos na tabela 5.4. $\delta(C_1) = \delta(C_2) = \{M_2\}$ e $\delta(C_6) = \sqcup^{+}\big(\delta(C_7) \sqcap \delta(C_8)\big) = \sqcup^{+}(\{M\} \sqcap \{M_1\}) = \{M_1\}$. Quanto à C_4, temos que $\delta(C_4) = \sqcup^{+}\big(\delta(C_5) \sqcap \delta(C_6)\big) = \sqcup^{+}\big(\emptyset \sqcup \{M_1\}\big) = \{M_1\}$. Quanto à C_0, temos $\delta(C_0) = \sqcup^{+}\big(\delta(C_1) \sqcup \delta(C_3)\big) = \sqcup^{+}\big(\{M_1\} \sqcup \{M_2\}\big) = \sqcup^{+}\big(\{M_1, M_2\}\big) = \{M\}$. Temos, portanto, que $\chi(C_0) = \top$. Logo, $[M, s_0 \models \varphi] = \top$, ou seja, o valor da verificação de modelos resulta em verdadeiro.

	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
$\delta(C_i)$	$\{M\}$	$\{M_2\}$	$\{M_2\}$	$\{M_1\}$	$\{M_1\}$	$\emptyset$	$\{M_1\}$	$\{M\}$	$\{M_1\}$

Tabela 5.4: Valores de δ para as configurações C_0 a C_8 do jogo da figura 5.5.

5.3.1 Complexidade

Diferente da verificação de modelos de fórmulas CTL em relação a uma estrutura de Kripke que tem complexidade polinomial no tamanho do modelo de Kripke, o problema da verificação de modelos de um KMTS interpretado como um conjunto de estruturas de Kripke é NP-Completo, como mostramos a seguir.

Teorema 5.3.1. *Seja M um KMTS qualquer, s um estado deste modelo, e φ uma fórmula CTL. Verificar se M satisfaz φ a partir de s ($M, s \models \varphi$) é NP-Completo.*

Prova. Mostramos que o problema é NP e que existe uma redução polinomial de SAT ao problema $M, s \models \varphi$.

- $M, s \models \varphi$ **é NP**

Construamos a seguinte máquina de Turing não determinística que verifica se algum modelo em $K(M)$ satisfaz φ em s.

$MT_\exists$: $< M, s, \varphi >$

- escolha de forma não determinística um modelo k em $K(M)$;
- se $k, s \models \varphi$, aceite; caso contrário rejeite

A máquina $MT_\exists$ acima recebe como entrada um modelo KMTS M, um estado s deste modelo e uma fórmula CTL φ e decide se existe algum modelo expandido de M que satisfaz φ. Usaremos esta máquina para resolver o problema $M, s \models \varphi$. Se $MT_\exists$ não aceitar a entrada $\langle M, s, \varphi \rangle$, então $[M, s \models \varphi] = F$, ou seja,

$$MT_\exists \langle M, s, \varphi \rangle = F \text{ sse } [M, s \models \varphi] = F \tag{5.14}$$

Se $M_\exists$ aceita a entrada $\langle M, s, \varphi \rangle$, então somos capazes de afirmar apenas que existe modelo em $K(M)$ que satisfaz φ a partir de s, mas não podemos decidir se todos a satisfazem. Em outras palavras, podemos afirmar que $M, s \models \varphi$ resulta em $\bot$ ou $\top$.

Dada uma estrutura de Kripke $k \in K(M)$; $k, s \models \varphi$ sse $k, s \not\models \neg\varphi$. Logo,

$$\forall k \in K(M), k, s \models \varphi \text{ sse } k, s \not\models \neg\varphi \tag{5.15}$$

Da identidade acima e de (5.14) temos as seguintes implicações:

1. $[M, s \models \varphi] = F$ sse $M_\exists \langle M, s, \varphi \rangle = F$;

2. $[M, s \models \varphi] = \top$ sse $M_\exists\langle M, s, \varphi\rangle = \top$ e $M_\exists\langle M, \neg\varphi\rangle = F$;
3. $[M, s \models \varphi] = \bot$ sse $M_\exists\langle M, s, \varphi\rangle = \top$ e $M_\exists\langle M, s, \neg\varphi\rangle = \top$;

Logo, a seguinte máquina de Turing não determinística resolve $M, s \models \varphi$:

N: $\langle M, s, \varphi\rangle$

- execute $M_\exists\langle M, s, \varphi\rangle$;
- execute $M_\exists\langle M, s, \neg\varphi\rangle$;
- se a primeira máquina rejeitar a entrada, retorne F;
- se a primeira máquina aceitar e a segunda rejeitar, retorne $\top$;
- caso contrário, retorne $\bot$.

A máquina não determinística N acima resolve o problema $M, s \models \varphi$ em tempo polinomial. Portanto, $M, s \models \varphi$ é NP.

- **Redução polinomial de SAT a** $M, s \models \varphi$.

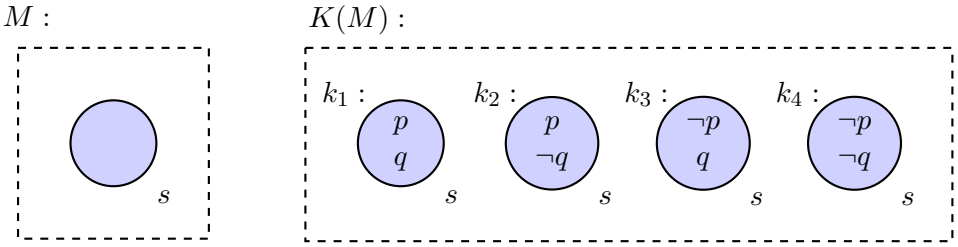

Figura 5.6: Modelo KMTS com todos os literais sobre AP indefinidos no estado s.

Toda fórmula da lógica proposicional é uma fórmula CTL bem formada, portanto não precisamos fazer nenhuma modificação sobre φ. Devemos, construir um modelo M de forma que os modelos expandidos de M correspondam às valorações de φ. Construamos um KMTS $M = (AP, S, R^+, R^-, L)$, onde AP é o conjunto dos símbolos proposicionais da fórmula φ, $S = \{s\}$ com um único estado, $L(s) = \emptyset$ (todos os literais estão indefinidos em s) e $R^- = R^+ = \emptyset$. Desta forma, a cada estrutura de Kripke $k_i = (AP, S, R_i, L_i) \in K(M)$, o conjunto $L_i(s)$ corresponde a um conjunto de literais que pode ser tomado com um valor verdade sobre φ. Desta forma, se $[M, s \models \varphi] = T$ ou $[M, s \models \varphi] = \bot$ a fórmula φ é satisfazível; caso contrário é insatisfazível. Reduzimos, desta forma o problema da satisfazibilidade ao problema $M, s \models \varphi$.

Mostramos um exemplo de redução de SAT a $M, s \models \varphi$. Tomemos a fórmula $\varphi = p \vee q$ e construamos para a redução o modelo da figura 5.6. Observamos que os modelos que satisfazem φ são exatamente os modelos k_1, k_2 e k_3. Temos que para cada um destes modelos, pelo menos um entre p e q ocorrem no estado s e, portanto, satisfazem φ. Logo, $[M, s \models \varphi] = \top$. Vale ressaltar que cada um dos modelos do conjunto expansão representa uma das valorações de φ.

∎

5.3.2 Discussão

Propomos neste capítulo um verificador de modelos para a lógica CTL e um KMTS interpretado como um conjunto de estruturas de Kripke. Definimos inicialmente a semântica da lógica CTL com respeito a um KMTS interpretado como conjunto de modelos. Em seguida, propomos e definimos formalmente operadores de interseção e contração sobre KMTSs que são utilizados na definição do verificador de modelos por contração. Para provar a correção do nosso verificador, introduzimos o conceito de conjunto partição e conjunto partição completo e provamos algumas propriedades destes conjuntos. Propomos uma estrutura para representar estes conjuntos o qual chamamos de Árvore Conjunto Partição e utilizamos esta árvore para provar a correção do nosso verificador. Provamos também que o problema da verificação de modelos por contração é NP-Completo.

Capítulo

6

REFINADOR DE MODELOS KMTS

Neste capítulo, propomos um refinador de modelos KMTSs interpretados como um conjunto de estruturas de Kripke. O refinador é aplicado a um modelo KMTS M, quando a verificação de modelos de M contra uma fórmula CTL φ resulta em $\perp$. A verificação de modelos com jogos fornece as possíveis causas que levam a verificação a resultar em indefinido. Consideramos estas informações para construir uma abstração da arena do jogo com as informações necessárias para gerar os modelos minimais do refinamento. Na seção 6.1 apresentamos o conceito de testemunhas de falhas e na seção 6.2 apresentamos a abstração da arena do jogo ao qual chamamos de grafo de testemunhas. Finalmente, na seção 6.3 apresentamos os algoritmos do refinador.

Para as definições deste capítulo, consideramos A $= (N, T)$, a arena de um jogo qualquer, onde N é o conjunto de configurações de A e $T \subseteq N \times N$ é o conjunto de arestas de A; e $W \subseteq T$ o conjunto de todas as testemunhas de falhas de A. Denotamos a configuração raiz de A por C_0. Usamos, neste capítulo, $C_i \xrightarrow{A} C_j$ para denotar as arestas de uma arena A. Lembramos, também, que escrevemos $[M, s \models \varphi] = \top$ para indicar que a verificação de modelos de uma fórmula φ sobre um KMTS M em um estado s do mesmo resulta em $\top$; escrevemos $[M, s \models \varphi] = F$ para indicar que $[M, s \models \neg\varphi] = \top$; e $[M, s \models \varphi] = \perp$ para indicar que a verificação de modelos resulta em indefinido.

6.1 TESTEMUNHAS DE FALHAS

As testemunhas de falhas são as possíveis causas do resultado indefinido de um jogo. Em outras palavras, as testemunhas de falhas são arestas ou configurações de uma arena que correspondem a transições ou estados do modelo por onde ∃va ou ∀belardo jogaram para não perder. As testemunhas de falha de um jogo compreendem:

1. uma transição *may* genuína (R^-/R^+), pertencente a uma estratégia de não perder de Abelardo (∀), proveniente de um nó AX colorido com $\perp$ para um nó colorido com F ou $\perp$;

2. uma transição *may* genuína, pertencente a uma estratégia de não perder de Eva ($\exists$), proveniente de um nó EX colorido com $\bot$ para um nó colorido com T ou $\bot$;
3. uma configuração terminal, pertencente a uma estratégia de não perder de Eva ou Abelardo, anotada com um literal e colorida com indefinido, isto é, uma configuração $s \vdash l$ em que $l, \neg l \notin L(s)$.

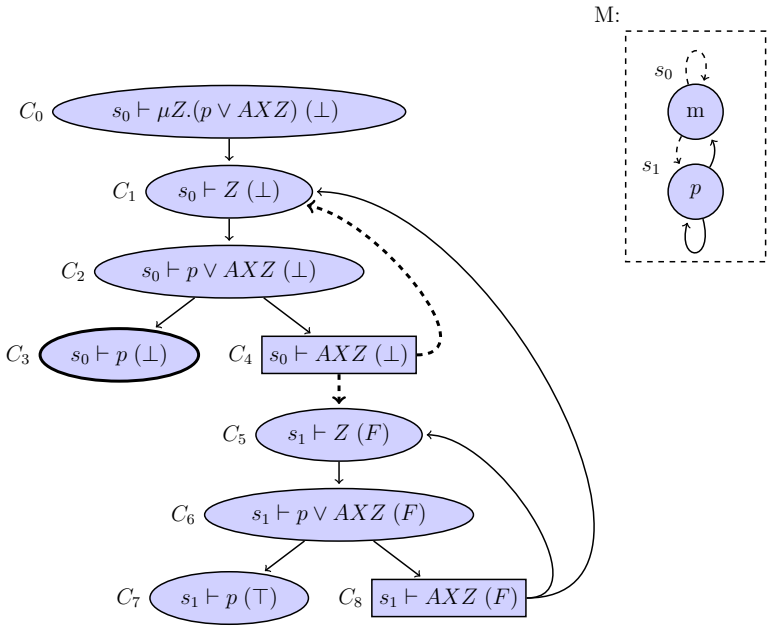

Figura 6.1: Exemplo de verificação de modelos. As arestas e configurações em negrito representam as testemunhas de falhas do jogo.

Tomemos, como exemplo, a arena da figura 6.1. A configuração terminal C_3 está colorida com $\bot$, e de acordo com a terceira restrição da definição acima, C_3 é uma testemunha de falha. As demais testemunhas de falha desta arena são as arestas $C_4 \xrightarrow{A} C_5$ e $C_4 \xrightarrow{A} C_1$.

As testemunhas de falhas são utilizadas para reparar o modelo KMTS, a fim de permitir que $\exists$va tenha estratégia de ganhar a partir da configuração inicial do jogo, ou seja, que a verificação de modelos resulte em $\top$. Como queremos modificar o modelo KMTS de forma a satisfazer uma fórmula φ a partir de um estado s, devemos aplicar mudanças sobre o modelo de forma a alterar a arena de jogo e garantir a Eva uma estratégia de ganhar a partir de s. Em outras palavras, dada uma testemunha de falha $C_m \xrightarrow{A} C_n$ ou C_m, devemos aplicar uma ou mais mudanças de forma a garantir uma estratégia de ganhar a Eva em C_m. A função $Change$, definida a seguir, mapeia cada testemunha de falha de uma arena a uma mudança que, aplicada ao modelo do jogo, remove a respectiva testemunha de falha da arena. A mudança definida pela função $Change$ tem como objetivo garantir a Eva estratégia de ganhar em C_m, ou seja, transformar uma estratégia de não perder de Eva em uma estratégia de ganhar.

Definição 6.1.1. *Sejam $A = (N, T)$ a arena de um jogo para a verificação de modelos $[M, s \models \varphi] = \bot$, $W \subseteq T$ o conjunto das testemunhas de falhas de A, e Ω o conjunto de todas as mudanças aplicáveis sobre M. A função $Change : W \to \Omega$ mapeia cada testemunha de falha de A em uma mudança aplicável sobre M e é definida como:*

$$\begin{aligned} Change(s \vdash l) &= \{P_3(s, l)\}; \\ Change(s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi) &= \{P_2(s, s')\} \\ Change(s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi) &= \{P_1(s, s')\} \end{aligned}$$

Nas expressões acima, l é um literal, e se Eva joga em uma configuração $C_m = s \vdash l$ terminal colorida com $\bot$, então C_m é uma testemunha de falha e a atribuição de l ao estado s (operação de mudança primitiva $P_3(s, l)$) define uma estratégia de ganhar e Eva em C_m.

Dadas duas configurações $C_m = s \vdash EX\varphi$ e $C_n = s' \vdash \varphi$ de uma arena A, onde C_n é nó filho de C_m. Se $C_m \xrightarrow{A} C_n$ é uma testemunha de falha, então por definição Eva joga por uma transição *may* genuína. A transformação da aresta (s, s') faz com que Eva jogue consistentemente por uma aresta *must*. Lembramos que jogar por uma aresta *must* não garante a Eva estratégia de ganhar em C_n, mas é condição necessária. Desta forma, para aproximar Eva de uma estratégia de ganhar é necessário que transformemos a transição (s, s') do modelo em uma transição *must* (lembramos que as estratégias de ganhar e perder de Eva e Abelardo estão definidas na seção 3.2 do capítulo 3).

Se Abelardo joga por uma transição $C_m \xrightarrow{A} C_n$, onde $C_m = s \vdash AX\varphi$ e $C_n = s' \vdash \varphi$, então (s, s') é uma transição *may* genuína, pela definição de testemunha de falha, e a remoção desta transição remove do jogo uma estratégia de não perder de Abelardo a partir de C_m.

6.2 GRAFO DE TESTEMUNHAS

O grafo de testemunhas é uma abstração da arena de um jogo de verificação de modelos e captura as informações necessárias para realizar o refinamento de modelos KMTS. Fixamos C_0 como a configuração raiz da arena de um jogo.

Para a definição do grafo de testemunhas, consideramos duas informações: as testemunhas de falhas do jogo e os menores ancestrais comuns entre as testemunhas da arena.

Definição 6.2.1. *Dadas duas testemunhas de falha $t_1 = C_i \xrightarrow{A} C_{i+1}$ (ou $t_1 = C_i$) e $t_2 = C_j \xrightarrow{A} C_{j+1}$ (ou $t_2 = C_j$) de uma arena A, uma configuração C_k é dita menor ancestral comum de t_1 e t_2 sse existe um caminho $P_0 = C_k \xrightarrow{A} C_{k+l} \xrightarrow{A} \ldots \xrightarrow{A} C_i$ e um caminho $P_1 = C_k \xrightarrow{A} C_{k+l'} \xrightarrow{A} \ldots \xrightarrow{A} C_j$ e não existe uma configuração $C_{k'}$, $C_{k'} \neq C_k$, tal que $C_{k'} \in P_0 \setminus \{C_k\}$ e $C_{k'} \in P_1 \setminus \{C_k\}$.*

Definição 6.2.2. *Seja $A = (N, T)$ a arena de um jogo, W o conjunto das testemunhas de falhas de A, e $D_A = \{(v) \mid v \in N \text{ ou } v \in W\}$ o conjunto das configuração e testemunhas de falhas de uma arena. O Grafo de Testemunhas da arena A é o grafo $G = (V, E)$, onde*

o conjunto $V \subseteq D_A$ de vértices satisfaz as condições (i),(ii),(iii), e o conjunto E de arestas satisfaz a condição (iv).

(i) $(C_0) \in V$;

(ii) *Se* $v \in W$, *então* $(v) \in V$;

(iii) *para quaisquer vértices* $v_1, v_2 \in V$, *se* $v_1, v_2 \in W$ *e existe uma configuração* $C_x \in N$, *tal que* C_x *é o menor ancestral comum entre* v_1 *e* v_2, *então* $(C_x) \in V$;

(iv) *Seja* $v_1, v_2 \in V$, *t.q,* $v_1 = (C_j)$ *ou* $v_1 = (C_i \xrightarrow{A} C_j)$, $v_2 = (C_m)$ *ou* $v_2 = (C_m \xrightarrow{A} C_n)$. *Se existe um caminho* $P = C_j \xrightarrow{A} \ldots \xrightarrow{A} C_m$ *em A, e não existe configuração* $v_3 \in V$ *tal que* $v_3 = (C_x)$ *ou* $v_3 = (C_x \xrightarrow{A} C_y)$ *e* $v_3 \notin P$, *com* $v_3 \neq v_1$, $v_3 \neq v_2$ *então* $v_1 \rightarrow v_2 \in E$.

A raiz de um grafo de testemunhas representa a configuração raiz da arena sobre o qual ele é construído, esta propriedade é representada pela condição (i) da definição acima. Todas as testemunhas de falhas são capturadas pelo grafo de testemunhas – condição (ii) –, bem como os menores ancestrais comuns entre elas: a condição (iii) garante esta restrição. As arestas dos grafos de testemunhas abstraem caminhos da arena do jogo. Assim, se existe um caminho entre duas configurações C_i e C_j, onde C_i e C_j são menores ancestrais comuns, então o caminho entre C_i e C_j será representado por uma aresta no grafo de testemunhas, desde que um terceiro menor ancestral comum não pertença a este caminho. O mesmo vale para caminhos entre testemunhas de falhas e/ou menores ancestrais comum. A condição (iv) garante, exatamente, tal restrição, ou seja, que estas abstrações de caminho serão capturadas pelo grafo de testemunhas e representadas através de arestas no mesmo.

Neste capítulo, reservamos os termos configuração e vértice para nos referir respectivamente aos nós da arena de um jogo e de um grafo de testemunhas. Denotamos uma configuração por C_n e um vértice por v_n, com $n \in \mathbb{N}$. Denotamos as arestas de um grafo de testemunhas por $v_i \rightarrow v_j$. Chamamos os vértices anotados com testemunhas de falhas de vértices testemunhas, e os demais de vértices configurações. Seja v um vértice qualquer de um grafo de testemunhas G, onde $v = (C_m)$ ou $v = (C_m \xrightarrow{A} C_n)$; se Eva joga em C_m então dizemos que v é um vértice Eva, caso contrário, ou seja, Abelardo joga em C_m, dizemos que v é um vértice Abelardo. Ilustramos na figura figura 6.3 o grafo de testemunhas da arena A apresentada na figura 6.2. O vértice $v_6 = (C_6 \xrightarrow{A} C_3)$ deste grafo representa uma testemunha de falha da arena A, enquanto o vértice $v_7 = (C_{15})$ representa um menor ancestral comum da arena A. Portanto, os vértices v_6 e v_7 são respectivamente vértices testemunhas e configuração. Além disso, v_7 é um vértice Eva, enquanto v_6 é um vértice Abelardo; pois a configuração C_6, cabeça da aresta $C_6 \xrightarrow{A} C_3$ anotada em v_6, é uma configuração Abelardo, e a configuração C_{15}, anotada em v_7, é uma configuração Eva. De forma análoga, os vértices $v_2 = (C_4)$ e $v_8 = (C_{15} \xrightarrow{A} C_{16})$ são respectivamente vértices Abelardo e Eva.

Explicamos a construção de um grafo de testemunhas de uma arena através de um exemplo. Consideramos, para tanto, a arena apresentada na figura 6.2 e o seu respectivo grafo de testemunhas na figura 6.3. Os vértices Eva são representados por elipses, enquanto os vértices Abelardo são representados por retângulos.

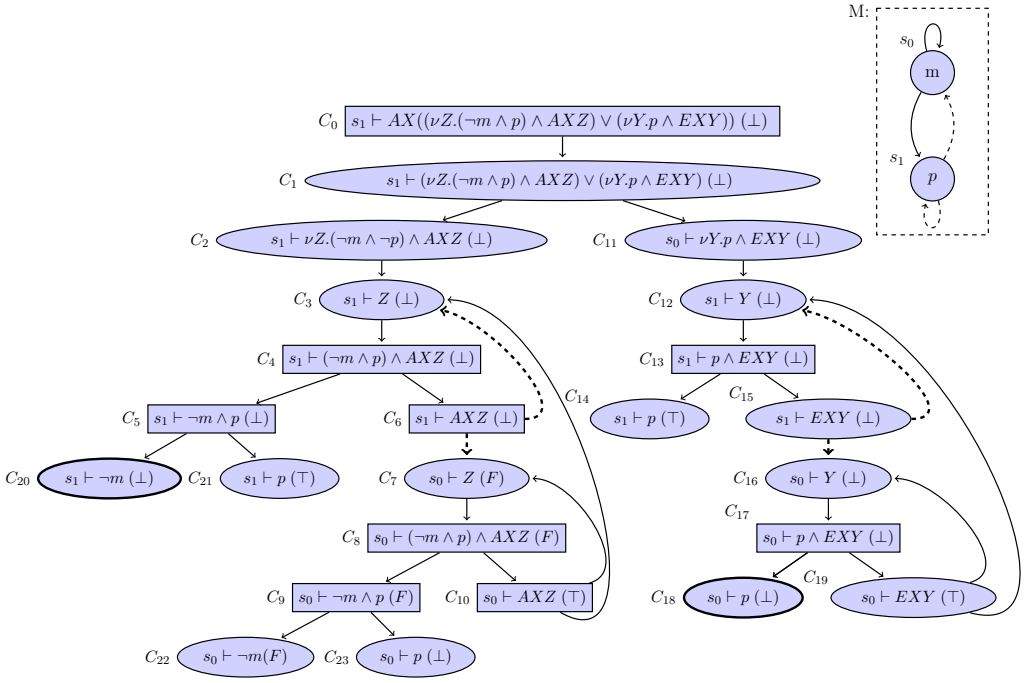

Figura 6.2: Arena de um jogo com modelo M e fórmula $\varphi = AX((\nu Z.(\neg m \wedge p) \wedge AXZ) \vee (\nu Y.p \wedge EXY))$ a partir do estado s_1 do modelo. As arestas e configurações em negrito representam as testemunhas de falhas do jogo.

O vértice v_0 da figura 6.3 representa a configuração raiz C_0 da figura 6.2. As testemunhas de falhas da arena são: $C_{20}, C_{18}, C_6 \xrightarrow{A} C_7, C_6 \xrightarrow{A} C_3, C_{15}, \xrightarrow{A} C_{16}$ e $C_{15} \xrightarrow{A} C_{12}$. As testemunhas de falhas são capturadas pelo grafo de testemunhas, respectivamente, pelos vértices v_3, v_9, v_5, v_6, v_8 e v_{10}. Os menores ancestrais comuns, C_1, C_4, C_6 e C_{15}, entre as testemunhas são representados no grafo pelos vértices v_1, v_2, v_4 e v_7. Logo, o grafo de testemunhas do nosso exemplo é composto por um conjunto de vértices $V = \{v_0, v_1, \ldots, v_{10}\}$.

As arestas são definidas de acordo com a restrição (iv) da definição 6.2.2. Tomemos os vértices v_1 e v_7, que representam as configurações C_1 e C_{15}. O caminho entre estas duas configurações é composto pelas configurações $C_1 \xrightarrow{A} C_{11} \xrightarrow{A} C_{12} \xrightarrow{A} C_{13} \xrightarrow{A} C_{15}$. Observamos que não existe nenhum vértice em V que represente alguma destas configurações, portanto tal caminho será abstraído por uma aresta pelo grafo de testemunhas. Tomemos agora, os vértices v_2 e v_6 que representam, respectivamente, a testemunha C_4 e a aresta $C_6 \xrightarrow{A} C_3$ da arena do jogo. Podemos observar que a configuração C_6, representada pela configuração v_4, pertence ao caminho entre a testemunha de falha em questão e a configuração C_4. Portanto, tal caminho não deve ser abstraído. Contudo, o caminho entre a configuração representada por v_4, ou seja, C_6, e a testemunha de falha $C_6 \xrightarrow{A} C_3$

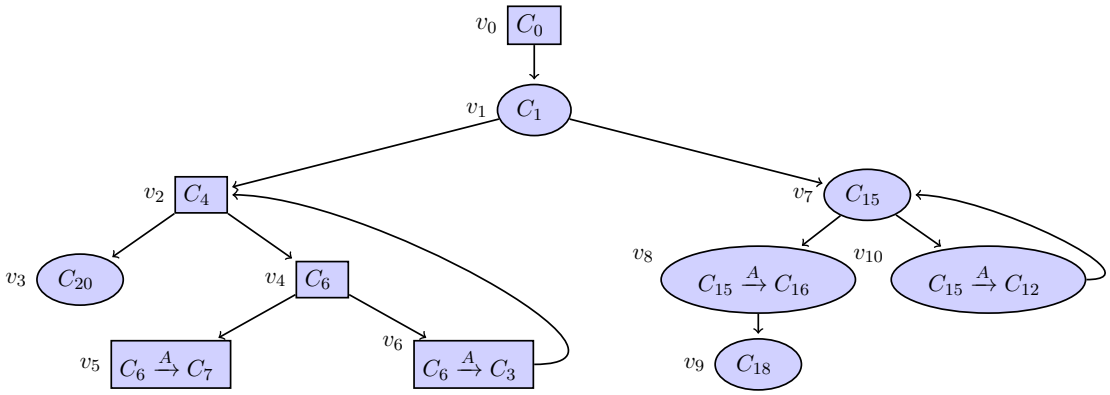

Figura 6.3: Grafo de testemunhas da arena na figura 6.2.

representada por v_6, envolve somente estes vértices. Logo, tal caminho é abstraído pela aresta $v_4 \to v_6$ no grafo de testemunhas.

6.3 REFINAMENTO SOBRE GRAFO DE TESTEMUNHAS

Apresentamos nesta seção algoritmos sobre o grafo de testemunhas de uma arena que calculam as mudanças que reparam o modelo de um jogo que resultou em indefinido.

Sejam A a arena do jogo da verificação de modelos $M, s_0 \models \varphi$, G o grafo de testemunhas de A e v_r um vértice de G, onde v_r é da forma $(C_m \xrightarrow{A} C_n)$ ou (C_m), e X' uma mudança aplicável sobre M. Dizemos que X' define uma estratégia de ganhar a Eva em v_r sse Eva tem estratégia de ganhar em C_m após a aplicação de X' sobre M. Se X' define uma estratégia de ganhar a Eva no vértice raiz v_0 de G, então X' repara o modelo KMTS M em que a verificação de modelos sobre uma propriedade φ resultou em $\bot$. Em outras palavras, se a aplicação de X' sobre M define uma estratégia de ganhar a Eva em v_0, então $[M', s_0 \models \varphi] = \top$, onde $M' = M(X')$. Dessa forma, podemos reduzir o problema do refinamento de $M, s_0 \models \varphi$ em encontrar as mudanças que aplicadas a M definem uma estratégia de ganhar a Eva em v_0.

Apresentamos nesta seção algoritmos capazes de encontrar mudanças que definem estratégias de ganhar a Eva em qualquer vértice de um grafo de testemunhas. Inicialmente, damos uma ideia geral do processo e, em seguida, apresentamos os algoritmos de refinamento.

Definição 6.3.1. *Seja A um conjunto qualquer, $B \subseteq 2^A$ e $C \subseteq 2^A$ coleções de subconjuntos de A. Definimos a união simétrica, $\uplus$, entre B e C como:*

$$B \uplus C = \{X \cup Y \mid X \in B \text{ e } Y \in C\}.$$

A união simétrica dos conjuntos $\alpha = \{\{a\}, \{b\}\}$ e $\beta = \{\{c\}, \{d\}\}$, por exemplo, é dado por $\alpha \uplus \beta = \{\{a, c\}, \{a, d\}, \{b, c\}, \{b, d\}\}$. Cada elemento de α é combinado com cada elemento em β.

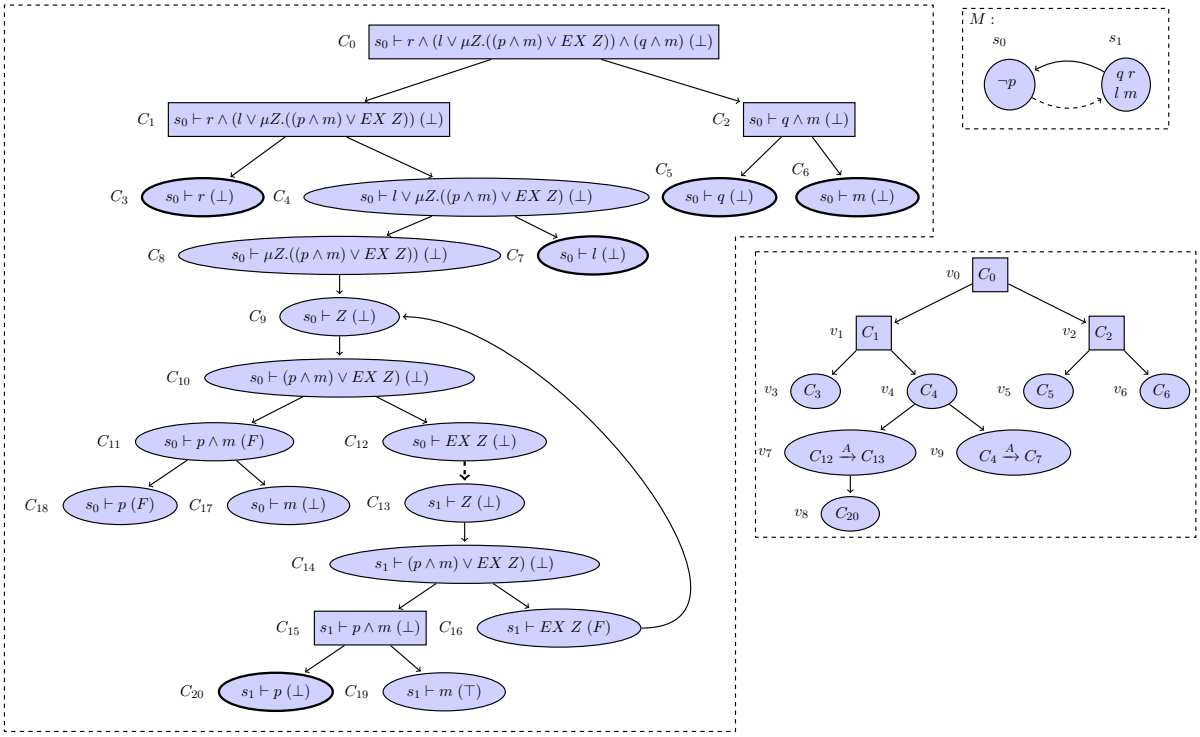

Figura 6.4: À esquerda, arena A de um jogo de verificação de modelos para a fórmula CTL $\varphi = r \wedge (l \vee EF\ (p \wedge m)$ escrita em μ-calculus e KMTS M (lado superior direito). À direta, o grafo de testemunhas da arena A.

Para encontrarmos as mudanças minimais que definem estratégias de ganhar a Eva em um vértice qualquer de um grafo de testemunhas, podemos buscar as mudanças minimais dos nós filhos do respectivo vértice e combiná-los. Estas combinações são feitas de forma diferente a depender do tipo do vértice: Abelardo ou Eva. Para um vértice Eva, é suficiente unir os conjuntos de mudanças dos nós filhos, enquanto para vértice Abelardo é necessário combinar os conjuntos das mudanças dos nós filhos. Para os nós terminais é suficiente tomar o conjunto de mudança designado pelo respectivo vértice, ou seja, o conjunto designado pela função $Change$. Por exemplo, para o vértice terminal $v_8 = (C_{20})$ da figura 6.4 é suficiente definir o literal p ao estado s_1 do modelo para garantir a Eva estratégia de ganhar em v_8, ou seja, é suficiente tomar a mudança $X_1 = \{P_3(s_1, p)\}$. O conjunto de mudanças minimais, portanto, de v_8 é o conjunto unitário $\{X_1\} = \{Change(v_8)\}$. Os conjuntos de mudanças minimais dos vértices do grafo de testemunhas da figura 6.4 são apresentados na tabela 6.1. Para o vértice Eva $v_4 = (C_4)$, é suficiente que Eva tenha estratégia de ganhar ao jogar por v_7 ou por v_9, portanto o conjunto das mudanças minimais em v_4 é $Min(v_7) \cup Min(v_9)$, onde $Min(v_r)$ denota o conjunto das mudanças minimais de um vértice v_r. Por outro lado, para o vértice $v_2 = (C_2)$ precisamos remover as estratégias de não perder de Abelardo a partir dos vértices terminais v_5 e v_6, filhos de v_2. As mudanças minimais que removem as estratégias de não perder de Abelardo em v_5 e v_6 são respectivamente $Change(v_5)$ e $Change(v_6)$.

v_i	$Min(G_A, v_i)$	
	Regra de Formação	Conj. Mudanças
v_0	$Min(G_A, v_1) \uplus Min(G_A, v_2)$	$\{P_2(s_0, s_1), P_3(s_1, p), P_3(s_0, r), P_3(s_0, q), P_3(s_0, m)\}$, $\{P_3(s_0, l), P_3(s_0, r), P_3(s_0, q), P_3(s_0, m)\}$
v_1	$Min(G_A, v_3) \uplus Min(G_A, v_4)$	$\{P_2(s_0, s_1), P_3(s_1, p), P_3(s_0, r)\}$, $\{P_3(s_0, l), P_3(s_0, r)\}$
v_2	$Min(G_A, v_5) \cup Min(G_A, v_6)$	$\{P_3(s_0, q), P_3(s_0, m)\}$
v_3	$Change(v_3)$	$\{P_3(s_0, r)\}$
v_4	$Min(G_A, v_7) \cup Min(G_A, v_9)$	$\{P_2(s_0, s_1), P_3(s_1, p)\}$, $\{P_3(s_0, l)\}$
v_5	$Change(v_5)$	$\{P_3(s_0, q)\}$
v_6	$Change(v_6)$	$\{P_3(s_0, m)\}$
v_7	$\{Change(v_7)\} \uplus \textsc{Repair}(G_A, v_8)$	$\{P_2(s_0, s_1), P_3(s_1, p)\}$
v_8	$Change(v_8)$	$\{P_3(s_1, p)\}$
v_9	$Change(v_9)$	$\{P_3(s_0, l)\}$

Tabela 6.1: Tabela com o conjunto $Min(G_A, v_i)$ das mudanças minimais de cada vértice do grafo de testemunhas G_A da figura 6.4.

Ao combinarmos estas duas mudanças, definimos uma estratégia de ganhar a Eva em v_2. Esta combinação pode ser feita através da operação de união simétrica, isto é, $Min(v_2) = \{Min(G_A, v_5)\} \uplus \{Min(G_A, v_6)\} = \{Change(v_5)\} \uplus \{Change(v_6)\}$. De forma análoga, para os demais vértices Abelardo é suficiente aplicar a operação de união simétrica aos nós filhos. Por exemplo, o conjunto das mudanças minimais de v_0 é dado por $Min(G_A, v_1) \uplus Min(G_A, v_2)$.

Para este exemplo em específico, a combinação e união das mudanças minimais dos nós filhos de um vértice foram suficientes para encontrar as mudanças minimais deste vértice. Contudo, nem sempre a simples combinação ou união de mudanças minimais resulta em uma mudança minimal. Para estes casos, é necessário que apliquemos um filtro sobre o conjunto resultante. Por exemplo, o vértice Eva v_0 da figura 6.5 tem dois vértices filhos v_1 e v_2, e a princípio o conjunto das mudanças minimais de v_0 é $Min(G_A, v_1) \cup Min(G_A, v_2) = \{\{P_3(s_0, q)\}, \{P_2(s_0, s_0), P_3(s_0, q)\}, \{P_1(s_0, s_0), P_3(s_0, m)\}\}$. Percebemos, contudo, que a mudança $\{P_2(s_0, s_0), P_3(s_0, q)\}$ não é minimal e, portanto, deveria ser descartada. Desta forma, o conjunto das mudanças minimais de v_0 é $\{\{P_3(s_0, q)\}, \{P_2(s_0, s_0), P_3(s_0, m)\}\}$. Ou seja, devemos selecionar de $Min(G_A, v_1) \cup Min(G_A, v_2)$ as mudanças minimais. Os conjuntos das mudanças minimais de cada vértice do grafo de testemunhas da figura 6.6 é apresentado na tabela 6.2.

O algoritmo MinRepair, apresentado a seguir na página 71, encontra as mudanças

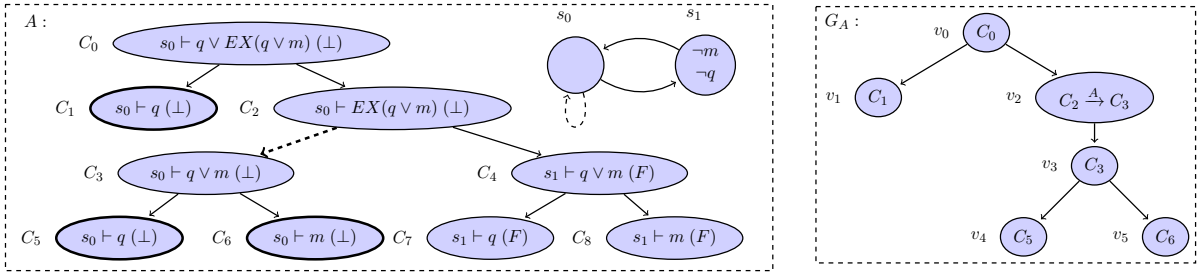

Figura 6.5: Arena A de um jogo de verificação de modelos com fórmula CTL $q \vee EX(q \vee m)$ e modelo KMTS M, e o grafo de testemunhas G_A de A.

minimais de um vértice v_r qualquer de um grafo de testemunhas G_A dado. O algoritmo considera as Componentes Fortemente Conexas Maximais – CFCM – (definição 3.1.5, capítulo 3) de um grafo de testemunhas. Caso v_r pertença a um ciclo do grafo de testemunhas, a responsabilidade de encontrar as mudanças minimais é atribuída ao algoritmo 3 MinCycle, caso contrário é atribuída ao algoritmo PartialRepair.

v_i	$Min(G_A, v_i)$	
	Regra de Formação	Conj. Mudanças
v_0	$Minimal(Min(G_A, v_1) \uplus Min(G_A, v_2))$	$\{P_3(s_0, q)\}$, $\{P_2(s_0, s_0), P_3(s_0, m)\}$
v_1	$Change(v_1)$	$\{P_3(s_0, q)\}$
v_2	$Change(v_2) \uplus Min(G_A, v_3)$	$\{P_2(s_0, s_0), P_3(s_0, s_q)\}$ $\{P_2(s_0, s_0), P_3(s_0, m)\}$
v_3	$Min(G_A, v_4) \cup Min(G_A, v_5)$	$\{P_3(s_0, q)\}$ $\{P_3(s_0, m)\}$
v_4	$Change(v_4)$	$\{P_3(s_0, q)\}$
v_5	$Change(v_5)$	$\{P_3(s_0, m)\}$

Tabela 6.2: Tabela com os valores das mudanças minimais $Min(G_A, v_i)$ em cada vértice v_i do grafo de testemunhas G_A da figura 6.5. $Minimal(\alpha)$ na primeira linha seleciona as mudanças minimais de um conjunto α.

Definição 6.3.2. *Dizemos que uma mudança X qualquer é inconsistente sse existem operações primitivas $\varrho_1, \varrho_2 \in X$ tal que $\varrho_1 = \neg\varrho_2$.*

Dizemos que uma mudança é consistente se e somente se ela não for inconsistente. Explicamos inicialmente o algoritmo PartialRepair que trata dos vértices que não pertencem a ciclos do grafo de testemunhas e em seguida explicamos o algoritmo MinCycle (página 76) .

Algoritmo 1: MinRepair(G, v_r)

Entrada: Um grafo G de testemunhas, e um vértice Eva v_r de G
Saída: Conjunto de mudanças minimais que garantem a Eva estratégia de ganhar em v_r

```
1 Q := componentFC(G, v_r);
2 if |Q| = 1 then
3 |   PartialRepair(G, v_r);
4 else
5 |   MinCycle(G, v_r, Q);
6 end
```

As linhas 2 a 13 do algoritmo PartialRepair cuidam dos vértices Evas, enquanto as linhas 14 a 23 calculam as mudanças minimais de vértices Abelardo. A linha 24 realiza o filtro das mudanças calculadas e retorna somente as minimais. Para os vértices Eva terminais (linha 2) é suficiente tomarmos a mudança designada pelo respectivo vértice. Para os vértices Eva v_r da forma $(s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)$, linha 4, devemos tomar as mudanças minimais do único vértice filho de v_r e transformar a transição (s, s') em transição *must*, pois Eva ganha somente se jogar consistentemente (por transições *must*). Desta forma, as mudanças minimais de v_r é dada pela combinação de $\{Change(v_r)\}$ (que transforma (s, s') em transição *must*) com as mudanças minimais do vértice filho de v_r (linha 5). Para garantir a Eva estratégia de ganhar em um vértice v_r da forma $(s \vdash EX\varphi)$, é suficiente garantir estratégia de ganhar a Eva em pelo menos um de seus vértices filhos. Se um vértice v_i filho de v_r for da forma $(s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)$, conjunto γ na linha 9 do algoritmo, então precisamos transformar a transição *may* genuína (s, s') em *must* e garantir a Eva estratégia de ganhar no vértice filho de v_i. Caso contrário, é suficiente garantir estratégia de ganhar a Eva em v_i. Para os vértices Eva da forma $(s \vdash \varphi \vee \psi)$, é suficiente definir estratégia de ganhar a Eva em pelo menos um de seus vértices filhos. Logo, calculamos as mudanças que definem estratégias de ganhar nos vértices em seus vértices filhos e aplicamos o filtro sobre o conjunto calculado, a fim de garantir a minimalidade das mudanças (linha 24).

Para os vértices Abelardo, é suficiente que combinemos as mudanças minimais dos vértices filhos e apliquemos o filtro sobre o conjunto calculado. Explicamos em mais detalhes alguns casos particulares para os vértices Abelardo. Particionamos os vértices filhos de v_r em dois conjuntos: um conjunto γ (linha 19) com os vértices da forma $(s \vdash AX \xrightarrow{A} s' \vdash \varphi)$ e um conjunto ω (linha 20) com os vértices que não são deste tipo. Para removermos as estratégias de não perder de Abelardo, devemos combinar as mudanças dos vértices filhos de Abelardo. Temos duas formas de remover uma estratégia de não perder de Abelardo ao jogar por um vértice v_i do conjunto γ: podemos remover

Algoritmo 2: PARTIALREPAIR(G, v_r)

Entrada: Um grafo G de testemunhas, e um vértice Eva v_r de G
Saída : Conjunto de mudanças que garantem a Eva estratégia de ganhar em v_r

1 $conjResultado := \emptyset$;
2 **if** $v_r = (s \vdash l)$ **then**
3 | **return** $\{Change(v_r)\}$;
4 **else if** $v_r = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)$ **then**
5 | $conjResultado := \{Change(v_r)\} \uplus$ MINREPAIR(CHILD(v_i))) ;
6 **else if** $v_r = (s \vdash \varphi \vee \psi)$ **then**
7 | $conjResultado := \bigcup_{v_i \in \text{CHILDREN}(v_r)}$ MINREPAIR(G, v_i);
8 **else if** $v_r = (s \vdash EX\varphi)$ **then**
9 | $\gamma := \{v_i \in \text{CHILDREN}(v_r) \mid v_i = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)\}$;
10 | $\omega := \text{CHILDREN}(v_r) \setminus \gamma$;
11 | $\beta := \bigcup_{v_i \in \gamma} \{Change(v_i)\} \uplus$ MINREPAIR$(G, \text{CHILD}(v_i))$;
12 | $conjResultado := \beta \cup \bigcup_{v_j \in \omega}$ MINREPAIR(G, v_j);
13 **end**
/* v_r é um vértice Abelardo */
14 **if** $v_r = (s \vdash \varphi \wedge \psi)$ **then**
15 | $conjResultado := \biguplus_{v_i \in \text{CHILDREN}(v_r)}$ MINREPAIR(G, v_i) ;
16 **else if** $v_r = (s \vdash AX\ \varphi \xrightarrow{A} s \vdash \varphi)$ ***and*** PARENT$(v_r) \neq (s \vdash AX\ \varphi)$ **then**
17 | $conjResultado := \{Change(v_r)\} \cup$ MINREPAIR$(G, \text{CHILD}(v_r))$;
18 **else** // $v_r = (s \vdash AX\varphi)$
19 | $\gamma := \{v_i \in \text{CHILDREN}(v_r) \mid v_i = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi)\}$;
20 | $\omega := \{v_i \in \text{CHILDREN}(v_r) \mid v_i \notin \gamma\}$;
21 | $\beta = \biguplus_{v_i \in \gamma} \{Change(v_i)\} \cup$ MINREPAIR$(\text{CHILD}(v_i))$;
22 | $conjResultado := \beta \uplus \biguplus_{v_i \in \omega}$ MINREPAIR(G, v_i);
23 **end**
24 **return** $\{X \in conjResultado \mid X$ é consistente e $\nexists Y \in conjResultado;\ Y \subset X\}$;

a transição *may* (s, s') $(Change(v_i) = \{P_1(s, s')\})$ ou aplicar as mudanças minimais que definem estratégias de ganhar a Eva no vértice filho de v_i. Portanto, dado um vértice $v_i \in \gamma$, as mudanças que removem as estratégias de não perder de Abelardo ao jogar por v_i é dado pelo conjunto $\beta_i = \{Change(v_i)\} \cup$ MINREPAIR(G_A, v_j), onde v_j é o vértice filho de v_i e G_A o grafo de testemunhas ao qual v_i pertence. O conjunto das mudanças que removem as estratégias de não perder de Abelardo em todos os vértices de γ é dado, portanto, pela combinação dos conjuntos β_i associados a cada vértice $v_i \in \gamma$: conjunto β da linha 21. Logo, para removermos as estratégias de não perder de Abelardo em v_r,

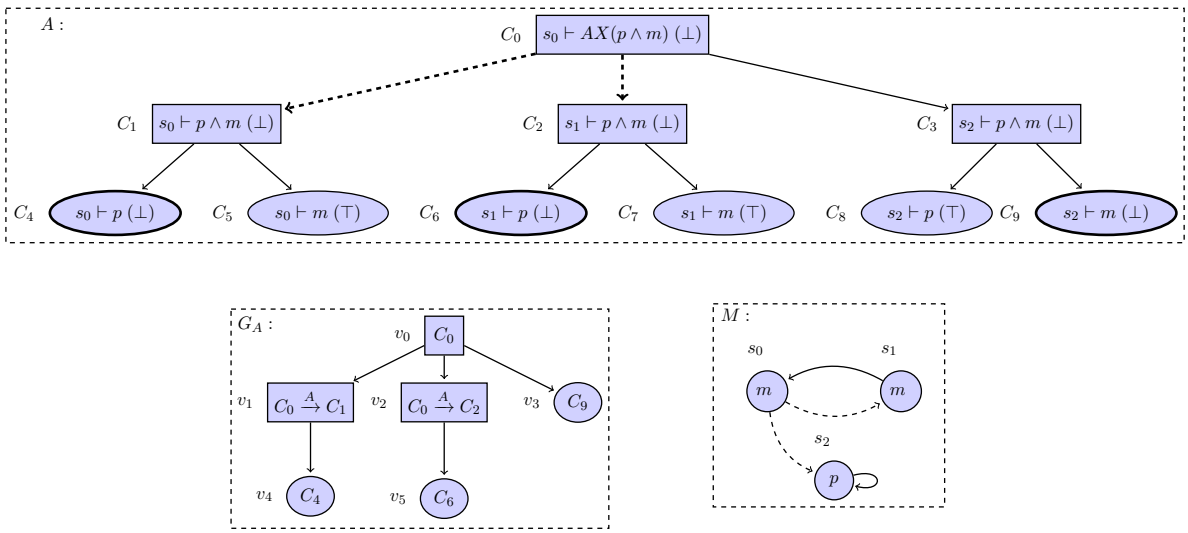

Figura 6.6: Arena A do jogo de verificação de modelos com fórmula CTL $AXp \wedge m$ e modelo KMTS M, e o grafo de testemunhas G_A de A.

combinamos as mudanças de não perder dos vértices em γ com os em ω (linha 22).

Tomemos como exemplo o vértice $v_0 = (C_0) = \{s_0 \vdash AX(p \wedge m)\}$ da figura 6.6. Temos que $\gamma = \{v_1, v_2\}$ e $\omega = \{v_3\}$. Para removermos as estratégias de não perder de Abelardo em v_1 tomamos a mudança $Change(v_1) = \{P_1(s_0, s_0)\}$ ou as mudanças minimais em seu vértice filho v_4 ($\{P_3(s_0, p)\}$). Da mesma forma, para removermos as estratégias de não perder de Abelardo em v_2, tomamos a mudança $Change(v_2) = \{P_1(s_0, s_1)\}$ ou as mudanças minimais em seu vértice filho v_4 ($\{P_3(s_1, p)\}$). Segue que o conjunto das mudanças que removem as estratégias de não perder de Abelardo em v_1 e v_2 são respectivamente $\beta_1 = \{\{P_1(s_0, s_0)\},\ \{P_3(s_0, p)\}\}$ e $\beta_2 = \{\{P_1(s_0, s_1)\},\ \{P_3(s_1, p)\}\}$. O conjunto das mudanças que removem as estratégias de não perder de Abelardo nos vértices em γ é dado pela combinação de β_1 com β_2, ou seja, o conjunto $\beta = \beta_1 \uplus \beta_2$. O conjunto das mudanças que define estratégia de ganhar a Eva é dado pela combinação das mudanças que removem as estratégias de não perder de Abelardo nos vértices de γ e ω. O conjunto ω tem apenas um vértice, v_3, cujo conjunto de mudanças minimais é $Change(v_3) = \{P_3(s_2, m)\}$. Logo, o conjunto das mudanças que define estratégia de ganhar a Eva em v_0 é o conjunto obtido pela união simétrica de β com o conjunto unitário de mudanças $\{\{P_3(s_2, m)\}\}$, ou seja,

$$
\begin{aligned}
\beta \uplus \{\{P_3(s_2,m)\}\} &= \beta_1 \uplus \beta_2 \uplus \{P_3(s_2,m)\} \\
&= \{\ \{P_1(s_0,s_0)\},\ \{P_3(s_0,p)\}\ \} \uplus \{\ \{P_1(s_0,s_1)\},\ \{P_3(s_1,p)\}\ \} \\
&\quad \uplus \{\ \{P_3(s_2,m)\}\ \} \\
&= \Big\{ \{P_1(s_0,s_0), P_1(s_0,s_1), P_3(s_2,m)\}, \\
&\quad\ \{P_1(s_0,s_0), P_3(s_1,p), P_3(s_2,m)\},
\end{aligned}
$$

$$\{P_3(s_0, p), P_1(s_0, s_1), P_3(s_2, m)\},$$
$$\{P_3(s_0, p), P_3(s_1, p), P_3(s_2, m)\}\Big\}.$$

O algoritmo MinCycle calcula as mudanças minimais de um vértice v_r que ocorre em um ciclo do grafo de testemunhas. A ideia principal do algoritmo consiste em remover do ciclo os elementos que não geram mudanças minimais. Estes elementos quando removidos do grafo, desconectam o ciclo, e o algoritmo PartialRepair é utilizado para encontrar as mudanças minimais que definem estratégias de ganhar em v_r. Explicamos primeiro quais são estes elementos através de um exemplo. Tomemos o vértice v_1 do grafo de testemunhas da figura 6.7 que pertence a uma CFCM de maior ponto fixo $Q = \{v_1, v_3, v_4, v_6\}$. Abelardo pode jogar para não perder, a partir de v_1, por v_2 ou por v_3. Ao jogar por v_3, uma aresta do ciclo, ele pode jogar para v_4 onde tem duas opções: jogar para v_5 ou v_6. Se jogar por v_6, outra aresta do ciclo, Abelardo jogará infinitamente frequentemente por um ciclo de maior ponto fixo e, portanto, perderá o jogo. Logo, o vértice v_6 não pertence às estratégias de não perder de Abelardo a partir de v_1, e podemos descartá-lo do grafo de testemunhas ao buscarmos as mudanças minimais que definem estratégias de ganhar a Eva em v_1. A exclusão de v_6 remove o ciclo ao qual v_1 pertencia, e o algoritmo PartialRepair pode ser utilizado para encontrar as mudanças minimais em v_1. Chamamos estes vértices, que não estão ligados a uma estratégia de não perder de um vértice qualquer de uma CFCM de vértices pseudo-testemunhas.

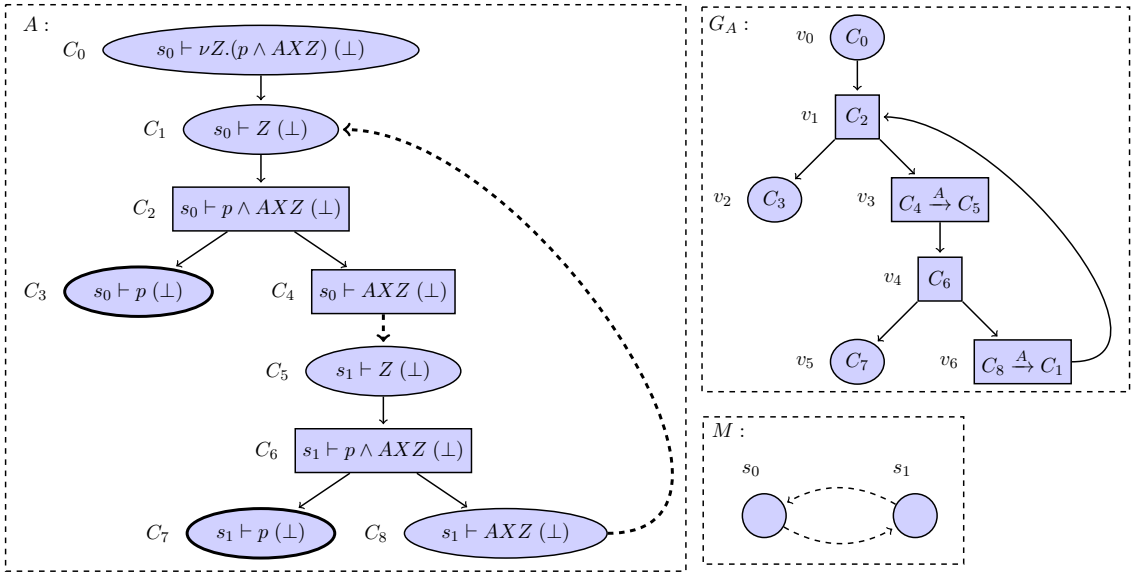

Figura 6.7: Arena A de um jogo de verificação de modelos com fórmula CTL $AG\ p$ escrita em μ-calculus ($\nu Z.(p \wedge AXZ)$) e modelo KMTS M, e o grafo de testemunhas G_A de A.

Definição 6.3.3. *Seja G um grafo de testemunhas, v_r um vértice de G e Q a CFCM a*

qual v_r pertence. Definimos os conjunto pre(G, v_r) e pseudo-W(G, v_r) como

$$\begin{aligned} \mathit{pre}(G, v_r) = \{v_x \in Q \mid \exists \pi = v_x \to v_r \in Q,\ &\text{e } v_x = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi) \\ &\text{ou } v_x = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)\} \\ \mathit{pseudo\text{-}W}(G, v_r) = \{v_x \in \mathit{pre}(G, v_r) \mid \exists \pi = v_x \to v_{x+1} \to \cdots \to v_{r-1} \to v_r \\ &\text{um caminho em } Q, \text{ e } \forall v_i \in \pi \setminus \{v_r\}, v_i = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi) \\ &\text{ou } v_i = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)\}, \end{aligned}$$

Para os vértices v_1 e v_4, temos que *pseudo-W*$(G_A, v_1) = \{v_6\}$ e *pseudo-W*$(G_A, v_4) = \{v_3\}$. Para encontrarmos as mudanças minimais de cada vértice de um ciclo, é suficiente removermos durante o processo os vértices pseudo-testemunhas de cada um deles. Ou seja, para encontrarmos as mudanças minimais em v_1 desconsideramos a mudança definida por v_6, e para o vértice v_4 excluímos a mudança definida por v_3. Desta forma, $\textsc{PartialRepair}(G_A, v_1) = \mathit{Change}(v_2) \uplus \textsc{PartialRepair}(G_A \setminus \mathit{pseudo} - W(G, v_1), v_3)$, onde $G_A \setminus \mathit{pseudo} - W(G_A, v_1)$ remove de G_A os vértices em $\mathit{pseudo} - W(G_A, v_1)$. Em nosso exemplo temos que o conjunto das mudanças minimais de v_1 é

$$\gamma = \mathit{Change}(v_2) \uplus \textsc{PartialRepair}(G_A \setminus \{v_6\}, v_3)$$

Visto que removemos o vértice v_6 do grafo de testemunhas G_A da figura 6.7, o único vértice filho de v_4 é o vértice v_5. Logo,

$$\begin{aligned} \textsc{PartialRepair}(G_A \setminus \{v_6\}, v_4) &= \{\mathit{Change}(v_5)\} \\ \textsc{PartialRepair}(G_A \setminus \{v_6\}, v_3) &= \{\mathit{Change}(v_3)\} \cup \textsc{PartialRepair}(G_A \setminus \{v_6\}, v_4) \\ &= \{\mathit{Change}(v_3), \mathit{Change}(v_5)\} \end{aligned}$$

Segue das equações acima que:

$$\begin{aligned} \gamma &= \mathit{Change}(v_2) \uplus \textsc{PartialRepair}(G \setminus \{v_6\}, v_3) \\ &= \mathit{Change}(v_2) \uplus \{\mathit{Change}(v_3), \mathit{Change}(v_5)\} \\ &= \{\mathit{Change}(v_2) \cup \mathit{Change}(v_3), \mathit{Change}(v_2) \cup \mathit{Change}(v_5)\} \\ &= \{\{P_3(s_0, p), P_1(s_0, s_1)\}, \{P_3(s_0, p), P_3(s_1, p)\}\} \end{aligned}$$

Obtivemos o conjunto das mudanças minimais de v_1. Enquanto Abelardo perde ao jogar infinitamente frequentemente por um ciclo de maior ponto fixo, Eva perde ao jogar por um ciclo de menor ponto fixo. Desta forma, devemos remover do grafo de testemunhas os vértices pseudo-testemunhas de um vértice v_r dos ciclos de menor ponto fixo em que Eva joga, para encontrar as mudanças minimais em v_r.

Lembramos que Abelardo joga por ciclos somente através das configurações da forma $s \vdash AX$ e Eva por configurações da forma $s \vdash EX$. Escrevemos $Q = (\eta, \Psi)$ para

indicar que a CFCM Q de um grafo de testemunhas G_A é caracterizada por um ponto fixo η (maior ou menor ponto fixo), e Ψ indica o jogador do ciclo (Eva ou Abelardo); onde $\eta \in \{\mu, \nu\}$ e $\Psi \in \{EX, AX\}$. Por exemplo, a componente $Q = \{v_1, v_3, v_4, v_6\}$ da figura 6.7 é uma componente de maior ponto fixo e da forma AX, portanto escrevemos $Q = (\nu, AX)$.

O algoritmo MinCycle é apresentado a seguir.

Algoritmo 3: MinCycle(G, v_r, Q, β)

Entrada: Um grafo G de testemunhas, um vértice v_r de G, e a CFC Q a qual v_r pertence

Saída: Conjunto de mudanças minimais que garantem a Eva estratégia de ganhar em v_r

```
1  if Q = (ν, AX) ou Q = (μ, EX) then
2  |  G' := G \ pseudo-W(G, v_r);
3  |  return MinRepair(G', v_r);
4  else
5  |  E' = ∅;
6  |  foreach v_i ∈ pre(G, v_r) do
7  |  |  if v_i → v_r ∈ G then
8  |  |  |  E' := E' ∪ {v_i → v_r};
9  |  |  end
10 |  end
11 |  G' := G \ E';
12 |  return MinRepair(G', v_r);
13 end
```

Dado um grafo de testemunhas G_A e um vértice v_r deste grafo, se v_r pertence a uma CFCM Q classificada como (ν, AX) ou (μ, EX) (linha 1 do algoritmo MinCycle), então devemos remover os vértices pseudo-testemunhas de G_A e calcular o conjunto de mudanças minimais de v_r através do algoritmo MinRepair. Para os demais casos, removemos as arestas entre os vértices de $pre(G_A, v_r)$ e v_r e calculamos as mudanças minimais de v_r sobre um grafo de testemunhas em que v_r não participa de nenhum ciclo (linhas 5 a 12).

Analisemos agora as CFCM da forma (ν, EX) e (μ, AX). Eva e Abelardo ganham ao jogar respectivamente por ciclos de maior e menor ponto fixo. Portanto, Eva ganha ao jogar por um ciclo em uma CFCM (ν, EX), e Abelardo ganha ao jogar por um ciclo em uma CFCM (μ, AX). Analisemos quais os elementos que devem ser excluídos de uma CFCM $Q = (\nu, EX)$ para encontrar as mudanças minimais de um vértice $v_r \in Q$. Q é uma CFCM de maior ponto fixo, portanto Eva ganha ao jogar por ciclos desta componente. Se Eva joga de um vértice v_r de uma CFCM $Q = (\nu, EX)$ até um vértice

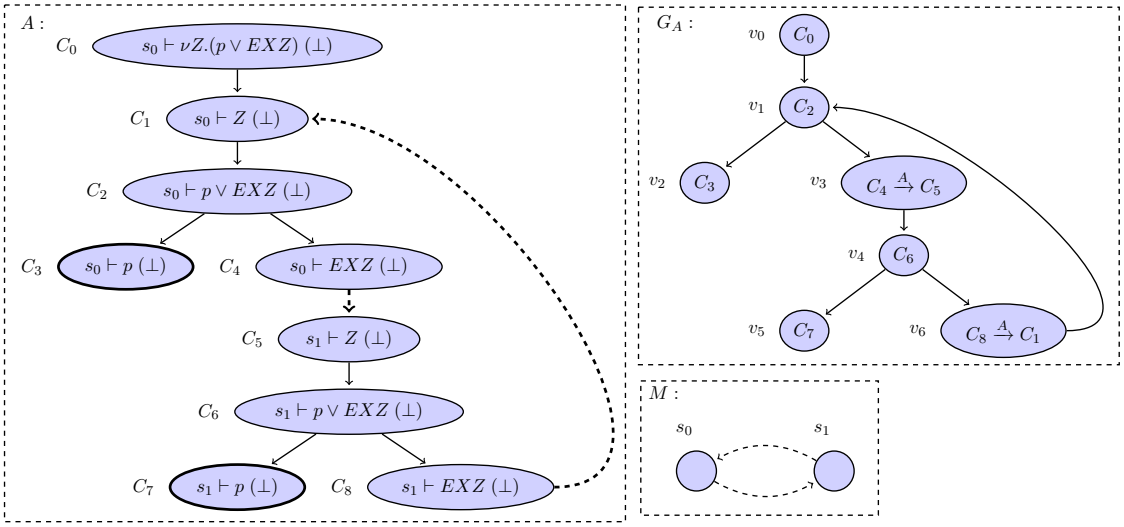

Figura 6.8: Arena A de um jogo de verificação de modelos com fórmula CTL $EF\ p$ escrita em μ-calculus ($\nu Z.(p \vee EXZ)$) e modelo KMTS M, e o grafo de testemunhas G_A de A.

v_i pelo grafo de testemunhas e $v_i \in pre(G_A, v_r)$, então a estratégia de não perder de Eva por este caminho consiste em jogar infinitamente frequentemente pelo ciclo de maior ponto fixo. Os vértives em $pre(G_A, v_r)$ representam as arestas do jogo por onde Eva joga infinitamente frequentemente por ciclos de maior ponto fixo para não perder. Logo, não precisamos das arestas $v_i \to v_r$ no grafo de testemunhas para representar estes ciclos, uma vez que estes ciclos já estão representados pelos vértices em $pre(G_A, v_r)$. Em outras palavras, é suficiente que removamos as arestas $v_i \to v_r$ em Q de G_A e apliquemos o algoritmo MinRepair($G_A \setminus pre(G_A, v_r)$) para encontrarmos as mudanças minimais que definem estratégia de ganhar a Eva em v_r. Lembramos que o grafo $G' = G_A \setminus pre(G_A, v_r)$ é um grafo em que v_r não pertence a nenhum ciclo, e por esta razão o algoritmo MinRepair(G', v_r) encontra as mudanças minimais de v_r. Tomemos, por exemplo, o vértice v_1 do grafo de testemunhas da figura 6.8 que pertence a uma componente de maior fixo por onde Eva joga. Para encontrarmos as mudanças minimais em v_1, calculamos as mudanças minimais de v_2 e v_3 e continuamos a análise de forma descendente. Observamos que Eva joga infinitamente frequentemente pelo ciclo $v_1 \to v_3 \to v_4 \to v_6 \to v_1$. Para ganhar em v_6 é suficiente que Eva jogue consistentemente (por transições *must*) por este ciclo, logo aplicamos apenas a mudança $Change(v_6)$ que transforma (s_1, s_0) em transição *must*. Se desconectarmos a aresta $v_6 \to v_1$ do grafo de testemunhas, teremos um grafo sem cíclos e o algoritmo MinCycle poderá ser aplicado para calcular as mudanças minimais em v_1 esta mudança para v_6 e as demais mudanças do ciclo e ao final encontrará as mudanças minimais de v_1. O raciocínio para as componentes da forma $Q = (\mu, AX)$ é análogo. Façamos $G' = G_A \setminus \{v_6 \to v_1\}$. Em nosso exemplo temos que o conjunto das mudanças minimais de v_1 é

$$\begin{aligned}\textsc{MinRepair}(G_A, v_1) &= \textsc{MinRepair}(G', v_1)\\ &= \{Change(v_2)\} \cup \textsc{PartialRepair}(G', v_3).\end{aligned}$$

Visto que removemos a aresta $v_6 \to v_1$ do grafo de testemunhas G_A da figura 6.7, v_6 torna-se um o vértice terminal em G'. Logo,

$$\begin{aligned}\textsc{PartialRepair}(G', v_6) &= \{Chage(v_6)\} = \{\ \{P_3(s_1, p)\}\ \}\\ \textsc{PartialRepair}(G', v_5) &= \{Chage(v_5)\} = \{\ \{P_2(s_1, s_0)\}\ \}\\ \textsc{PartialRepair}(G', v_4) &= \textsc{PartialRepair}(G', v_5) \cup \textsc{PartialRepair}(G', v_6)\\ \textsc{PartialRepair}(G', v_3) &= \{Change(G', v_3)\} \uplus \textsc{PartialRepair}(G', v_4)\\ &= \{\ \{P_2(s_0, s_1)\}\ \} \uplus \{\ \{P_3(s_1, p)\},\ \{P_2(s_1, s_0)\}\ \}\\ &= \{\ \{P_2(s_0, s_1), P_3(s_1, p)\},\ \{P_2(s_0, s_1), P_2(s_1, s_0)\}\ \}\end{aligned}$$

Segue das equações acima que:

$$\begin{aligned}\textsc{MinRepair}(G_A, v_1) &= \{Change(v_2)\} \cup \textsc{PartialRepair}(G', v_3)\\ &= Change(v_2) \uplus \{Change(v_3), Change(v_5)\}\\ &= \{Change(v_2) \cup Change(v_3), Change(v_2) \cup Change(v_5)\}\\ &= \{\ \{P_3(s_0, p)\},\ \{P_2(s_0, s_1), P_3(s_1, p)\},\ \{P_2(s_0, s_1), P_2(s_1, s_0)\}\ \}\end{aligned}$$

6.3.1 Correção do Algoritmo

Apresentamos nesta seção a prova de que o Algoritmo MinRepair retorna o conjunto das mudanças minimais que definem estratégias de ganhar a Eva em um vértice de um grafo de testemunhas.

Teorema 6.3.1. *Seja G_A o grafo de testemunhas de uma arena A, e v_r um vértice de G_A. O algoritmo* $\textsc{MinRepair}(G_A, v_r)$ *retorna o conjunto minimal de mudanças que definem estratégias de ganhar a Eva em v_r, se $v_r \neq (C_i \xrightarrow{A} C_j)$ ou* $\textsc{Parent}(v_r) \neq (C_i)$*; onde $C_i = (s \vdash AX\varphi)$ ou $C_i = (s \vdash EX\varphi)$, e $C_j = (s' \vdash \varphi)$.*

Prova. O algoritmo MinRepair remove do grafo de testemunhas todos os ciclos presentes, e transforma o grafo de testemunhas em uma árvore. Portanto, podemos aplicar uma indução estrutural sobre a árvore gerada. Faremos a prova por indução na estrutura da árvore.

- **Base**: um vértice v_r terminal de G_A.

1. $v_r = (s \vdash l)$ (linha 2): segue da linha 3 do algoritmo PartialRepair que o conjunto de mudanças de v_r é $\{\ \{P_3(s, l)\}\ \}$. $P_3(s, l)$ define o literal l ao estado s do modelo e recolore a configuração $s \vdash l$ com $\top$. Logo, $\textsc{PartialRepair}(G, v_r)$ define

estratégia de ganhar a Eva em v_r. Visto que não existe outra mudança menor que esta que defina estratégia de ganhar a Eva em $s \vdash l$, o conjunto $\{ \{P_3(s,l)\} \}$ é minimal;

2. $v_r = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)$ e $\textsc{Parent}(v_r) \neq s \vdash EX\varphi$ (linha 4): Como v_r é um vértice terminal, tem-se que $s' \vdash \varphi$ está colorido com $\top$ e (s, s') é uma transição *may*.

   Logo, a simples transformação de (s, s') em transição *must* garante a Eva estratégia de ganhar em v_r. Em outras palavras, a mudança $\{P_2(s, s')\}$ define estratégia de ganhar a Eva em v_r e também é minimal. Da linha 5 do algoritmo, temos que o conjunto de mudanças de v_r é $\{Change(v_r)\}$, e de acordo com a definição 6.1.1 temos que $Change(v_r) = \{P_1(s, s')\}$. Logo, $\textsc{MinRepair}(G_A, v_r)$ é minimal.

3. $v_r = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi)$ e $\textsc{Parent}(v_r) \neq s \vdash AX\varphi$ (linha 16): Como v_r é um vértice terminal, tem-se que $s' \vdash \varphi$ está colorido com F e (s, s') é uma transição *may*. Logo, a simples remorção da transição *may* genuína (s, s') remove a estratégia de não perder de Abelardo em v_r. Visto que $\textsc{Parent}(v_r) \neq s \vdash AX\varphi$, a única estratégia de não perder de Abelardo em v_r é jogar para a configuração $s' \vdash \varphi$. Portanto, $\{P_1(s, s')\}$ remove a única estratégia de não perder de Abelardo em v_r e garante estratégia de ganhar a Eva em v_r. Em outras palavras, a mudança $\{P_1(s, s')\}$ define estratégia de ganhar a Eva em v_r e também é minimal. Da linha 5 do algoritmo, temos que o conjunto de mudanças de v_r é $\{Change(v_r)\}$, e de acordo com a definição 6.1.1 temos que $Change(v_r) = \{P_1(s, s')\}$. Logo, $\textsc{MinRepair}(G_A, v_r)$ é minimal.

- **Passo da Indução**: suponhamos que o algoritmo $\textsc{MinRepair}(G_A, v_i)$ retorna as mudanças minimais de qualquer vértice v_i descendente de um vértice v_r, se $v_r \neq (C_i \xrightarrow{A} C_j)$ ou $\textsc{Parent}(v_r) \neq (C_i)$; onde $C_i = (s \vdash AX\varphi)$ e $C_j = (s' \vdash \varphi)$, para uma fórmula CTL φ não atômica. Provaremos que $\textsc{MinRepair}(G_A, v_r)$ retorna o conjunto das mudanças minimais que definem estratégias de ganhar a Eva em v_r de G_A.

  1. Suponhamos que v_r não pertença a nenhum ciclo em G_A. Portanto, o algoritmo $\textsc{PartialRepair}(G, v_r)$ retorna as mudanças de v_r.

     (a) $v_r = (s \vdash \varphi \vee \psi)$ (linha 6). O vértice v_r é um menor ancestral comum, e para definir a Eva estratégia de ganhar em v_r é suficiente garantir que Eva ganhe a partir de um dos seus vértices filhos. Da linha 7, temos que

     $$conjResult = \bigcup_{v_i \in \textsc{Children}(G_A, v_i)} \textsc{MinRepair}(G_A, v_i). \tag{6.1}$$

     Por hipótese, para qualquer vértice filho v_i de v_r, $\textsc{MinRepair}(G_A, v_i)$ retorna o conjunto de mudanças minimais que definem estratégias de ganhar a Eva em v_i. Portanto, todas as mudanças em $conjResult$ definem

estratégias de ganhar a Eva em v_r. Se existirem mudanças $X_j, X_w \in conjResult$ tais que $X_j \subset X_w$ é suficiente que removamos X_w (linha 19). Logo, as mudanças em MinRepair(G_A, v_r) definem estratégias de ganhar a Eva em v_r e são minimais.

(b) $v_r = (s \vdash EX\varphi)$. O vértice v_r é um menor ancestral comum, e para definir a Eva estratégia de ganhar em v_r é suficiente garantir que Eva ganhe a partir de um dos seus vértices filhos. Particionemos o conjunto dos vértices filhos de v_r em dois conjuntos: $\gamma = \{v_i \in \text{Children}(v_r) \mid v_i = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)\}$ (linha 9) e $\omega = \text{Children}(v_r) \setminus \gamma$ (linha 10). Para definir uma estratégia de ganhar a Eva em um vértice $v_i = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi) \in \gamma$, devemos transformar transição *may* (s, s') de transição *must* (aplicar mudança $\{P_2(s, s')\}$) e definir uma estratégia de ganhar a Eva no nó vértice filho de v_i. De acordo com a definição 6.1.1 tem-se que $Change(v_r) = \{P_1(s, s')\}$. Por hipótese, temos que $\forall v' \in \text{Children}(v_i)$, as mudanças em MinRepair(G_A, v') definem estratégias de ganhar a Eva em v', visto que $\text{Parent}(v') = v_i \neq (s \vdash AX\varphi)$. Logo, as mudanças em $Change(v_i) \cup$ MinRepair(G_A, v') remove a estratégia de não perder de Abelardo ao jogar por v_i. Desta forma, cada mudança em

$$\beta = \bigcup_{v_i \in \gamma} \{Change(v_i)\} \uplus \text{MinRepair}(G, v_i)$$

garante a Eva estratégia de ganhar ao jogar por qualquer vértice em γ, isto é, estratégia de ganhar em v_r. Por hipótese, as mudanças em MinRepair(G_A, v'), para qualquer $v' \in \omega$, definem estratégias de ganhar a Eva em v' e são minimais. Portanto, cada mudança em

$$conjResult = \beta \cup \bigcup_{v_i \in \omega} \text{MinRepair}(G, v_i)$$

remove as estratégias de não perder de Abelardo para qualquer vértice filho de v_r, isto é, define uma estratégia de ganhar a Eva em v_r. Se existirem mudanças $X_j, X_w \in conjResult$ tais que $X_j \subset X_w$, é suficiente remover X_w de $conjResult$, para garantir a minimalidade das mudanças. A linha 22 do algoritmo define o conjunto $conjResulta$ acima, e a linha 24 filtra de $conjResult$ as mudanças minimais. Concluímos, portanto, que MinRepair(G_A, v_r) encontra as mudanças minimais que definem estratégias de ganhar a Eva em v_r.

(c) $v_r = (s \vdash \varphi \wedge \psi)$ (linha 14). Para definirmos estratégias de ganhar a Eva em v_r temos que remover as estratégias de não perder de Abelardo em seus dois vértices filhos v_1 e v_2. Por hipótese, para todos os vértices filhos v_i de v_r, MinRepair(G_A, v_i) retorna as mudanças minimais que definem estratégias de ganhar a Eva em v_i. Portanto, $conjResult$ tem as mudanças que definem estratégias de ganhar a Eva em v_r. Se existirem mudanças

$X_j, X_w \in conjResult$ tais que $X_j \subset X_w$, é suficiente que removamos X_w de $conjResult$ A linha 24 faz este filtro e garante a minimalidade das mudanças. Logo, as mudanças em $\text{MinRepair}(G_A, v_r)$ definem estratégias de ganhar a Eva em v_r e são minimais.

(d) $v_r = (s \vdash EX\varphi \xrightarrow{A} s' \vdash \varphi)$ e $\text{Parent}(v_r) \neq (s \vdash EX\varphi)$. Logo, a única estratégia de não perder de Eva em $s \vdash EX\varphi$ é jogar para $s' \vdash \varphi$. Por hipótese, $\text{PartialRepair}(G_A, \text{Child}(v_r))$ retorna as mudanças minimais que definem estratégias de ganhar a Eva em $\text{Child}(v_r)$. Para ganhar em v_r, Eva deve jogar por uma aresta *must* para um configuração v_i que tenha estratégia de ganhar. A combinação da mudança $Change(v_r) = \{P_2(s, s')\}$, que transforma a transição *may* genuína (s, s') em *must*, garante a Eva estratégia de ganhar em v_r. Em outras palavras, as mudanças em $\{change(v_r)\} \uplus \text{PartialRepair}(G_A, \text{Child}(v_r))$ ($\text{PartialRepair}(G_A, v_r)$, linha 5) garantem a Eva estratégia de ganhar em v_r. Visto que a única estratégia de não perder de Eva em $s \vdash EX\varphi$ é jogar para $s' \vdash \varphi$, tem-se que o filtro sobre $change(v_r) \uplus \text{PartialRepair}(G_A, \text{Child}(v_r))$ retorna as mudanças minimais (linha 24), ou seja, $\text{PartialRepair}(G_A, v_r)$ é minimal e garannte a Eva estratégia de ganhar em v_r.

(e) $v_r = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi)$ e $\text{Parent}(v_r) \neq (s \vdash AX\varphi)$. Logo, a única estratégia de não perder de Eva em $s \vdash EX\varphi$ é jogar para $s' \vdash \varphi$. Por hipótese, $\text{PartialRepair}(G_A, \text{Child}(v_r))$ retorna as mudanças minimais que definem estratégias de ganhar a Eva em $\text{Child}(v_r)$. Para ganhar em v_r, Eva deve jogar por uma aresta *must* para um configuração v_i que tenha estratégia de ganhar. A mudança $Change(v_r) = \{P_1(s, s')\}$, que remove a transição *may* genuína (s, s'), garante a Eva estratégia de ganhar em v_r. Além disso, a aplicação das mudanças que definem estratégia de ganhar a Eva no vértice filho de v_r, remove a única estratégia de não perder de Abelardo em v_r. Logo, as mudanças em $\{change(v_r)\} \cup \text{PartialRepair}(G_A, \text{Child}(v_r))$ ($\text{PartialRepair}(G_A, v_r)$, linha 17) garantem a Eva estratégia de ganhar em v_r. Visto que a única estratégia de não perder de Eva em $s \vdash EX\varphi$ é jogar para $s' \vdash \varphi$, tem-se que o filtro aplicado sobre $change(v_r) \uplus \text{PartialRepair}G_A, \text{Child}(v_r)$ retorna as mudanças minimais (linha 24), ou seja, $\text{PartialRepair}(G_A, v_r)$ é minimal e garannte a Eva estratégia de ganhar em v_r.

(f) $v_r = (s \vdash AX\varphi)$ (linha 18). Para definir uma estratégia de ganhar a Eva em v_r, devemos remover todas as estratégias de não perder de Abelardo em v_r. Particionemos o conjunto dos vértices filhos de v_r em dois conjuntos: $\gamma = \{v_i \in \text{Children}(v_r) \mid v_i = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi)\}$ e $\omega = \{v_i \in \text{Children}(v_r) \mid v_i \notin \gamma\}$.
Temos duas formas de remover uma estratégia de não perder de Abelardo para cada vértice $v_i = (s \vdash AX\varphi \xrightarrow{A} s' \vdash \varphi) \in \gamma$: definir uma estratégia de ganhar a Eva no vértice filho de v_i ou remover a transição *may* (s, s')

do modelo. Por hipótese, MinRepair retorna as mudanças minimais que definem estratégias de ganhar a Eva a qualquer vértice v' descendente de v_r, se v' não for da forma $(s \vdash AX\psi \xrightarrow{A} s' \vdash \psi)$ e $\text{Parent}(v')$ for diferente todos os descendentes de v_r. Por hipótese, temos que $\forall v' \in \text{Children}(v_i)$, as mudanças em $\text{MinRepair}(G_A, v')$ definem estratégias de ganhar a Eva em v', visto que $\text{Parent}(v') = v_i \neq (s \vdash AX\varphi)$. Logo, as mudanças em $Z_i = Change(v_i) \cup \text{MinRepair}(G_A, v')$ remove a estratégia de não perder de Abelardo ao jogar por v_i. Desta forma, cada mudança em

$$\beta = \bigcup_{v_i \in \gamma} \{Change(v_i)\} \cup \text{MinRepair}(G, v_i)$$

remove as estratégias de não perder de Abelardo ao jogar por qualquer vértice em γ. Por hipótese, as mudanças em $\text{MinRepair}(G_A, v')$, para qualquer $v' \in \omega$, definem estratégias de ganhar a Eva em v' e são minimais. Portanto, cada mudança em

$$conjResult = \beta \cup \biguplus_{v_i \in \omega} \text{MinRepair}(G, v_i)$$

remove as estratégias de não perder de Abelardo para qualquer vértice filho de v_r, isto é, define uma estratégia de ganhar a Eva em v_r. Se existirem mudanças $X_j, X_w \in conjResult$ tais que $X_j \subset X_w$, é suficiente remover X_w de $conjResult$, para garantir a minimalidade das mudanças. A linha 22 do algoritmo define o conjunto $conjResulta$ acima, e a linha 24 filtra de $conjResult$ as mudanças minimais. Concluímos, portanto, que $\text{MinRepair}(G_A, v_r)$ encontra as mudanças minimais que definem estratégias de ganhar a Eva em v_r.

2. Suponhamos que v_r pertença a um ciclo de G_A. Logo, as mudanças em v_r serão calculadas pelo algoritmo MinCycle. Seja Q a CFC à qual v_r pertence, vamos provar que MiNCycle retorna o conjunto das mudanças minimais que definem estratégia de ganhar a Eva em v_r.
   (a) $Q = (\nu, AX)$. O conjunto das mudanças de v_r é calculado sobre o grafo $G' = G_A \setminus \text{pseudo-W}(G_A, v_r)$ (eliminação dos vértices pseudo testemunhas). Em G', v_r não pertence mais a um ciclo do grafo, visto que as arestas que partiam dos vértices em $\text{pseudo-w}(G_A, v_r)$ fechavam os ciclos aos quais v_r pertenciam. O cálculo das mudanças de v_r é realizado, portanto, por $\text{PartialRepair}(G', v_r)$ (volta na recursão) que como já provamos retorna as mudanças minimais que definem estratégias de ganhar a Eva em v_r.
   (b) $Q = (\mu, EX)$. Análogo a (ν, AX).
   (c) $Q = (\mu, AX)$. O conjunto das mudanças de v_r é dado pelo grafo $G' = G \setminus E'$ (linha 11); onde E' é o conjunto das arestas entre os vértices em

PRE(G, v_r) e o vértice v_r. Logo, o vértice v_r no grafo G' não pertence a nenhum ciclo do grafo (as arestas que envolviam v_r em um ciclo foram removidas). O conjunto das mudanças em v_r será calculado pelo algoritmo PartialRepair sobre G' que, como já provamos, retorna as mudanças minimais que definnem estratégia de ganhar a Eva em v_r.

(d) $Q = (\nu, EX)$. Análogo a $Q = (\mu, AX)$.

Concluímos portanto que MinRepair calcula todas as mudanças minimais que garantem a Eva estratégia de ganhar em v_r.

∎

6.3.2 Complexidade

Nesta seção, discutimos sobre a complexidade computacional do problema do refinamento. De fato, mostramos que o refinamento é NP-Difícil.

Teorema 6.3.2. *O refinamento de modelos KMTS é NP-Difícil.*

Prova. Reduzimos SAT ao problema do refinamento.

Toda fórmula da lógica proposicional é uma fórmula CTL bem formada, portanto não precisamos fazer nenhuma modificação sobre φ. Devemos, construir um modelo M de forma que os modelos expandidos de M correspondam às valorações de φ. Construamos um KMTS $M = (AP, S, R^+, R^-, L)$, onde AP é o conjunto dos símbolos proposicionais da fórmula φ, $S = \{s\}$ com um único estado, $L(s) = \emptyset$ (todos os literais estão indefinidos em s) e $R^- = R^+ = \emptyset$. Desta forma, a cada estrutura de Kripke $k_i = (AP, S, R_i, L_i) \in K(M)$, o conjunto $L_i(s)$ corresponde a uma conjunto de literais que pode ser tomado com um valor verdade sobre φ. Logo, φ é satisfazível sse $\exists k \in K(M);\ k, s \models \varphi$, e φ é insatisfazível caso contrário. Seja $Ref(M, s, \varphi)$ o refinamento de M em ralação a φ a partir de s, segue que,

$$\varphi \text{ é insatisfazível sse } Ref(M, s, \varphi) = \emptyset$$
$$\varphi \text{ é satisfazível sse } Ref(M, s, \varphi) \neq \emptyset$$

Logo, o resultado do refinamento de modelos nos informa se φ é ou não satisfazível através desta redução, ou seja, resolve SAT. Concluímos, portanto, que o refinamento de modelos KMTS é NP-Difícil.

∎

6.3.3 Discussão

Ao utilizarmos o grafo de testemunhas ficamos apenas com as informações necessárias da arena de um jogo para encontrarmos as mudanças minimais: as testemunhas de falhas e os menores ancestrais comuns. Enquanto as testemunhas de falhas fornecem as mudanças necessárias para definir as estratégias de ganhar a Eva ao longo do jogo, os menores ancestrais comuns informam quais as mudanças que devem ou não ser combinadas, bem

como as condições para encontrar as mudanças minimais a cada vértice do grafo de testemunhas.

Os algoritmos propostos nesta dissertação utilizam a abordagem de divisão e conquista para encontrar as mudanças minimais dos vértices descendentes de um nó e realiza um filtro para selecionar as mudanças minimais no respectivo nó. Esta abordagem permite que a cada vértice o número de combinações entre mudanças, bem como a seleção das mudanças minimais seja reduzida, visto que a cada vértice descendente as mudanças não minimais são descartadas. Outra vantagem do grafo de testemunhas é o fato de não precisarmos aplicar uma nova verificação de modelos a cada mudança aplicada sobre o modelo, para verificar se o modelo modificado atende à especificação. De fato, o grafo de testemunhas informa as mudanças necessárias que devem ser aplicadas para definir estratégias de ganhar a Eva a cada vértice do grafo.

Como trabalhos futuros, pretendemos fazer uma análise de complexidade de caso médio para os algoritmos propostos neste capítulo, bem como analisar e propor heurísticas para a otimização do refinamento através do grafo de testemunhas. Pretendemos também analisar o percentual de redução realizado pelo grafo de testemunhas ao abstrair a arena de um jogo. Os algoritmos propostos neste trabalho foram implementados e deram origem ao verificador e refinador de modelos ARIIS (RIBEIRO et al., 2015), e pretendemos fazer uma análise experimental sobre esta implementação para verificar o seu desempenho.

Capítulo 7

CONCLUSÃO

Neste trabalho, apresentamos uma solução para o refinamento de modelos KMTSs através de um verificador de modelos para estas estruturas interpretadas como conjunto de estruturas de Kripke; e de algoritmos que retornam uma solução completa do refinamento.

7.1 VERIFICADOR DE MODELOS

Na verificação de uma propriedade CTL sobre KMTS interpretado como conjunto de modelos, o valor verdade indefinido não é composicional sobre a conjunção e a disjunção. Contudo, as abordagens de verificação de modelos para KMTS com lógicas de três valores existentes consideram a lógica de três valores de Kleene (KLEENE et al., 1952) ou uma interpretação equivalente as quais são composicionais sobre a conjunção e disjunção de fórmulas. Portanto, não podemos utilizar estes verificadores de modelos em nosso trabalho, visto que não atendem à interpretação adotada.

Nesse sentido, propomos um verificador de modelos baseado em jogos para um modelo KMTS interpretado como um conjunto de estruturas de Kripke. Para tal, representamos o valor verdade de um fórmula CTL, no processo de verificação, como um conjunto de KMTSs. Nesta interpretação, definimos uma operação de contração que calcula o valor verdade (conjunto de KMTSs) da composição entre duas fórmulas φ e ψ, tal que uma função de coloração apropriada é definida sobre a arena do jogo.

A motivação inicial para a definição deste verificador de modelos foi dar suporte ao refinamento de modelos KMTS ao qual este trabalho está inserido. Em (GUERRA; ANDRADE; WASSERMANN, 2013), os autores utilizam a verificação de modelos com lógica de 3 valores proposta por Grumberg, a qual não atende à semântica adotada e fornece resultados incompletos para o refinamento, devido à diferença na semântica adota sobre os KMTSs. Nosso verificador de modelos soluciona este problema e fornece as informações necessárias para um refinamento completo de um modelo KMTS. Acreditamos também que a nossa proposta de verificação de modelos, através das operações de contrações, possam ser utilizadas em outros contextos e ajustadas para lidar com outros lógicas além de CTL.

7.2 REFINAMENTO DE MODELOS KMTS

Apresentamos uma solução para o refinamento de um modelo KMTS, interpretado como um conjunto de estruturas de Kripke, através do grafo de testemunhas que abstrai a arena de um jogo de verificação de modelos e captura as informações necessárias para encontrar as mudanças minimais de forma eficaz. Além disso, provamos a correção da nossa solução e demonstramos que o refinamento é NP-Difícil. Acreditamos também que o grafo de testemunhas que propomos nesse trabalho fornece as informações necessárias para o processo de revisão de modelos KMTS e que possa ser utilizado para reparar um modelo KMTS quando este falhar em atender a uma especificação, isto, é quando o resultado da verificação for falso. Além disso, acreditamos que a nossa proposta pode ser utilizada em outros contextos e adaptada para lidar com outras lógicas além de CTL, como a LTL, por exemplo.

7.3 CONTRIBUIÇÕES

As contribuições deste trabalho são as seguintes:

- definição de uma estrutura, a qual chamamos de grafo de testemunhas, que captura as informações essenciais para o processo de refinamento de modelos KMTS;
- algoritmos para o refinamento que retornam os modelos gerados por mudanças minimais;
- definição de uma semântica para a lógica CTL cujos modelos são KMTSs interpretados como conjuntos de estruturas de Kripke;
- verificador de modelos combinado com jogos para KMTS interpretado como um conjunto de estruturas de Kripke;
- prova de que o refinamento admite solução única;
- prova da complexidade do problema de refinamento: NP-Difícil;
- prova da complexidade da verificação de modelos KMTS interpretado como um conjunto de estruturas de Kripke: NP-Completo;
- implementação de uma solução para o refinamento de um KMTS interpretado como conjunto de estruturas de Kripke.

7.4 TRABALHOS FUTUROS

Como trabalhos futuros, pretendemos fazer uma análise de complexidade de caso médio para os algoritmos propostos neste trabalho, bem como analisar e propor heurísticas para a otimização do refinamento através do grafo de testemunhas. Pretendemos também analisar o percentual de redução realizado pelo grafo de testemunhas ao abstrair a arena de um jogo. Os algoritmos propostos neste trabalho foram implementados e deram origem

ao verificador e refinador de modelos ARIIS (RIBEIRO et al., 2015), e pretendemos fazer uma análise experimental sobre esta implementação para verificar o desempenho desta em função do tempo. Pretendemos também implementar o verificador de modelos por contração proposto neste trabalho e realizar uma análise de complexidade de tempo para o caso médio, bem como definir heurísticas para otimizar a execução destes algoritmos. Estamos investigando algoritmos em tempo polinomial para o caso médio e temos a intenção de prover algoritmos eficientes de verificação de modelos para a semântica de modelos KMTSs interpretados como conjunto de estruturas de Kripke.

APÊNDICE A

PROVAS DO CAPÍTULO 4

Lemma 4.3.1. Seja Γ um conjunto não vazio de mudanças, $\forall X \in \Gamma\ \forall Y \in Comp(\Gamma)$ $X \not\simeq Y$.

Prova. Segue por indução.

Base: $\Gamma = \{X_0\}$. Por definição, $Comp(\{X_0\}) = \overline{X_0}$. Segue direto da definição 4.3.1 que $\forall Y \in \overline{X_0}, Y \not\simeq X_0$.

Passo da indução: suponhamos $\Gamma = \{X_0, \ldots, X_n, X_{n+1}\}$ e que $\forall X \in \Gamma, \forall Y \in Comp(\{X_0, \ldots, X_n\})$, $Y \not\simeq X$ para qualquer $X \in \{X_0, \ldots, X_n\}$.

Façamos $\gamma = \{X_0, \ldots, X_n\}$ e particionemos $Comp(\gamma)$ em dois conjuntos:

$$\beta_0 = \{X \in Comp(\gamma) \mid X \simeq X_{n+1}\}$$
$$\beta_1 = \{X \in Comp(\gamma) \mid X \not\simeq X_{n+1}\}$$

$Comp(\gamma) = \beta_0 \cup \beta_1$.

Por hipótese, $\forall X \in \gamma,\ \forall Y \in Comp(\gamma),\ X \not\simeq Y$. Portanto,

$$\forall X \in \gamma, \forall Y \in \beta_0, X \not\simeq Y \tag{A.1}$$
$$\forall X \in \gamma, \forall Y \in \beta_1, X \not\simeq Y \tag{A.2}$$

Dada uma operação primitiva ϱ qualquer, segue de (A.1) que

$$\forall X \in \gamma, \forall Y \in \beta_0, Y \cup \{\varrho\} \not\simeq X \tag{A.3}$$

Por construção, $\forall Y \in \beta_0,\ Y \simeq X_{n+1}$, logo

$$\forall Y \in \beta_0, Y - X_{n+1} = \{Y \cup \{\neg\varrho\} \mid \varrho \in X_{n+1} \text{ e } \varrho \notin Y\} \tag{A.4}$$

que implica em

$$\forall X \in \gamma,\ \forall Y \in \beta_0, \forall Z \in\ (Y - X_{n+1}), Z \not\simeq X_{n+1} \tag{A.5}$$

De (A.4) e de (A.3), tem-se que

$$\forall X \in \gamma, \forall Y \in \beta_0, \forall Z \in (Y - X_{n+1}), Z \not\supset X. \tag{A.6}$$

Da expressão acima e de (A.5), segue que

$$\forall X \in \gamma,\ \forall Y \in \left(\bigcup_{X_i \in\ \beta_0} X_i - X_{n+1} \right)\ Y \not\supset X_{n+1} \text{ e } Y \not\supset X; \tag{A.7}$$

ou seja,

$$\forall X \in \gamma,\ \forall Y \in (\beta_0 \backslash\backslash X_{n+1}), Y \not\supset X_{n+1} \text{ e } Y \not\supset X. \tag{A.8}$$

Por construção,$\forall Y \in \beta_1, Y \not\supset X_{n+1}$. Logo, por definição $\forall Y \in \beta_1, Y - X_{n+1} = \{Y\}$ que implica em

$$\left(\bigcup_{X_i \in\ \beta_1} X_i - X_{n+1} \right) = \bigcup_{X_i \in\ \beta_1} \{X_i\} = \beta_1; \tag{A.9}$$

Portanto,

$$\beta_1 \backslash\backslash X_{n+1} = \beta_1. \tag{A.10}$$

Por definição,

$$\begin{aligned}
Comp(\Gamma) &= Comp(\{X_0, \ldots, X_n, X_{n+1}\}) && \text{(A.11)}\\
&= Comp(\{X_0, \ldots, X_n\}) \backslash\backslash X_{n+1} && \text{(A.12)}\\
&= Comp(\gamma) \backslash\backslash X_{n+1} && \text{(A.13)}\\
&= \bigcup_{X_i \in\ Comp(\gamma)} X_i - X_{n+1} && \text{(A.14)}\\
&= \bigcup_{X_i \in \beta_0 \cup \beta_1} X_i - X_{n+1} && \text{(A.15)}\\
&= \left(\bigcup_{X_i \in \beta_0} X_i - X_{n+1} \right) \cup \left(\bigcup_{X_i \in \beta_1} X_i - X_{n+1} \right) && \text{(A.16)}\\
&= (\beta_0 \backslash\backslash X_{n+1}) \cup (\beta_1 \backslash\backslash X_{n+1}) && \text{(A.17)}\\
&= (\beta_0 \backslash\backslash X_{n+1}) \cup \beta_1. \qquad \text{(de A.10)} && \text{(A.18)}
\end{aligned}$$

Da última equação acima e de (A.8), segue que

$$\forall X \in \gamma, \forall Y \in (\beta_0 \setminus\setminus X_{n+1}) \cup \beta_1,\ Y \not\simeq X_{n+1} \text{ e } Y \not\simeq X.$$

Logo, $\forall X \in \gamma \cup \{X_{n+1}\}, \forall Y \in Comp(\Gamma),\ Y \not\simeq X$, isto é,

$$\forall X \in \Gamma,\ \forall Y \in Comp(\Gamma), Y \not\simeq X.$$

∎

Lemma 4.3.2. Seja Γ um conjunto não vazio de mudanças e Y uma mudança qualquer. Se $\forall X \in \Gamma$, $X \cap Y = \emptyset$ e $Y \simeq X$; então $\forall X \in Comp(\Gamma), X \simeq Y$.

Prova. Segue por indução.

Base: $\Gamma = \{X_0\}$. Suponhamos que $X_0 = \{\varrho_0, \ldots, \varrho_w\}$. Por definição, $Comp(\{X_0\}) = \{\{\neg\varrho_0\}, \ldots, \{\neg\varrho_w\}\}$.
Por hipótese, $X_0 \simeq Y$ e $X_0 \cap Y = \emptyset$, isto é,

$$\forall \varrho \in X_0,\ \varrho \notin Y \text{ e } \neg\varrho \notin Y.$$

Desta forma, $\forall \varrho \in X_0, \{\neg\varrho\} \simeq Y$. Visto que $Comp(\{X_0\}) = \{\overline{X_0}\}$ e $\overline{X_0} = \{\neg\varrho\}$, tem-se que $\overline{X_0} \simeq Y$ e, portanto, $\forall X \in Comp(\{X_0\}), X \simeq Y$.

Passo da Indução: suponhamos $\Gamma = \{X_0, \ldots, X_n, X_{n+1}\}$ e que $\forall X \in Comp(\{X_0, \ldots, X_n\})$, $X \simeq Y$. Vamos provar que $\forall X \in Comp(\{X_0, \ldots, X_n, X_{n+1}\})$, $X \simeq Y$.
Façamos $\gamma = \{X_0, \ldots, X_n\}$ e particionemos $Comp(\gamma)$ em dois conjuntos:

$$\beta_0 = \{X \in Comp(\gamma) \mid X \simeq X_{n+1}\}$$
$$\beta_1 = \{X \in Comp(\gamma) \mid X \not\simeq X_{n+1}\}$$

$Comp(\gamma) = \beta_0 \cup \beta_1$.
Por hipótese,$\forall Z \in \Gamma, Z \simeq Y$ e $Z \cap Y = \emptyset$. Logo,

$$\forall Z \in \Gamma, \forall \varrho \in Z, \varrho \notin Y \text{ e } \neg\varrho \notin Y. \tag{A.19}$$

Pois, como $Z \simeq Y$, não existe operação em Z que seja complementar a alguma em Y, ou seja, $\forall \varrho \in Z, \neg\varrho \notin Y$. Além disso, $\forall \varrho \in Z, \varrho \notin Y$, uma vez que $Z \cap Y = \emptyset$.
Como $X_{n+1} \in \Gamma$, segue da expressão anterior que

$$\forall \varrho \in X_{n+1}, \varrho \notin Y \text{ e } \neg\varrho \notin Y. \tag{A.20}$$

Pela hipótese da indução, $\forall Z \in Comp(\gamma)$, $Z \simeq Y$. Logo,

$$\forall Z \in \Gamma, \forall \varrho \in Z,\ \neg\varrho \notin Y. \tag{A.21}$$

Visto que $\forall Z \in \beta_0, Z \simeq X_{n+1}$, temos da definição 4.3.1 que

$$\forall Z \in \beta_0, (Z - X_{n+1}) = \{Z \cup \{\neg\varrho\} \mid \varrho \in X_{n+1} \text{ e } \varrho \notin Z\} \tag{A.22}$$

Tomemos uma mudança Z_i qualquer em β_0, logo

$$\forall W \in (Z_i - X_{n+1}), \forall \varrho \in W, \varrho \in Z_i \text{ ou } \neg\varrho \in X_{n+1}. \tag{A.23}$$

(I) Se $\varrho \in Z_i$, então $\{\varrho\} \simeq Y$. Pois, $Z_i \in \beta_0, \beta_0 \subseteq \gamma$ e, por hipótese, $\forall X \in \gamma, X \simeq Y$.

(II) $\neg\varrho \in X_{n+1}$. Temos por hipótese que a mudança $X_{n+1} \simeq Y$ e $X_{n+1} \cap Y = \emptyset$, visto que $X_{n+1} \in \Gamma$. Logo, de (A.19) tem-se que

$$\forall \varrho \in X_{n+1}, \varrho \notin Y \text{ e } \neg\varrho \notin Y.$$

Segue de (II) e da expressão acima que $\{\varrho\} \simeq Y$ e $\{\neg\varrho\} \simeq Y$.

Temos de (I) e (II) em (A.23) que $\forall W \in (Z_i - X_{n+1}), W \simeq Y$. Desta forma,

$$\forall Z \in \beta_0, \forall X \in (Z - X_{n+1}), X \simeq Y$$

que implica em

$$\forall X \in \left(\bigcup_{Z \in \beta_0} Z - X_{n+1} \right), X \simeq Y$$

Logo, pela definição 4.3.1

$$\forall Z \in (\beta_0 \backslash\backslash X_{n+1}), Z \simeq Y \tag{A.24}$$

Por construção,$\forall Z \in \beta_1, Z \not\simeq X_{n+1}$. Logo, $\forall Z \in \beta_1, Z - X_{n+1} = \{Z\}$ que implica em

$$\left(\bigcup_{X_i \in\, \beta_1} X_i - X_{n+1}\right) = \bigcup_{X_i \in\, \beta_1} \{X_i\} = \beta_1; \tag{A.25}$$

Portanto, pela definição 4.3.1

$$\beta_1 \backslash\backslash X_{n+1} = \beta_1. \tag{A.26}$$

$\Gamma = \{X_0, \ldots, X_n, X_{n+1}\}$. Logo, $Comp(\Gamma) = Comp(\{X_0, \ldots, X_n, X_{n+1}\})$ e, pela definição 4.3.2 da função $Comp$

$$\begin{aligned}
Comp(\Gamma) &= Comp(\{X_0, \ldots, X_n\}) \backslash\backslash X_{n+1} && \text{(A.27)}\\
&= Comp(\gamma) \backslash\backslash X_{n+1} && \text{(A.28)}\\
&= \bigcup_{X_i \in\, Comp(\gamma)} X_i - X_{n+1} && \text{(definição 4.3.1)}\\
& && \text{(A.29)}\\
&= \bigcup_{X_i \in \beta_0 \cup \beta_1} X_i - X_{n+1} && \text{(A.30)}\\
&= \left(\bigcup_{X_i \in \beta_0} X_i - X_{n+1}\right) \cup \left(\bigcup_{X_i \in \beta_1} X_i - X_{n+1}\right) && \text{(A.31)}\\
&= (\beta_0 \backslash\backslash X_{n+1}) \cup (\beta_1 \backslash\backslash X_{n+1}) && \text{(definição 4.3.1)}\\
& && \text{(A.32)}\\
&= (\beta_0 \backslash\backslash X_{n+1}) \cup \beta_1. && \text{(de A.26)}\\
& && \text{(A.33)}
\end{aligned}$$

Por hipótese da indução, $\forall X \in Comp(\gamma), X \simeq Y$. Logo, $\forall X \in \beta_1, X \simeq Y$, uma vez que $\beta_1 \subseteq Comp(\gamma)$. Desta forma, da última expressão acima e de (A.24), segue que $\forall X \in (\beta_0 \backslash\backslash X_{n+1}) \cup \beta_1$, $X \simeq Y$ e $X \simeq Y$. Logo, $\forall X \in Comp(\Gamma), X \simeq Y$.

∎

REFERÊNCIAS BIBLIOGRÁFICAS

ALCHOURRON, C. E.; GÄRDENFORS, P.; MAKINSON, D. On the logic of theory change: Partial meet contraction and revision functions. *The Journal of Symbolic Logic*, v. 50, n. 2, p. 510–530, 1985.

BRUNS, G.; GODEFROID, P. Model checking partial state spaces with 3-valued temporal logics. In: INTERNATIONAL CONFERENCE ON COMPUTER-AIDED VERIFICATION (CAV'99), 11., 1999, Trento. *Computer Aided Verification*. Berlin, Heidelberg: Springer, 1999. (LNCS, v. 1633), p. 274–287.

BRUNS, G.; GODEFROID, P. Generalized model checking: Reasoning about partial state spaces. In: INTERNATIONAL CONFERENCE ON CONCURRENCY THEORY, 11., 2000, University Park, PA, USA. *CONCUR 2000 - Concurrency Theory*. Berlin, Heidelberg: Springer, 2000. (LNCS, v. 1877), p. 168–182.

CHATZIELEFTHERIOU, G. et al. Abstract model repair. In: 4TH NASA FORMAL METHODS INTERNATIONAL SYMPOSIUM (NFM 2012), 4., 2012, Norfolk. *NASA Formal Methods - 2012*. Berlin, Heidelberg: Springer, 2012. (LNCS, v. 7226), p. 341–355.

CLARKE, E. M.; GRUMBERG, O.; PELED, D. A. *Model checking*. Cambridge, MA, USA: The MIT press, 1999. ISBN 0-262-03270-8.

GÄRDENFORS, P. *Knowledge in flux: Modeling the dynamics of epistemic states*. Cambridge, MA, USA: The MIT press, 1988.

GODEFROID, P.; HUTH, M.; JAGADEESAN, R. Abstraction-based model checking using modal transition systems. In: INTERNATIONAL CONFERENCE ON CONCURRENCY THEORY, 12., 2001, Aalborg, Denmark. *CONCUR 2001 - Concurrency Theory*. Berlin, Heidelberg: Springer, 2001, (LNCS, v. 2154). p. 426–440.

GRUMBERG, O. et al. When not losing is better than winning: Abstraction and refinement for the full μ-calculus. *Information and Computation*, Elsevier, v. 205, n. 8, p. 1130–1148, 2007.

GUERRA, P. T.; ANDRADE, A.; WASSERMANN, R. Toward the revision of CTL models through kripke modal transition systems. In: BRAZILIAN SYMPOSIUM ON FORMAL METHODS (SBMF), 16., 2013, Brasília, Brasil. *Formal Methods: Foundations and Applications*. Berlin, Heidelberg: Springer, 2013. (LNCS, v. 8195), p. 115–130.

GUERRA, P. T.; WASSERMANN, R. Revision of CTL models. In: IBERO-AMERICAN CONFERENCE ON ARTIFICIAL INTELLIGENCE (IBERAMIA 2010), 12., 2010,

Bahía Blanca, Argentina. *Advances in Artificial Intelligence - IBERAMIA 2010*. Berlin, Heidelberg: Springer, 2010. (LNCS, v. 6433), p. 153–162.

HUTH, M. Model checking modal transition systems using kripke structures. In: INTERNATIONAL CONFERENCE ON VERIFICATION, MODEL CHECKING, AND ABSTRACT INTERPRETATION (VMCAI), 3., 2002, Veneza, Itália. *Verification, Model Checking, and Abstract Interpretation*. Berlin, Heidelberg: Springer, Heidelberg, 2002, (LNCS, v. 2294). p. 302–316.

HUTH, M.; JAGADEESAN, R.; SCHMIDT, D. Modal transition systems: A foundation for three-valued program analysis. In: EUROPEAN SYMPOSIUM ON PROGRAMMING (ESOP'01), 10., 2001, Gênova. *Programming Languages and Systems*. Berlin, Heidelberg: Springer, 2001. (LNCS), p. 155–169.

HUTH, M.; RYAN, M. *Lógica em ciência da computação: modelagem e argumentação sobre sistemas*. Tradução e revisão técnica Valéria de Magalhães Iório. 2. ed. Rio de Janeiro: LTC, 2008.

KLEENE, S. C. et al. *Introduction to metamathematics*. New York: Van Nostrand, 1952.

RIBEIRO, J. S.; ANDRADE, A. A 3–valued contraction model checking game: Deciding on the world of partial information. In: INTERNATIONAL CONFERENCE ON FORMAL ENGINEERING METHODS (ICFEM), 17., 2015, Paris, França,. *Formal Methods and Software Engineering*. Berlin, Heidelberg: Springer, 2015. (LNCS, v. 9407), p. 84–99.

RIBEIRO, J. S. et al. *ARIIS - Analyser and Reviser for Incomplete Information Systems*. 2015. Disponível em: ⟨http://ariis.github.io⟩.

SHOHAM, S.; GRUMBERG, O. A game-based framework for CTL counterexamples and 3-valued abstraction-refinement. *ACM Trans. Comput. Logic*, ACM, New York, NY, USA, v. 9, n. 1, 2007.

STIRLING, C. *Modal and temporal properties of processes*. Berlin, Heidelberg: Springer, 2001.

WEHRHEIM, H. Bounded model checking for partial Kripke structures. In: INTERNATIONAL COLLOQUIUM ON THEORETICAL ASPECTS OF COMPUTING (ICTAC), 5., 2008, Istambul, Turquia,. *Theoretical Aspects of Computing - ICTAC 2008*. Berlin, Heidelberg: Springer, Heidelberg, 2008, (LNCS, v. 5160). p. 380–394.