Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Tableaux Clausal para Lógica Modal

Daniella Albuquerque dos Angelos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.a Dr.a Cláudia Nalon

Brasília
2016

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Tableaux Clausal para Lógica Modal

Daniella Albuquerque dos Angelos

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.a Dr.a Cláudia Nalon (Orientadora)
CIC/UnB

Prof.a Dr.a Célia Ghedini Ralha
CIC/UnB

Prof. Dr. Vander Ramos Alves
CIC/UnB

Prof. Dr. Rodrigo Bonifácio de Almeida
Coordenador do Bacharelado em Ciência da Computação

Brasília, 30 de janeiro de 2016

Dedicatória

Este trabalho é dedicado à memória de meu pai, Adauto dos Angelos, que sempre esteve presente incentivando minha boa formação e que com certeza estaria extremamente orgulhoso ao fim de mais essa etapa. Dedico também à minha mãe, Ilma Albuquerque, minha melhor amiga, sempre me apoiando em meus caminhos e vigiando para evitar que eu tropece.

Às minhas amigas, Jéssica e Fyama, às minhas irmãs amadas, Iasmin, Larissa e Aline, à minha tia Alba Albuquerque, por tudo que fez e faz por toda a família, e às amizades formadas ao longo do curso, também dedico este trabalho.

Agradecimentos

Agradeço à minha professora e orientadora Cláudia Nalon, por toda a dedicação, paciência e carinho. Por todas as correções, inspirações e por todo o tempo investido neste trabalho e na minha formação.

Agradeço ao Departamento de Ciência da Computação da Universidade de Brasília e a todos os professores que de alguma forma estiveram envolvidos na minha graduação. Em especial, ao professor Rodrigo Bonifácio, por todo o apoio ao longo do curso e por tantas oportunidades que surgiram trabalhando sob sua orientação.

Resumo

Este trabalho apresenta a definição de um cálculo tableaux clausal proposto para lógicas modais, além de apresentar uma implementação do mesmo. O objetivo geral é aplicar o método refutacional tableaux utilizando a forma normal construída para resolução clausal na tentativa de gerar o máximo de informações possíveis sobre conjuntos de fórmulas na linguagem modal, sem que seja necessário a especificação de um número elevado de regras de inferência no cálculo, o que tornaria a tarefa de implementar razoavelmente mais difícil. Estas informações extraídas contêm singularidades dos conjuntos de fórmulas que podem auxiliar em decisões relacionadas aos métodos de provas aplicados aos mesmos. As provas de correção para o cálculo são apresentadas. A implementação realizada, apesar de bem sucedida, revelou imperfeições com relação ao desempenho de execução. Esforços direcionados à melhoria de tais imperfeições, bem como uma análise das informações extraídas, são deixados como trabalhos futuros.

Palavras-chave: lógica modal, métodos de prova, tableaux clausal, cálculo

Abstract

This work presents a clausal tableaux calculus for modal logics and an implementation for it. The main goal is to combine aspects of two different proof methods, the inference rules of a tableaux calculus and the normal form employed in a clausal resolution method. The goal is to generate as much information as possible about sets of formulae in modal logic, without the need to define a large number of inference rules in the calculus, which would make the job of implementing the calculus reasonably harder. The pieces of extracted information may help to characterise singularities of the set of formulae, which could then help to make decisions about what proof method to apply to such a set. The implementation is correct, but does not perform well. Efforts in the direction of a more robust implementation and the analysis of the extracted information are left as future work.

Keywords: modal logic, proof methods, clausal tableaux, calculus

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Linguagens lógicas podem ser utilizadas para verificar propriedades de sistemas complexos, expressas através de possibilidade, crença, probabilidade entre outras modalidades [8, 26, 16, 15]. Lógicas modais têm sido estudadas na Ciência da Computação por permitirem a caracterização desses sistemas complexos, envolvendo noções tais como conhecimento e crença, por exemplo, em sistemas multiagentes [8, 26, 5], ou tempo, por exemplo, na formalização de problemas na área de *hardware* e verificação de sistemas concorrentes e de sistemas distribuídos [16, 15]. Essas noções expressam modalidades de verdade, ou quantificadores de verdade, isto é, a possibilidade de expressar quando ou onde um determinado objeto ou um resultado de uma ação é verdadeiro, se é necessariamente verdadeiro ou dificilmente verdadeiro, entre outros. Essas modalidades definem um ganho relevante em expressividade quando comparadas com outras linguagens lógicas. Diferentes modalidades definem diferentes lógicas modais.

Uma vez que um sistema tenha sido *especificado* na linguagem lógica, é possível utilizar métodos de provas para *verificar* se a implementação do sistema atende aos requisitos. Em geral, se Δ é o conjunto de fórmulas que representa a implementação e Γ é o conjunto de fórmulas que caracteriza a especificação, a verificação consiste em mostrar que é possível *deduzir* os aspectos especificados em Γ a partir de Δ. Cada método de prova define uma abordagem diferente para lidar com estas fórmulas e, portanto, apresenta características específicas lidando com diferentes lógicas em possíveis cenários distintos. Além disso, é possível combinar métodos com a finalidade de tentar explorar vantagens apresentadas por cada um.

Grande parte dos provadores para lógicas modais são baseados em tradução [28, 30, 31], o que acaba, por vezes, sendo inconveniente ao usuário, pois os resultados não são expressos na linguagem originalmente utilizada para a especificação. Além disso, para caracterizá-los de modo natural, diferentes aspectos de sistemas computacionais são descritos por *combinações* de diferentes lógicas modais. Para uma tal combinação, o método

de prova pode ser obtido pela combinação dos métodos para as linguagens componentes [2, 14]. É necessário, entretanto, algum cuidado para assegurar que informação suficiente seja compartilhada pelos métodos que estão sendo combinados. A combinação de linguagens, todavia, pode acarretar no aumento da complexidade ou mesmo na indecidibilidade do problema de satisfatibilidade na lógica resultante [11]. Portanto, é importante o desenvolvimento de técnicas que possam ser utilizadas de modo uniforme na combinação de métodos de prova para lógicas obtidas a partir de combinações.

O problema básico de satisfatibilidade local da lógica modal K é PSPACE-completo, tanto para o caso monomodal [21] quanto para o caso multimodal [17]. Além disso, sabe-se que métodos de prova para a lógica proposicional são intratáveis [7]. Em [23], uma estratégia de busca por provas, baseada em níveis modais foi apresentada. A implementação e a avaliação desta estratégia, comparando resultados de eficiência com provadores no estado da arte, mostram que a separação em níveis modais pode auxiliar na melhoria do tempo para a obtenção de provas [24].

Não existe, na literatura atual, uma proposta de cálculo tableaux clausal para a linguagem baseada em níveis modais definida em [23], sendo, portanto, o propósito geral deste trabalho definir o cálculo, chamado de $\mathcal{C}\mathsf{K}_n$, que consiste em um método de prova baseado em tableaux no qual as regras de inferência utilizam conjuntos de cláusulas (i.e. disjunções de literais) como numeradores. Para o cálculo proposicional, este método consiste de uma única regra de inferência que é aplicada repetidamente a um conjunto de cláusulas. Por se basear em um cálculo tableaux, esta combinação pode gerar uma grande quantidade de informação semântica sobre o conjunto de fórmulas inicial, o que, em nível de implementação, pode ocasionar em uma piora significativa no tempo de execução. Entretanto, a extração dessas informações serve como ferramenta de suporte ao entendimento teórico acerca das fórmulas e da linguagem estudada, ou seja, a prova obtida para cada fórmula contém informações importantes, por nível modal, que podem, por exemplo, expressar conhecimento sobre a satisfatibilidade da mesma, ou gerar um possível modelo que a satisfaz, no caso da fórmula ser satisfatível.

Os objetivos específicos deste trabalho consistem na prova de que as regras do cálculo proposto estão corretas, bem como na implementação deste cálculo, na avaliação experimental da implementação realizada e comparação com os resultados obtidos pelo provador KSP implementado em [24]. A entrada da implementação é o conjunto de cláusulas fornecido pelo provador KSP. A saída é a declaração de satisfatibilidade ou insatisfatibilidade do conjunto recebido como entrada. Os dados obtidos na experimentação prática são analisados, em conformidade com o enfoque corrente, de forma avaliativa e comparativa a partir de conjuntos de testes (*benchmarks*) estabelecidos. Tais resultados são apresentados no Capítulo 4.

O restante deste trabalho está organizado da seguinte maneira: o Capítulo 2 contém uma introdução teórica sobre linguagens lógicas e métodos de prova; o Capítulo 3 apresenta o cálculo proposto, bem como a prova de que suas regras são corretas; no Capítulo 4 é dada uma visão geral da implementação do cálculo e os testes realizados são apresentados e analisados; finalmente, o Capítulo 5 conclui o trabalho e apresenta propostas de possíveis trabalhos futuros.

Capítulo 2

Revisão Teórica

A lógica modal é uma extensão da lógica proposicional bastante conhecida. É determinada semanticamente pela avaliação de necessidade e possibilidade de uma proposição, levando em consideração diferentes possíveis contextos (ou *mundos*). Na literatura, representações análogas à de necessidade e possibilidade são utilizadas para a avaliação de estados epistemológicos, como por exemplo, conhecimento e crença, respectivamente [9]. Basicamente, uma proposição é *necessária* se ocorre em todos os mundos possíveis e *possível* se ocorre em algum mundo [6]. A ideia é que o resultado de uma ação ou objeto podem possuir valorações distintas em mundos diferentes, mas qualquer objeto que possuir valoração verdadeira em todos os mundos possíveis é necessária, enquanto a que ocorre em pelo menos um mundo, é possível.

Os operadores utilizados são os mesmos já conhecidos da lógica proposicional ($\wedge$, $\vee$, $\Rightarrow$ etc) com a inclusão de dois novos: $\boxed{a}$ e $\diamondsuit_a$. Uma sentença da forma $\boxed{a}\varphi$ (*necessariamente* φ) é verdadeira se, e somente se, φ é uma proposição verdadeira em todos os mundos possíveis, de acordo com o agente a; já uma sentença da forma $\diamondsuit_a\varphi$ (*possivelmente* φ) é verdadeira no caso em que φ é uma proposição verdadeira em algum mundo existente, também de acordo com o agente a.

Uma ilustração válida para a lógica modal é pensar em uma coleção de mundos possíveis, incluindo nosso próprio, o mundo real, e em cada mundo, as sentenças da linguagem assumem uma valoração verdadeira ou falsa. O propósito principal da lógica modal é modelar a ocorrência verdadeira destas sentenças. Esta lógica pode ser expressa de diferentes formas e ser caracterizada por diferentes axiomas.

2.1 Lógica

A lógica permite mostrar que uma determinada conclusão pode ser obtida a partir de um conjunto de premissas (ou hipóteses), isto é, lógica expressa raciocínio. É necessário

entender claramente o que é lógica e por que seu estudo é importante, portanto, esta seção é dedicada a este fim. Nesta seção são apresentadas algumas definições importantes que são centrais no estudo da lógica.

Lógica é a análise e a avaliação de argumentos [12], ou seja, a lógica procura estabelecer os mecanismos formais para determinar quais argumentos são bons e quais não são. É possível definir raciocínio sobre qualquer tópico, tanto filosófico como não-filosófico. Sendo assim, a lógica é uma ferramenta muito útil no raciocínio tanto de questões com um significado mais profundo, quanto em questões do dia a dia.

Pode-se pensar em três principais justificativas para o estudo de lógica. Primeiro, é difícil imaginar uma tomada de decisão ou a chegada a uma conclusão, uma resposta a alguma pergunta, sem pensar que houve uma linha de raciocínio envolvida, portanto raciocínio é importante. Raciocínio e habilidades analíticas em geral são fundamentais nas mais diversas áreas. Observe, então, que o estudo da lógica é importante pois lógica é uma ferramenta útil na evolução da compreensão e da avaliação de raciocínios. Segundo, a lógica pode aprofundar conhecimentos filosóficos. Sem uma linha de raciocínio concreta, um indivíduo pode apenas refletir de forma vaga sobre as questões de vida que a filosofia levanta, ou seja, há ausência de suporte ferramental para compreender e avaliar raciocínios desta natureza. Finalmente, lógica pode ser bastante divertido. Ela desafia o pensamento a seguir caminhos novos e realizar processos diferentes, como a montagem de um quebra-cabeça.

O objetivo do estudo da lógica na Ciência da Computação é desenvolver linguagens que modelam situações, de forma que se possa realizar raciocínio sobre elas formalmente. Pode-se pensar no raciocínio sobre situações como construção de argumentos. Em Ciência da Computação, essa construção deve ser formal para que os argumentos sejam válidos e possam ser defendidos rigorosamente ou executados em uma máquina [19].

O Exemplo 1 ilustra um problema sobre o qual é possível raciocinar usando lógica.

Exemplo 1 Problema sobre o qual o raciocínio é possível

Caso 1	Caso 2
Se o cachorro latir, o bebê vai acordar.	Se o cachorro latir, o bebê vai acordar.
O bebê não acordou.	O cachorro não latiu.

No primeiro caso, a conclusão de que o cachorro não latiu é trivial. Já no segundo caso, algumas pessoas se sentiriam tentadas a concluir que o bebê não acordou, o que não é necessariamente verdade, já que o bebê poderia ter acordado com outros ruídos ou por estar com fome, por exemplo. A intuição lógica pode ser trabalhada com a finalidade de

servir como ferramenta de raciocínio sobre argumentos. Por se tratar de um conhecimento acumulativo, quanto mais aprofundado o estudo de lógica, mais precisa se torna esta ferramenta.

2.1.1 Linguagens Lógicas

Uma linguagem lógica é uma formalização. Nesse sentido, é possível definir diversas linguagens lógicas que podem diferir em modalidades de aplicabilidade, expressividade e complexidade, por exemplo. A linguagem foco deste trabalho é a Linguagem Modal K_n. A atual seção tem o objetivo de apresentar teoricamente, e de forma abstrata, uma visão geral desta linguagem, que será construída a partir de linguagens menos expressivas, já apresentadas com a noção de mundos possíveis, muito utilizada na lógica modal.

É bastante usual que cientistas da computação tenham contato com lógica em algum momento da vida profissional. Este contato é, em geral, com a Lógica Proposicional Clássica. Esta lógica estuda argumentos cuja validade depende das noções de "se-então", "e", "ou", "não", além de noções similares. Estas noções definem os operadores mais usuais da lógica.

A Linguagem Proposicional é, portanto, uma formalização que provê regras específicas para construção de argumentos e testes de validade que utilizam estas noções [12]. A especificação apresentada neste documento, utiliza consoantes minúsculas como símbolos atômicos, ou proposicionais, que correspondem às sentenças que assumem a *valoração* verdadeira ou falsa no mundo foco do raciocínio, ou seja, as sentenças mais simples possíveis. Estes símbolos proposicionais podem ser combinados por meio de operadores. Os operadores desta linguagem, e suas respectivas semânticas, estão expressos na Tabela 2.1. Eles são usados para representar os fatos expressos por sentenças.

Tabela 2.1: Operadores da Lógica Proposicional.

$\neg p$	$:=$	não p
$p \wedge q$	$:=$	p e q
$p \vee q$	$:=$	p ou q
$p \Rightarrow q$	$:=$	se p então q
$p \Leftrightarrow q$	$:=$	p se, e somente se, q

Pode-se, ainda, combinar um ou mais operadores com os símbolos atômicos para formar fórmulas. As fórmulas que pertencem a Linguagem Proposicional são as *bem*

formadas como definido a seguir:

Definição 1 As fórmulas bem formadas da Linguagem Proposicional são definidas recursivamente como segue:

1. Os símbolos proposicionais $(p, q, \ldots)$ são fórmulas bem formadas

Agora considere que φ e ψ são fórmulas bem formadas. Então:

2. $\neg\varphi$
3. $\varphi \wedge \psi$
4. $\varphi \vee \psi$
5. $\varphi \Rightarrow \psi$
6. $\varphi \Leftrightarrow \psi$

também são fórmulas bem formadas.

Observe que é possível representar equivalências entre os operadores, conforme exemplos na Tabela 2.2. Neste sentido, os operadores $\neg$ e $\vee$ são suficientes para expressar toda a Lógica Proposicional. Os demais são utilizados para facilitar a leitura e diminuir o tamanho das fórmulas.

Tabela 2.2: Equivalência entre operadores.

$\neg(p \vee q)$	$=$	$\neg p \wedge \neg q$
$p \Rightarrow q$	$=$	$\neg p \vee q$
$p \Leftrightarrow q$	$=$	$(p \Rightarrow q) \wedge (q \Rightarrow p)$

Agora já podemos começar a raciocinar sobre algum mundo possível utilizando Lógica Proposicional.

Seja, então, τ o planeta em que vivemos. É possível escrever diversas afirmações que podem ter valoração verdadeira ou falsa em τ e é possível combiná-las para escrever fórmulas. Considere as proposições contidas no Exemplo 2.

Exemplo 2 Afirmações sobre o nosso planeta:

- r : A língua oficial do Brasil é o português.
- s : A lua é maior que a Terra.

- z : A Argentina fica na Ásia.
- q : Buenos Aires é a capital da Argentina.
- l : Buenos Aires fica na Ásia.

Sabemos que r e q são verdadeiras e que s, z e l são falsas. Observe as seguintes fórmulas criadas a partir destas proposições:

$$(a)\ r \vee s \qquad (b)\ r \wedge s \qquad (c)\ z \wedge q \Rightarrow l \tag{2.1}$$

Note que as fórmulas (a) e (c) são satisfeitas no planeta Terra (τ), ou seja, neste contexto, existe uma valoração verdadeira destas fórmulas. O que não ocorre para a fórmula (b), já que não é verdade que *A lua é maior que a Terra* em τ, a fórmula $r \wedge s$ nunca será verdadeira neste mundo. Podemos, então, esboçar esta valoração que satisfaz as fórmulas verdadeiras, conforme Figura 2.1.

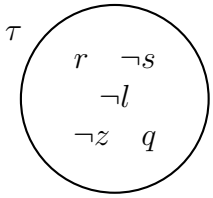

Figura 2.1: Valoração que satisfaz as fórmulas (a) e (c) em 2.1.

Na Lógica Proposicional, existe apenas um contexto no qual se pode raciocinar. Se uma fórmula é verdadeira, não é possível que ela seja falsa. Podemos estender a noção estudada nesta lógica para uma visão mais ampla, que dependa dos contextos que se é possível enxergar ou raciocinar sobre, aumentando o poder de expressividade. A Lógica Modal define uma dessas possíveis extensões.

Além das noções expressas através dos operadores proposicionais, a Lógica Modal estuda argumentos cuja validade depende das noções de "necessidade" e "possibilidade" (representações análogas são possíveis e foram mencionadas no início deste capítulo), introduzindo, portando, dois novos operadores com a semântica correspondente. Estas noções dizem respeito à avaliação das proposições em diferentes contextos, onde cada contexto é representado por um *mundo possível*. Basicamente, cada mundo possível, na lógica modal, corresponde a uma valoração clássica como vista anteriormente. A notação utilizada para representar esses dois novos operadores e suas respectivas semânticas na Linguagem Modal estão expressas na Tabela 2.3.

Tabela 2.3: Operadores Modais.

$\diamondsuit p$	:=	p é possível, ou seja, p é verdadeiro em algum mundo possível
$\Box p$	:=	p é necessário, ou seja, p é verdadeiro em todos os mundos possíveis

As regras de formação de fórmulas são as mesmas da Linguagem Proposicional, com a seguinte extensão para considerar os operadores modais:

Definição 2 Se φ é uma fórmula bem formada então:

6. $\diamondsuit \varphi$
7. $\Box \varphi$

também são fórmulas bem formadas.

Considere, por exemplo, τ e μ os planetas Terra e Marte, respectivamente, como os mundos sobre os quais é possível o raciocínio. Considere também, de forma meramente ilustrativa, as proposições apresentadas no Exemplo 3.

Exemplo 3 Proposições possíveis sobre a Terra e Marte

- p : Possui duas luas.
- r : Orbita ao redor do Sol.

A Tabela 2.4 relaciona os mundos τ e μ com as valorações para cada uma das proposições mostradas acima.

Tabela 2.4: Valoração para as proposições do Exemplo 3 nos mundos τ e μ.

Terra (τ)	Marte (μ)
$\neg p, r$	p, r

Observe que a proposição p possui valoração falsa no mundo τ e verdadeira no mundo μ. Portanto, tanto é possível p quanto $\neg p$, ou seja:

$$\diamondsuit p \wedge \diamondsuit \neg p \tag{2.2}$$

Por outro lado, a proposição r ocorre verdadeira em todos os mundos sobre os quais o raciocínio está sendo realizado, portando, r é necessário:

$$\Box r \tag{2.3}$$

A Figura 2.2 representa um possível modelo que satisfaz estas fórmulas.

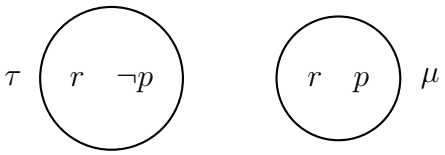

Figura 2.2: Modelo que satisfaz as fórmulas 2.2 e 2.3.

Um modelo é, portanto, uma estrutura que contém o conjunto de todos os mundos possíveis e uma função de valoração para os símbolos atômicos em cada um desses mundos. Além disso, o modelo também deve conter, necessariamente, a *relação* que os mundos exercem entre si. Na realidade, o raciocínio só é possível a partir de um mundo inicial e sobre os mundos que ele pode enxergar, ou seja, sobre os mundos que estão relacionados com ele. Essa relação é qualquer relação binária, podendo, por exemplo, um mundo se relacionar com ele mesmo. Além disso, a relação não é, em geral, simétrica (se ω_1 e ω_2 são mundos em um modelo e ω_1 está relacionado com ω_2, a recíproca não é necessariamente verdadeira).

Por simplicidade, a relação foi omitida no exemplo anterior, podendo, portanto, ser considerada total, ou seja, todos os mundos podem raciocinar sobre todos os mundos (incluindo eles próprios), pois a relação obrigatoriamente existe.

Como o raciocínio só é possível levando em consideração a relação e assumindo que esta não é total, a semântica dos operadores modais assume a seguinte definição:

Definição 3 Seja ω um mundo em um modelo dado e φ uma fórmula bem formada da Linguagem Modal:

1. φ é possível em ω se for verdadeira em *algum* mundo com o qual ω está relacionado
2. φ é necessária em ω se for verdadeira em *todos* os mundos com os quais ω está relacionado

A Figura 2.3 mostra um exemplo de modelo e, em seguida, fórmulas que são satisfeitas por este modelo na Linguagem Modal como definida até agora.

As seguintes fórmulas possuem valoração verdadeira em ω no modelo da Figura 2.3:

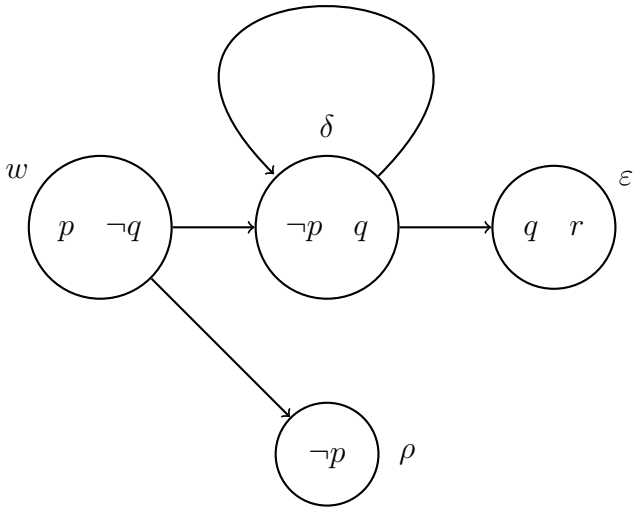

Figura 2.3: Exemplo de modelo e fórmulas satisfeitas por ele na Lógica Modal.

- $\Diamond\Diamond r$
- $p \wedge \Box\neg p$
- $\neg q \wedge \Diamond\Box q$

A estrutura de um modelo lógico vista pode ser estendida, a fim de se obter um ganho ainda maior de expressividade, levando em consideração diferentes *agentes*. Isto quer dizer que, ao invés do modelo ser definido com apenas uma relação, consideramos uma quantidade finita de relações rotuladas por um agente. Basicamente, as diferentes relações definem diferentes linhas de raciocínio possíveis. Esta lógica multiagente é conhecida como Lógica Multimodal. A Linguagem Modal apresentada anteriormente é um caso especial da Lógica Multimodal, onde o número de agentes é igual a um.

Os agentes são adicionados à sintaxe dos operadores modais e suas semânticas são estendidas para considerá-los conforme Definição 4.

Definição 4 Seja ω um mundo em um modelo dado, φ uma fórmula bem formada da Linguagem Modal e a um agente:

1. $\Diamond_a\varphi := \varphi$ é possível em ω pelo agente a se for verdadeira em *algum* mundo com o qual ω está relacionado pelo agente a
2. $\boxed{a}\varphi := \varphi$ é necessária em ω pelo agente a se for verdadeira em *todos* os mundos com o qual ω está relacionado pelo agente a

As Figuras 2.4 e 2.5 mostram exemplos de modelos e, em seguida, fórmulas que são satisfeitas por estes modelos na Linguagem Multimodal.

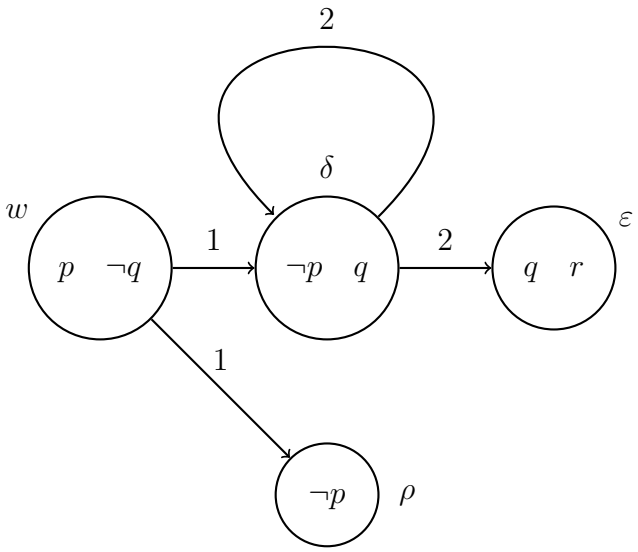

Figura 2.4: Exemplo da Figura 2.3 com as relações rotuladas por agentes.

As seguintes fórmulas possuem valoração verdadeira em ω no modelo da Figura 2.4:

- $\Diamond_1 \Diamond_2 r$
- $p \wedge \Box_1 \neg p$
- $\neg q \wedge \Diamond_1 \Box_2 q$

Já no modelo da Figura 2.5, são satisfatíveis as seguintes fórmulas:

- $\Box_2 \Diamond_1 q$
- $\Box_1 p \wedge \Box_2 \neg p$

2.2 Linguagem Modal K_n

A seção anterior tinha como objetivo apresentar uma introdução intuitiva às Linguagens Lógicas, sem se apegar a detalhes formais. A seção atual é dedicada a trazer uma abordagem mais técnica com relação aos conceitos de sintaxe e semântica da Linguagem Modal K_n.

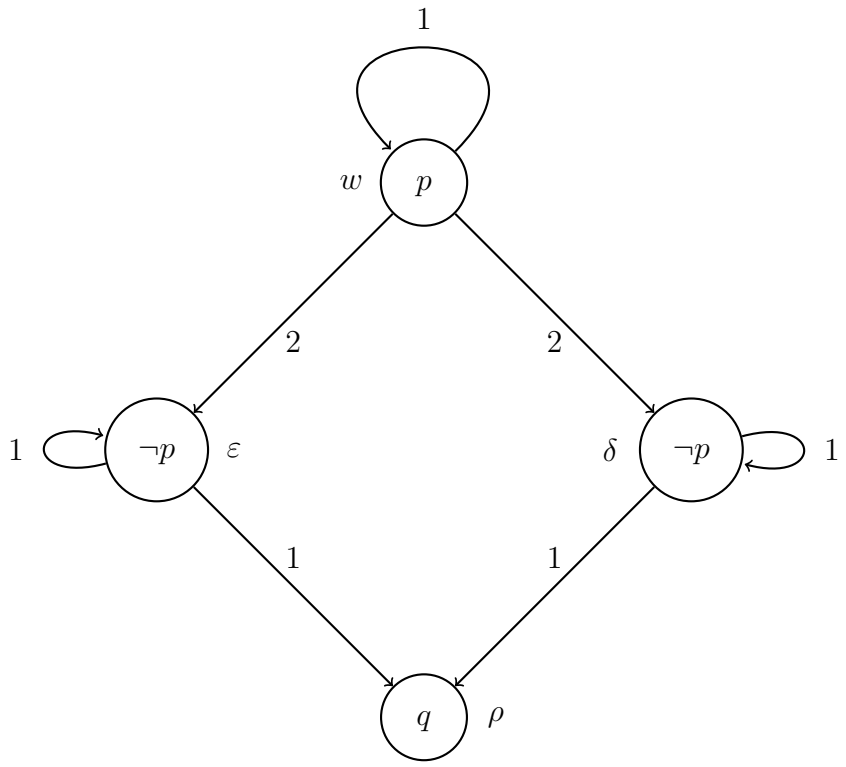

Figura 2.5: Exemplo de modelo e fórmulas satisfeitas por ele na Lógica Multimodal.

2.2.1 Sintaxe

A Linguagem Proposicional Clássica é formada por um conjunto enumerável de sentenças *atômicas* (ou símbolos proposicionais):

$$\mathcal{P} = \{p, q, r, \ldots\}$$

Estas são as sentenças mais simples possíveis.

Já as *sentenças moleculares* (não-atômicas) são formadas por meio das *operações sintáticas*, ou *operadores lógicos* de negação ($\neg$), de disjunção ($\vee$) e de necessidade ($\boxed{a}$, para todo $a \in \mathcal{A} = \{1, \ldots, n\}$, $n \in \mathbb{N}$). Observe que $\neg$ e $\boxed{a}$ são operadores unários e $\vee$ é um operador binário. Além disso, parênteses podem ser utilizados para evitar ambiguidade na leitura de sentenças.

Podemos então definir uma fórmula como uma combinação finita de sentenças atômicas e moleculares. A *Linguagem Modal* K_n é, portanto, equivalente ao seu conjunto de

fórmulas bem formadas, definido a seguir.

Definição 5 O conjunto de *fórmulas bem formadas da Linguagem Modal*, denotado por FBF_K é definido indutivamente como segue:

1. Os símbolos proposicionais estão em FBF_K;
2. Se $\varphi \in FBF_K$, então $\neg\varphi$ e $\boxed{a}\varphi$, $a \in \mathcal{A}$, estão em FBF_K; e
3. Se φ e ψ estão em FBF_K, então $(\varphi \vee \psi)$ está em FBF_K.

Outros operadores lógicos podem ser introduzidos como abreviações para fórmulas construídas a partir dos operadores já definidos, como usual. Em particular, neste trabalho as seguintes abreviações serão utilizadas: $(\varphi \wedge \psi) = \neg(\neg\varphi \vee \neg\psi)$ (conjunção), $(\varphi \Rightarrow \psi) = (\neg\varphi \vee \psi)$ (implicação), $\diamondsuit_a\varphi = \neg\boxed{a}\neg\varphi$ (possibilidade), $\mathbf{false} = (\varphi \wedge \neg\varphi)$ (*falsum*) e $\mathbf{true} = \neg\mathbf{false}$ (*verum*). A precedência dos operadores é dada na seguinte ordem (onde $a \in \mathcal{A}$):

1. $\{\neg, \boxed{a}, \diamondsuit_a\}$
2. $\{\wedge\}$
3. $\{\vee\}$
4. $\{\Rightarrow\}$

Quando não houver ambiguidade na leitura de uma fórmula, parênteses podem ser omitidos.

Como vimos, lógicas que envolvem n agentes na lógica modal, com $n \in \mathbb{N}$, como descrito acima, são chamadas de Lógicas Multimodais. A lógica modal com um único agente, onde temos apenas os operadores modais $\diamondsuit_1$ e $\boxed{1}$ (ou simplesmente $\diamondsuit$ e $\Box$), isto é, onde $\mathcal{A} = \{1\}$, bem como a Lógica Proposicional Clássica, onde $\mathcal{A} = \{\}$, são, portanto, linguagens contidas na Lógica Multimodal.

O *nível modal* de uma fórmula é o número máximo de ocorrências de operadores modais sob o qual a fórmula ocorre. Por exemplo, em $\boxed{a}\diamondsuit_b p$, o nível modal de p é 2.

É importante definir algumas convenções para minimizar dúvidas que poderiam surgir em algumas expressões. Expressões da forma $A_1 \wedge \ldots \wedge A_n$ e $A_1 \vee \ldots \vee A_n$ representam conjunções e disjunções arbitrárias mas não especificadas das sentenças $A_1, \ldots, A_n$. O objetivo é deixar claro que tanto $\wedge$ como $\vee$ são operadores associativos e que a omissão de parênteses não irá afetar o significado da fórmula.

2.2.2 Semântica

A ideia geral do significado de fórmulas na lógica proposicional modal pode ser entendida, de forma simples, em termos de uma coleção de mundos possíveis, uma função que atribui valores booleanos para cada símbolo proposicional em cada mundo e relações de acessibilidade, que são definidas com base em um conjunto de agentes.

Todas as definições apresentadas nesta seção foram adaptadas de [6].

Uma das estruturas mais estudadas em lógica modal corresponde à estrutura de Kripke.

Definição 6 Uma *estrutura de Kripke*, para o conjunto de símbolos proposicionais $\mathcal{P}$ e o conjunto de agentes $\mathcal{A} = \{1, \ldots, n\}$, é uma tupla $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$, onde: W é um conjunto não-vazio de mundos possíveis; $w_0 \in W$ é a raiz do modelo; $\forall\ a \in \mathcal{A}, R_a \subseteq W \times W$ e $\pi : W \times \mathcal{P} \longrightarrow \{false, true\}$

A partir da definição de estrutura de Kripke, podemos definir a satisfatibilidade e a validade de uma fórmula.

Definição 7 Seja $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ uma estrutura de Kripke sobre $\mathcal{P}$ e $\mathcal{A}$, e considere $w \in W$, $p \in \mathcal{P}$ e φ e ψ fórmulas bem formadas. A *relação de satisfatibilidade*, denotada por $\models^{\mathcal{M}}_{w} \varphi$, entre o mundo w e uma fórmula φ no modelo $\mathcal{M}$, é definida indutivamente como se segue:

1. $\models^{\mathcal{M}}_{w} p$ se e somente se, $\pi(w, p) = true$;
2. $\models^{\mathcal{M}}_{w} \neg\varphi$ se e somente se, $\not\models^{\mathcal{M}}_{w} \varphi$
3. $\models^{\mathcal{M}}_{w} \varphi \vee \psi$ se e somente se, $\models^{\mathcal{M}}_{w} \varphi$ ou $\models^{\mathcal{M}}_{w} \psi$
4. $\models^{\mathcal{M}}_{w} \boxed{a}\varphi$ se e somente se, $\forall\ t \in W$ com $a \in \mathcal{A}$ e $(w, t) \in R_a$, tem-se que $\models^{\mathcal{M}}_{t} \varphi$.

Um conjunto de fórmulas $\Gamma = \{\gamma_1, \ldots, \gamma_r\}$, $r \in \mathbb{N}$, é satisfeito em um mundo w se, e somente se, todas as suas fórmulas são satisfeitas neste mundo, isto é, $\models^{\mathcal{M}}_{w} \Gamma$ é equivalente a $\models^{\mathcal{M}}_{w} \gamma_1 \wedge \ldots \wedge \gamma_r$.

Definição 8 Seja $\varphi \in FBF_K$, dizemos que φ é *satisfatível* se existe um modelo $\mathcal{M}$ tal que $\models^{\mathcal{M}}_{w_0} \varphi$.

Definição 9 Seja $\varphi \in FBF_K$, dizemos que φ é *válida* se para todo modelo $\mathcal{M}$ temos que $\models^{\mathcal{M}}_{w_0} \varphi$.

É importante notar que a verificação da satisfatibilidade de fórmulas modais pode ser restrita a estruturas de Kripke que correspondem a árvores finitas (grafos acíclicos dirigidos) [4]. Nós denotamos por $\mathsf{profundidade}(w)$ o tamanho do caminho único entre w_0 e w através da união das relações de acessibilidade em $\mathcal{M}$. Nós chamamos de *camada modal* a classe de equivalência entre os mundos de mesma profundidade em um modelo. Daí segue que o problema de verificar a satisfatibilidade de uma fórmula φ em w_0 pode ser reduzido ao problema de se checar a satisfatibilidade de todas as suas subfórmulas que ocorrem na camada modal de um modelo correspondente ao nível modal onde estas subfórmulas ocorrem [1].

Observe que, como dito anteriormente, a Lógica Modal K_n inclui a Lógica Proposicional e a Lógica Modal com um único agente. Para tal, basta tomar $W = \{w_0\}$, com $\mathcal{A} = \{\}$, no caso da Lógica Proposicional, e, no caso da Lógica Modal com um único agente, W como um conjunto não-vazio de mundos possíveis, com $\mathcal{A} = \{1\}$.

2.3 Métodos de Prova

Informalmente, uma prova é um argumento que convence. Formalmente, uma prova é um conjunto de fórmulas, é um objeto finito construído de acordo com regras de sintaxe fixadas que fazem referência unicamente à estrutura das fómulas e não ao seu significado pretendido. As regras sintáticas que definem as provas que podem ser obtidas especificam o que chamamos de *método de prova*. Um método de prova é *correto* para uma lógica em particular se para qualquer fórmula que possuir uma prova, esta fórmula se trata de uma fórmula válida nesta lógica. Um método de prova é *completo* para uma lógica se toda fórmula válida daquela lógica possui uma prova. Dessa forma, um método correto e completo de provas nos permite construir "testemunhas", chamadas de provas, de que determinada fórmula é válida [10].

Existem inúmeros tipos de métodos de provas. Genericamente, podemos dividi-los em duas categorias: *analíticos* e *sintéticos*. Os termos são sugestivos, mas não extremamente precisos. Um método de prova analítico decompõe uma fórmula em partes mais simples. Um método de prova sintético, por outro lado, constrói uma prova até a fórmula que se deseja obter a demonstração. O primeiro tende a ser mais facilmente aplicado, já que o espaço em que se trabalha é limitado: nunca olha-se muito além do escopo onde se

encontra a fórmula que se está tentando provar. Já o segundo costuma ser mais utilizado por produzir provas especialmente elegantes.

O exemplo mais comum de método sintético é um *sistema axiomático*. Algumas fórmulas são tomadas como axiomas. Uma prova começa com esses axiomas e, usando regras de inferência definidas, que produzem novas fórmulas, é construída uma sequência de fórmulas que finalmente termina na fórmula que se está tentando provar.

Cálculos Tableaux são um dos mais comuns métodos de prova analítica. Este tipo de sistema é conhecido como *sistema refutacional*. Para provar uma fórmula, precisamos inicialmente negá-la, analisando as consequências de ser tomada tal ação, utilizando uma estrutura em forma de árvore. Se, as consequências resultarem em algo impossível, pode-se concluir que a fórmula original está provada. Falaremos mais sobre este sistema na Seção 2.3.2.

2.3.1 Resolução Clausal

Este método de prova é também um exemplo de sistema refutacional: para podermos provar a fórmula φ, nós transformamos a sua negação ($\neg\varphi$) em uma forma normal, para, então, aplicarmos as regras de inferência ao conjunto de cláusulas resultante com a finalidade de gerar a cláusula vazia, que indicará uma contradição no conjunto de cláusulas, provando a fórmula original.

No caso da lógica propositional, a forma normal mais utilizada é a Forma Normal Conjuntiva (CNF), onde fórmulas são conjunções de disjunções de símbolos proposicionais ou suas negações [20]. Existe uma forma normal específica para a Lógica Modal K_n: a Forma Normal Separada para Sistemas Normais baseada em Níveis Modais $\mathsf{SNF}_{\mathrm{K}}$, que faz com que os contextos referentes aos diferentes agentes e aos diferentes níveis modais sejam separados. As regras de transformação e a prova de correção do método de tradução para $\mathsf{SNF}_{\mathrm{K}}$ podem ser encontrados em [23]. Uma fórmula em $\mathsf{SNF}_{\mathrm{K}}$ é representada por um conjunto de cláusulas, que são verdadeiras nos respectivos níveis modais. Uma fórmula em $\mathsf{SNF}_{\mathrm{K}}$ possui o formato apresentado em 2.4:

$$\bigwedge_i ml : C_i \tag{2.4}$$

onde cada C_i é uma cláusula e ml é o nível modal onde a cláusula ocorre.

As seguintes definições também são necessárias:

Definição 10 *Literal* é um símbolo proposicional p ou sua negação $\neg p$ ($p \in \mathcal{P}$). Denotamos por $\mathcal{L}$ o conjunto de todos os literais.

Definição 11 Um *literal modal* é uma fórmula na forma $\boxed{a}l$ ou sua negação $\neg\boxed{a}l$, sendo $l \in \mathcal{L}$ e com $a \in \mathcal{A}$.

Uma cláusula pode assumir um dos seguintes formatos:

- Uma cláusula de literais:

$$ml : \bigvee_{b=1}^{r} l_b$$

- Uma cláusula a-positiva:

$$ml : l' \Rightarrow \boxed{a}l$$

- Uma cláusula a-negativa:

$$ml : l' \Rightarrow \neg\boxed{a}l$$

onde l, l' e $l_b \in \mathcal{L}$, $a \in \mathcal{A}$, r, b e $ml \in \mathbb{N}$. Quando a referência ao contexto do agente não for importante, cláusulas a-positivas (respectivamente, a-negativas) são referenciadas tão somente como *cláusulas positivas* (respectivamente, *cláusulas negativas*).

Uma vez que as fórmulas estejam em sua forma normal, o método de resolução pode ser aplicado, ou seja, as regras de inferência podem ser aplicadas ao conjunto de cláusulas resultante. Nós omitiremos a descrição do método, que pode ser encontrada em [23], uma vez que para os objetivos deste trabalho precisamos tão somente da definição da forma normal apresentada acima.

2.3.2 Tableaux

Métodos de demonstração utilizando tableaux são frequentemente adotados, com sucesso, em lógica modal para prover procedimentos de decisão. O tableaux consiste em uma prova representada graficamente na forma de árvore. É um método analítico baseado na prova por contradição [10].

Definição 12 Uma *regra* σ no método tableaux consiste em um numerador N, acima da linha, e uma lista (finita) de denominadores $D_1, D_2, \ldots, D_k$ (abaixo da linha), separados por barras verticais. O numerador e cada denominador são um conjunto finito de fórmulas [27].

O numerador de cada regra do tableaux contém uma ou mais fórmulas distintas chamadas de *fórmula(s) principal(is)*. Todas as fórmulas no numerador devem ser consideradas. Já as barras verticais no denominador têm o significado de disjunção. Assim, cada regra é lida de cima para baixo e se o conjunto de fórmulas do numerador é satisfatível, então pelo menos um dos denominadores também deve ser. Um *cálculo baseado em tableaux* para uma lógica L ($\mathcal{C}_L$) é um conjunto finito de regras σ [13].

Dado um $\mathcal{C}_L$, um *tableaux* para uma fórmula φ é uma árvore tal que sua raiz contém φ e os demais nós contém conjuntos finitos de fórmulas obtidas a partir de seus nós-pais, através do uso de uma regra em $\mathcal{C}_L$. Uma *haste*, em um tableaux, representa um caminho entre a raiz da árvore e um de seus nós.

As definições apresentadas a seguir são adaptações das definições encontradas em [10].

Definição 13 Seja Δ um conjunto de regras de tableaux. Dizemos que ψ é *obtível a partir de* φ por aplicações de regras em Δ se existe um tableaux para φ, que utiliza somente as regras contidas em Δ, e este tableaux possui um nó que contém ψ.

Após a aplicação de uma regra, é necessário verificar se foi inserida inconsistência na haste, ou seja, se a negação de alguma fórmula obtida já está presente naquela haste. Em caso positivo, um novo nó contendo $\perp$ é inserido ao mesmo.

Definição 14 Uma haste em um tableaux é dita *fechada* se seu nó final contém apenas $\perp$. Um tableaux é dito *fechado* se cada uma de suas hastes estiver fechada. Um tableaux é chamado de *aberto* se não estiver fechado.

Definição 15 Um conjunto finito Γ de fórmulas é dito *$\mathcal{C}_L$-satisfatível* se cada $\mathcal{C}_L$-tableaux para Γ é aberto. Se existir pelo menos um $\mathcal{C}_L$-tableaux fechado para Γ então Γ é *$\mathcal{C}_L$-insatisfatível*.

Definição 16 Um $\mathcal{C}_L$ é *correto* se para todos os conjuntos finitos Γ de fórmulas, se Γ é satisfatível em L então é também $\mathcal{C}_L$-satisfatível. É *completo* se para todos os conjuntos finitos Γ de fórmulas, se Γ é $\mathcal{C}_L$-satisfatível então Γ é satisfatível em L.

Qualquer cálculo baseado em tableaux para uma lógica L contendo somente regras corretas com respeito a L, como definido em 17, é correto [10].

Definição 17 Seja σ uma regra de inferência pertencente a um $\mathcal{C}_L$. Dizemos que σ é correta com respeito a L se para qualquer instância, ou aplicação, σ' de σ, sendo o numerador de σ' satisfatível em L, o denominador de σ' também for satisfatível em L.

Em uma especificação de um $\mathcal{C}_L$, é fundamental que se apresente também a prova de que suas regras são corretas, ou seja, a prova de que o cálculo proposto é correto.

2.3.3 Tableaux Clausal

Um cálculo tableaux é chamado de *tableaux clausal* quando suas regras são formadas por fórmulas em uma forma normal clausal. A transformação de fórmulas modais em cláusulas foi utilizado inicialmente por Mints em [22] e Hudelmaier em [18] e então aplicadas em um cálculo tableaux por Nguyen em [25]. Huderlmaier e Nguyen mostraram que fórmulas clausais podem apresentar melhorias no uso de espaço e nas tomadas de decisão para algumas lógicas modais [18, 25]. Em [13], Goré e Nguyen também mostraram que o uso de cláusulas pode simplificar a tarefa de desenvolver um cálculo tableaux para certas lógicas modais.

Em nível de implementação, além das vantagens mencionadas anteriormente, definir as regras de um cálculo tableaux como cláusulas também pode diminuir o número de regras que devem ser especificadas. Essa vantagem é importante pois reduz drasticamente a possibilidade de inserção de erros na implementação das regras deste cálculo.

O cálculo proposto por este trabalho para a Lógica K_n, chamado $\mathcal{C}\mathsf{K}_n$, se trata de um cálculo tableaux clausal, ou seja, utiliza, como numeradores de suas regras, cláusulas na forma $\mathsf{SNF}_{\mathrm{K}}$, especificada na Seção 2.3.1. Este cálculo está especificado no Capítulo 3.

Capítulo 3

Cálculo $\mathcal{C}\mathsf{K}_n$

O cálculo proposto compreende um conjunto de regras de inferência para lidar com raciocínio tanto proposicional quanto modal. Antes de apresentá-lo, é necessário definir a notação utilizada a fim de que a apresentação do cálculo seja suficientemente formal.

3.1 Cláusulas

Para a Definição 18, considere a forma normal $\mathsf{SNF}_{\mathrm{K}}$ apresentada na Seção 2.3.1.

Definição 18 Sejam γ^i, λ^i e θ^i representações de cláusulas proposicionais, a-negativas e a-positivas, respectivamente, conforme segue:

1. $\gamma^i := i : \bigvee_{k=1}^{t} l_k$
2. $\lambda^i := i : l \Rightarrow \langle a \rangle m$
3. $\theta^i := i : l \Rightarrow [a] m$

Sejam, também, Γ^i, Ω^i e Θ^i, os conjuntos iniciais das respectivas cláusulas no nível modal i de forma que:

1. $\Gamma^i := \{\gamma_1^i, \ldots, \gamma_p^i\}$, $p \in \mathbb{N}$
2. $\Lambda^i := \{\lambda_1^i, \ldots, \lambda_q^i\}$, $q \in \mathbb{N}$
3. $\Theta^i := \{\theta_1^i, \ldots, \theta_r^i\}$, $r \in \mathbb{N}$

Finalmente, considere Δ^i um conjunto de literais, modais ou não, que pertencem ao nível modal i com $\Pi^i = \{\Delta_1^i, \ldots, \Delta_k^i\}$, para algum $k \in \mathbb{N}$. Observe que se $d \in \Delta_j^i$ então $d \in \mathcal{L}$ ou d é da forma $\langle a \rangle l$ ou $[a] l$ (literal modal), para algum $l \in \mathcal{L}$.

3.2 Cálculo

Considere que inicialmente, Π^i contém um conjunto Δ^i_1 vazio, para todos os níveis modais. Apresentamos a seguir as regras de inferência que são aplicadas a estes conjuntos.

A primeira regra de inferência, (PROP), tem como numerador todos os conjuntos de literais Δ^i_j que são satisfeitos no nível modal i e considera uma cláusula $\{i : \bigvee_{k=1}^{t} l_k\}$ específica em Π^i. O resultado da aplicação de (PROP), dado no seu denominador, é a expansão de cada um dos conjuntos de literais do numerador com um dos literais da cláusula considerada. A intuição por trás desta regra de inferência é que se Δ^i_j e $\{i : \bigvee_{k=1}^{t} l_k\}$ são ambos satisfatíveis, então pelo menos um dos literais em $\{i : \bigvee_{k=1}^{t} l_k\}$ é satisfatível; logo, pelo menos um dos conjuntos obtidos no denominador também será satisfatível.

$$\frac{\Delta^i_j \in \Pi^i \qquad \Gamma^i \cup \{i : \bigvee_{k=1}^{t} l_k\}}{\Delta^i_j \cup \{l_1\} \mid \ldots \mid \Delta^i_j \cup \{l_t\}} \tag{PROP}$$

As regras (NEG) e (POS) dizem respeito às cláusulas modais. São regras equivalentes, apesar de tratarem de tipos de cláusulas modais diferentes, a diferença é somente a semântica dos operadores, portanto, a descrição que segue faz referência à regra (NEG), sendo a descrição pra regra (POS), análoga.

Essa regra possui como numerador todos os conjuntos de literais Δ^i_j que são satisfeitos no nível modal i e considera as cláusulas modais a-negativas no mesmo nível, ou seja, $\{i : l \Rightarrow \diamondsuit_a m\}$ em Λ^i. A aplicação de (NEG) resulta na duplicação do conjunto Δ^i_j para considerar tanto a negação do antecedente da cláusula modal como o literal modal ocorrendo como consequente desta cláusula. Como, por hipótese, a cláusula modal é satisfatível, temos que ou $\neg l$ é satisfatível, ou $\diamondsuit_a m$ é satisfatível e Δ^i_j também é satisfatível, então pelo menos um dos denominadores é satisfatível.

$$\frac{\Delta^i_j \in \Pi^i \qquad \Lambda^i \cup \{i : l \Rightarrow \diamondsuit_a m\}}{\Delta^i_j \cup \{\neg l\} \mid \Delta^i_j \cup \{l, \diamondsuit_a m\}} \tag{NEG}$$

$$\frac{\Delta^i_j \in \Pi^i \qquad \Theta^i \cup \{i : l \Rightarrow \boxed{a} m\}}{\Delta^i_j \cup \{\neg l\} \mid \Delta^i_j \cup \{l, \boxed{a} m\}} \tag{POS}$$

A última regra do cálculo, (EXP), tem como numerador um conjunto Δ^i_k no i-ésimo nível modal, tal que existam $m_0, m_1, \ldots, m_r \in \mathcal{L}, r \geq 0$ literais no escopo modal de forma

que $\{\langle a \rangle m_0, [a] m_1, \ldots, [a] m_r\} \subseteq \Delta_k^i$. Ou seja, este conjunto contém pelo menos um literal no escopo modal negativo e um número arbitrário de literais no escopo modal positivo. O que esta regra expressa é que, sendo Δ_k^i satisfatível, deve existir pelo menos um conjunto de literais, Δ_j^{i+1}, no próximo nível que satisfaça $\{m_0, m_1, \ldots, m_r\}$.

$$\frac{\Delta_k^i \cup \{\langle a \rangle m_0\} \cup \{[a] m_1, \ldots, [a] m_r\} \in \Pi^i}{\Delta_j^{i+1} \cup \{m_0, m_1, \ldots, m_r\}} \quad \text{para algum } \Delta_j^{i+1} \in \Pi^{i+1} \qquad \text{(EXP)}$$

3.2.1 $\mathcal{C}\mathsf{K}_n$ é Correto

Esta seção apresenta a prova de que o cálculo proposto na seção anterior é correto. Como discutido anteriormente, para provar que $\mathcal{C}\mathsf{K}_n$ é correto, basta mostrar que cada uma de suas regras são corretas [10].

Lema 1 (PROP). Sejam $\gamma^i = i : \bigvee_{k=1}^t l_k$ uma cláusula proposicional em SNF_K tal que $\gamma^i \in \Gamma^i$ e $\Delta_j^i \in \Pi^i$ um conjunto de literais no nível modal i. Se γ^i e Δ_j^i são satisfatíveis em K_n então existe $1 \leq r \leq t$ tal que $\Delta_j^i \cup \{l_r\}$ também é satisfatível em K_n.

Demonstração. Seja Γ^i o conjunto inicial de cláusulas proposicionais do nível modal i, com $\gamma^i = i : \bigvee_{k=1}^t l_k \in \Gamma^i$, e considere $\Delta_j^i \in \Pi^i$ um conjunto de literais que, por hipótese, pertencem ao i-ésimo nível modal. Suponha que estas hipóteses sejam satisfatíveis em K_n. Pela Definição 8, sabemos que existe um modelo $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ e um mundo $w \in W$ tais que:

- $\mathsf{profundidade}(w) = i$
- $\models_w^{\mathcal{M}} \bigvee_{k=1}^t l_k$ e
- $\models_w^{\mathcal{M}} d, \forall\ d \in \Delta_j^i$

Para provar que esta regra é correta, temos que mostrar que em uma instância dela, pelo menos um de seus denominadores é satisfatível. De fato, temos que:

- $\models_w^{\mathcal{M}} \bigvee_{k=1}^t l_k \Leftrightarrow \models_w^{\mathcal{M}} l_1$ ou $\ldots$ ou $\models_w^{\mathcal{M}} l_t$, pela Definição 7 item 3

Ou seja, existe pelo menos um índice r, onde $1 \leq r \leq t$, tal que $\models_w^{\mathcal{M}} l_r$. Portanto, $\Delta_j^i \cup \{l_r\}$ é satisfatível em K_n. □

Lema 2 (NEG). Sejam $\lambda^i = i : l \Rightarrow \langle a \rangle m$ uma cláusula a-negativa em SNF_K tal que $\lambda^i \in \Lambda^i$ e $\Delta_j^i \in \Pi^i$ um conjunto de literais no nível modal i. Se λ^i e Δ_j^i são satisfatíveis em K_n então pelo menos um de $\Delta_j^i \cup \{\neg l\}$ ou $\Delta_j^i \cup \{l, \langle a \rangle m\}$ também é satisfatível.

Demonstração. Seja Λ^i o conjunto inicial de cláusulas modais negativas no nível modal i, com $\lambda^i = i : l \Rightarrow \diamondsuit_a m \in \Lambda^i$, e considere $\Delta^i_j \in \Pi^i$ um conjunto de literais que, por hipótese, pertencem ao i-ésimo nível modal. Suponha que estas hipóteses sejam satisfatíveis em K_n. Pela Definição 8, sabemos que existe um modelo $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ e um mundo $w \in W$ tais que:

- $\mathsf{profundidade}(w) = i$
- $\models^{\mathcal{M}}_w l \Rightarrow \diamondsuit_a m$
- $\models^{\mathcal{M}}_w d, \forall\ d \in \Delta^i_j$

Para provar que esta regra é correta, temos que mostrar que em uma instância dela, pelo menos um de seus denominadores é satisfatível. De fato, temos que:

- $\models^{\mathcal{M}}_w l \Rightarrow \diamondsuit_a m \Leftrightarrow \models^{\mathcal{M}}_w \neg l$ ou $\models^{\mathcal{M}}_w \diamondsuit_a m$, pela Definição 7 item 3.

Se valer que $\models^{\mathcal{M}}_w \neg l$ então $\Delta^i_j \cup \{\neg l\}$ é satisfatível. Caso contrário, $\not\models^{\mathcal{M}}_w \neg l$ e $\models^{\mathcal{M}}_w \diamondsuit_a m$ valem, fazendo com que $\Delta^i_j \cup \{l, \diamondsuit_a m\}$ seja satisfatível, pela Definição 7 item 2 e a equivalência à disjunção do item 3. □

Lema 3 (POS). Sejam $\theta^i = i : l \Rightarrow \boxed{a} m$ uma cláusula a-positiva em $\mathsf{SNF_K}$ tal que $\theta^i \in \Theta^i$ e $\Delta^i_j \in \Pi^i$ um conjunto de literais no nível modal i. Se θ^i e Δ^i_j são satisfatíveis em K_n então pelo menos um dos conjuntos $\Delta^i_j \cup \{\neg l\}$ ou $\Delta^i_j \cup \{l, \boxed{a} m\}$ também é satisfatível.

Demonstração. Seja Θ^i o conjunto inicial de cláusulas modais positivas no nível modal i, com $\theta^i = i : l \Rightarrow \boxed{a} m \in \Theta^i$, e considere $\Delta^i_j \in \Pi^i$ um conjunto de literais que, por hipótese, pertencem ao i-ésimo nível modal. Suponha que estas hipóteses sejam satisfatíveis em K_n. Pela Definição 8, sabemos que existe um modelo $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ e um mundo $w \in W$ tais que:

- $\mathsf{profundidade}(w) = i$
- $\models^{\mathcal{M}}_w l \Rightarrow \boxed{a} m$
- $\models^{\mathcal{M}}_w d, \forall\ d \in \Delta^i_j$

Para provar que esta regra é correta, temos que mostrar que em uma instância dela, pelo menos um de seus denominadores é satisfatível. De fato, temos que:

- $\models^{\mathcal{M}}_w l \Rightarrow \boxed{a} m \Leftrightarrow \models^{\mathcal{M}}_w \neg l$ ou $\models^{\mathcal{M}}_w \boxed{a} m$, pela Definição 7 item 3.

Se valer que $\models^{\mathcal{M}}_w \neg l$ então $\Delta^i_j \cup \{\neg l\}$ é satisfatível. Caso contrário, $\not\models^{\mathcal{M}}_w \neg l$ e $\models^{\mathcal{M}}_w \boxed{a} m$ valem, fazendo com que $\Delta^i_j \cup \{l, \boxed{a} m\}$ seja satisfatível, pela Definição 7 item 2 e a equivalência à disjunção do item 3. □

Lema 4 (EXP). Seja $\Delta_k^i \in \Pi^i$ um conjunto de literais no i-ésimo nível modal para o qual existam $m_0, m_1 \in \mathcal{L}, \ldots, m_r, r \geq 0$ tais que $\{\langle a \rangle m_0, \boxed{a} m_1, \ldots, \boxed{a} m_r\} \subseteq \Delta_k^i$. Se Δ_k^i é satisfatível em K_n então existe $\Delta_j^{i+1} \in \Pi^{i+1}$ tal que $\Delta_j^{i+1} \cup \{m_0, m_1, \ldots, m_r\}$ seja satisfatível.

Demonstração. Assuma que $\Delta_k^i \cup \{\langle a \rangle m_0\} \cup \{\boxed{a} m_1, \ldots, \boxed{a} m_r\} \in \Pi^i$ é satisfatível em K_n, ou seja, que existe um modelo $\mathcal{M} = (W, w_0, R_1, \ldots, R_n, \pi)$ e um mundo $w \in W$, com $\mathsf{profundidade}(w) = i$, tal que, pela Definição 8:

- $\models_w^{\mathcal{M}} \Delta^i \cup \{\langle a \rangle m_0\} \cup \{\boxed{a} m_1, \ldots, \boxed{a} m_r\}$.

Pela definição de satisfatibilidade de conjuntos:

(1) $\models_w^{\mathcal{M}} \langle a \rangle m_0 \wedge \boxed{a} m_1 \wedge \ldots \wedge \boxed{a} m_r$.

Pela definição de satisfatibilidade da conjunção, temos que $\models_w^{\mathcal{M}} \langle a \rangle m_0$. Logo, existe um mundo w', onde wR_aw' e $\mathsf{profundidade}(w') = i + 1$, tal que $\models_{w'}^{\mathcal{M}} m_0$ (Definição 7, equivalência negativa do item 4). Como $\models_w^{\mathcal{M}} \boxed{a} m_k$, para todo $0 \leq k \leq r$, e wR_aw', pela definição de satisfatibilidade do operador $\boxed{a}$, temos que $\models_{w'}^{\mathcal{M}} m_k$, para todo $0 \leq k \leq r$.

Para fins de contradição, suponha que não exista $\Delta_j^{i+1} \in \Pi^{i+1}$, conjunto de literais no nível modal $i + 1$, tal que $\Delta_j^{i+1} \cup \{m_0, m_1, \ldots, m_r\}$ seja satisfatível no modelo $\mathcal{M}$. Isto significa que todos os conjuntos Δ_j^{i+1} satisfazem $\neg m_0 \vee \neg m_1 \vee \ldots \vee \neg m_r$. De novo pela definição de satisfatibilidade do operador $\boxed{a}$ temos que $\models_{w''}^{\mathcal{M}} \boxed{a}(\neg m_0 \vee \neg m_1 \vee \ldots \vee \neg m_r)$ para todo w'' com $\mathsf{profundidade}(w'') = i$. Em particular, porque $\mathsf{profundidade}(w) = i$, temos que:

(2) $\models_w^{\mathcal{M}} \boxed{a}(\neg m_0 \vee \neg m_1 \vee \ldots \vee \neg m_r)$.

De (1), (2) e porque $\boxed{a}\varphi \wedge \boxed{a}\psi$ é semanticamente equivalente a $\boxed{a}(\varphi \wedge \psi)$, obtemos que:

$$\models_w^{\mathcal{M}} \boxed{a}((\neg m_0 \vee \neg m_1 \vee \ldots \vee \neg m_r) \wedge m_1 \wedge \ldots \wedge m_r) \wedge \langle a \rangle m_0.$$

Pelo princípio da resolução, obtemos $\models_w^{\mathcal{M}} \boxed{a} \neg m_0 \wedge \langle a \rangle m_0$. Finalmente, pela definição de satisfatibilidade dos operadores $\boxed{a}$ e $\langle a \rangle$, temos $\models_w^{\mathcal{M}}$ **false**, uma contradição. Logo, nossa suposição da não existência de Δ_j^{i+1} tal que $\Delta_j^{i+1} \cup \{m_0, m_1, \ldots, m_r\}$ seja satisfatível no nível $i + 1$ do modelo é incorreta. Portanto, a conclusão da regra (EXP) é satisfeita. □

3.3 Exemplos

Esta seção apresenta exemplos de aplicações das regras de inferência definidas para o cálculo $\mathcal{CK}_n$ em conjuntos de cláusulas. O Exemplo 4 mostra as aplicações para um conjunto insatisfatível enquanto que o Exemplo 5 mostra para um conjunto satisfatível.

Exemplo 4 Conjuntos de cláusulas iniciais:

$$\Gamma^1 = \{1 : p\} \tag{3.1}$$

$$\Gamma^2 = \{2 : \neg q\} \tag{3.2}$$

$$\Lambda^1 = \{1 : p \Rightarrow \Diamond_1 q\} \tag{3.3}$$

$$\Theta^1 = \{\} \tag{3.4}$$

Inicialmente, os conjuntos de literais para cada nível modal se iniciam vazios. Desta forma, após aplicação de (PROP) em 3.1 e 3.2 temos que:

$$\Delta_1^1 = \{p\}, \Delta_1^2 = \{\neg q\} \tag{3.5}$$

Aplicando (NEG) em 3.3 e 3.5, obtemos:

$$\Delta_1^1 = \{p, \neg p\}, \Delta_2^1 = \{p, \Diamond_1 q\}, \Delta_1^2 = \{\neg q\} \tag{3.6}$$

Observe que agora o conjunto Δ_1^1 possui uma inconsistência, então ele é eliminado da prova:

$$\Delta_1^1 = \{\bot\}, \Delta_2^1 = \{p, \Diamond_1 q\}, \Delta_1^2 = \{\neg q\} \tag{3.7}$$

Finalmente, a aplicação da regra (EXP) em 3.7, nos dá:

$$\Delta_2^1 = \{p, \Diamond_1 q\}, \Delta_1^2 = \{\neg q, q\} \tag{3.8}$$

Ou seja, ao assumirmos que Δ_2^1 é satisfatível, adicionamos inconsistência no próximo nível modal. Portanto, Δ_2^1 não é satisfatível. Como não sobraram conjuntos de literais no nível 1, o conjunto inicial de cláusulas é insatisfatível.

Exemplo 5 Conjuntos de cláusulas iniciais:

$$\Gamma^1 = \{1 : t\} \tag{3.9}$$

$$\Gamma^2 = \{2 : \neg x \vee r\} \tag{3.10}$$

$$\Lambda^1 = \{1 : t \Rightarrow \Diamond_1 x\} \tag{3.11}$$

$$\Theta^1 = \{\} \tag{3.12}$$

Inicialmente, os conjuntos de literais para cada nível modal se iniciam vazios. Desta forma, após aplicação de (PROP) em 3.9 e 3.10 temos que:

$$\Delta_1^1 = \{t\}, \Delta_1^2 = \{\neg x\}, \Delta_2^2 = \{r\} \tag{3.13}$$

Aplicando (NEG) em 3.11 e 3.13, obtemos:

$$\Delta_1^1 = \{t, \neg t\}, \Delta_2^1 = \{t, \Diamond_1 x\}, \Delta_1^2 = \{\neg x\}, \Delta_2^2 = \{r\} \tag{3.14}$$

Como inconsistência foi inserida no conjunto Δ_1^1, fazemos $\Delta_1^1 = \{\bot\}$ e o eliminamos da prova. Finalmente, ao aplicarmos a regra (EXP) em 3.14, obtemos:

$$\Delta_2^1 = \{t, \Diamond_1 x\}, \Delta_1^2 = \{\neg x\}, \Delta_2^2 = \{r, x\} \tag{3.15}$$

Como ainda existem conjuntos de literais em todos os níveis e os literais modais existentes foram satisfeitos, o conjunto de cláusulas inicial é satisfatível.

Observe que, no caso em que o conjunto de cláusulas inicial é satisfatível, os conjuntos de literais gerados que não foram excluídos podem ser combinados para formar um possível modelo que testemunhe essa satisfatibilidade. Um desses modelos possíveis está ilustrado na Figura 3.1.

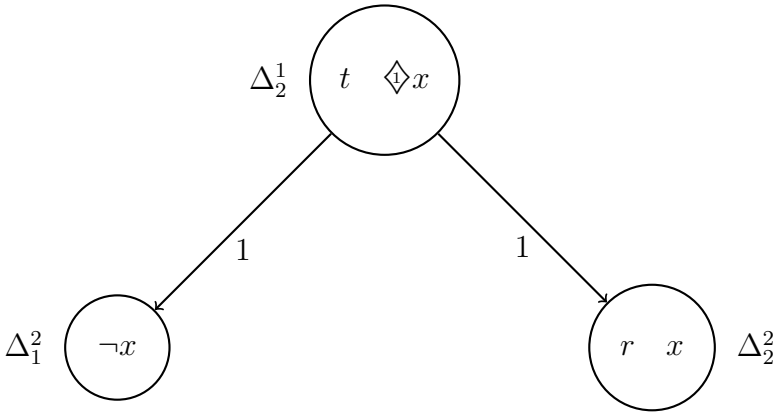

Figura 3.1: Exemplo de modelo que satisfaz o conjuntos de cláusulas do Exemplo 5.

Capítulo 4

Implementação e Avaliação

Uma versão do cálculo apresentado no Capítulo 3 foi implementada em `C` para fins de análise comparativa com o provador KSP implementado em [24], além de reforçar que suas regras estão corretas. Entretanto, algumas alterações práticas foram inseridas às regras originais dado que resultados iniciais, que foram omitidos do trabalho por fugirem do real escopo da proposta, mostraram inviabilidade de implementação. Estas alterações e uma visão geral da implementação encontram-se na próxima seção. Os resultados obtidos pela mesma encontram-se na Seção 4.2.

4.1 Implementação

A entrada da implementação corresponde a um conjunto de fórmulas transformado em um conjunto de cáusulas em $\mathsf{SNF}_{\mathrm{K}}$ pelo provador implementado em [24]. Ou seja, os conjuntos especificados no Capítulo 3 referentes aos conjuntos de cláusulas iniciais, Γ^i, Λ^i e Θ^i, em cada nível modal, são, de fato, a entrada do programa. A saída corresponde a uma resposta binária: *Satisfatível* ou *Insatisfatível* referente ao status semântico da entrada original.

Por uma estratégia de construção, mais relacionada ao controle da implementação que à eficiência da mesma, todas as cláusulas proposicionais são tratadas primeiro, um nível modal por vez. Em seguida, trata-se as cláusulas modais, também em um nível modal de cada vez. Após lidar com todos os conjuntos iniciais de cláusulas, a fase de expansão modal é iniciada.

Os conjuntos de literais definidos no Capítulo 3 (Δ^i) são inicialmente preenchidos pelos literais presentes nas cláusulas proposicionais conforme regra (PROP). Na implementação, estes conjuntos correspondem a uma estrutura organizada em uma tabela de *hash*, dividida em três níveis, conforme Figura 4.1.

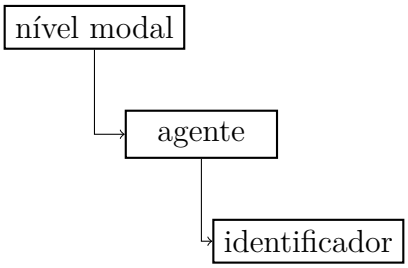

Figura 4.1: Níveis de *hash* dos conjuntos de literais.

A divisão por níveis modais faz sentido já que cada nível é tratado separadamente. Dado que as cláusulas modais são vinculadas a um agente, a separação por agentes também é justificada. Finalmente, um identificador é gerado a cada uma das estruturas para servir de chave primária do último nível de *hash*. Além do identificador, esta estrutura possui outros campos de controle e uma lista de literais. Dado que cada literal é representado por um número inteiro, esta lista também foi implementada como uma tabela *hash*, de apenas um nível indexado pelo inteiro correspondente ao literal, com a finalidade de facilitar a busca por inconsistências (ou seja, a presença de um literal e sua negação).

A execução da primeira fase, a fase proposicional, é a mais custosa, por realizar uma série de duplicações de conjuntos para cada cláusula do conjunto inicial a fim de tratar cada literal corretamente, o correspondente a gerar cada um dos denominadores da regra original. Uma vez que inconsistência é inserida a um conjunto (inserção de um literal l quando $\neg l$ já pertence ao conjunto), este conjunto é excluído do *hash*. Caso haja eliminação de todos os conjuntos gerados no nível modal inicial, o programa pode retornar a resposta *Insatisfatível*, mesmo que não tenha tratados as cláusulas modais. Ao final desta fase, se ainda restarem conjuntos, o tratamento das cláusulas modais é iniciado.

As regras relacionadas às cláusulas modais, (NEG) e (POS), são bastante similares, a principal diferença é semântica já que sintaticamente, diferenciam-se apenas nos operadores. Nesta etapa, para cada cláusula, os conjuntos de literais deveriam ser submetidos a novas duplicações para que os denominadores de suas regras fossem considerados corretamente. Entretanto, esta abordagem se mostrou custosa em termos de espaço e, portanto, uma nova estratégia foi utilizada, sem alterar a lógica das regras definidas. Ao invés de serem decompostas e inseridas aos conjuntos de literais gerados na fase proposicional, as cláusulas modais compõem uma nova estrutura, também organizada em uma tabela *hash*, agora com quatro níveis, conforme ilustrada na Figura 4.2.

O *tipo* faz referência ao tipo da cláusula, se é positiva ou negativa. A *cabeça* de uma cláusula modal corresponde ao literal do lado esquerdo da implicação. Para cada um destes literais, o último nível da tabela de *hash*, *literais*, contém todos os literais que

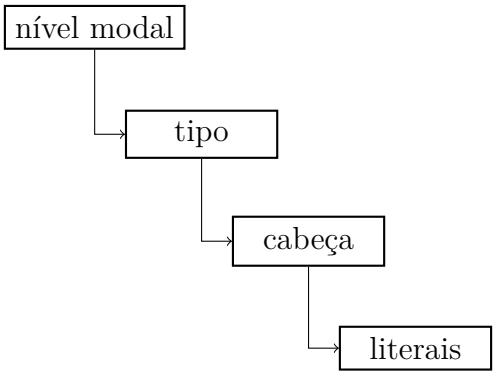

Figura 4.2: Níveis de *hash* dos literais modais.

aparecem do lado direito de cláusulas modais. Dessa forma, se for possível inserir, sem inconsistência, a negação da cabeça da cláusula em um conjunto de literais, para *aquele* conjunto, não será necessário realizar a expansão dos literais em escopo modal vinculados àquela cabeça.

Essa checagem é realizada já na fase de expansão. Como as cláusulas modais e os conjuntos de literais estão separados por níveis modais, é fácil realizar a verificação, em cada conjunto, de quais literais modais devem ser satisfeitos no próximo nível, para que o conjunto que está sendo analisado seja satisfatível. Ou seja, a implementação da regra [EXP] corresponde a uma passagem em todos os conjuntos de literais do próximo nível, na tentativa de satisfazer cada literal modal negativo em *pelo menos um* conjunto e os literais modais positivos em *todos* os conjuntos. Se não for possível satisfazer um dos literais modais, o conjunto do nível anterior é eliminado do *hash*.

Ao final da expansão, basta verificar se o *hash* está vazio. Em caso positivo, o conjunto inicial de cláusulas é *Insatisfatível*. Caso contrário, a resposta dada será *Satisfatível* restando, portanto, informações suficientes para se obter um possível modelo que o satisfaz.

4.2 Resultados Experimentais

A implementação foi testada em duas etapas. A etapa inicial de testes se concentrou em encontrar e corrigir erros de implementação e verificar se as regras do cálculo estavam sendo cumpridas conforme especificado. Nesta fase, as diferentes fases de execução foram testadas separadamente, dado sua interseção quase nula.

Uma vez que os erros encontrados foram corrigidos, a etapa seguinte de experimentação foi realizada a partir de famílias de *benchmarks*. Com enfoque avaliativo para o benchmark *tptp* [29], com 44 conjuntos de fórmulas proposicionais, e enfoque comparativo para o

benchmark *lwb* [3], com 378 conjuntos de fórmulas modais de 9 famílias diferentes, onde cada família é constituída de 21 fórmulas satisfatíveis e 21 insatisfatíveis.

Os experimentos foram executados em um computador com sistema operacional Ubuntu 14.04 (GNU/Linux 3.19.0-30-generic x86_64), processador Intel® Xeon® Processor E5-2620 v3, que possui frequência de *clock* 2.4GHz e 15M de memória *cache* e memória RAM de 64GiB.

O enfoque avaliativo realizado em famílias de fórmulas proposicionais foi executado como uma segunda tarefa de garantia da correta implementação da regra (PROP). Por se tratar de uma das fases mais custosas, é fundamental que se esteja tão certo quanto possível do correto funcionamento da mesma. A implementação do cálculo foi executada, para os conjuntos de fórmulas proposicionais do benchmark *tptp*, com um limite de tempo de 60 segundos para, além de testar a correção, avaliar a eficiência da implementação. O tempo dado corresponde ao necessário para um provador no estado da arte resolver todos os 44 conjuntos de fórmulas.

O resultado obtido superou as expectativas para essa fase já que, dos 44 conjuntos de fórmulas testados, foram obtidas respostas corretas para 40 deles, conforme Tabela 4.1, ou seja, apenas quatro conjuntos deste benchmark extrapolaram o limite de tempo estabelecido. Além disso, o programa produziu resposta correta para todas as entradas e, de todos os tempos coletados, apenas três acima de 0.1 segundos. Os resultados obtidos estão expressos com mais detalhes na Tabela A.2 do Apêndice A.

Tabela 4.1: Conjuntos de fórmulas proposicionais testadas.

	Satisfatível	Insatisfatível	Total
$\mathcal{C}\mathsf{K}_n$	6	34	40
Total	7	37	44

O enfoque comparativo dado à avaliação do benchmark de fórmulas modais foi realizado com o objetivo de se obter uma análise crítica da implementação. A comparação foi realizada com o provador implementado em [24], chamado KSP, que utiliza como método de prova apenas a resolução clausal. Como o cálculo apresentado neste trabalho é baseado em tableaux (mesmo utilizando cláusulas como numeradores das regras), já era esperado que o desempenho obtido fosse consideravelmente inferior ao obtido pelo KSP, já que as regras no tableaux são definidas com base em duplicações de conjuntos de fórmulas e expansões dos níveis modais. Além disso, nessa implementação em específico, toda informação semântica possível das cláusulas de entrada, em cada nível modal, é gerada. Como mencionado antes, essa informação pode, por exemplo, ser útil em uma possível geração automática de modelo que satisfaz estas cláusulas, mas contribui significativa-

mente para uma eficiência menor, dado que a busca é realizada em largura, ao invés de em profundidade, como em implementações padrões de tableaux.

Ainda que um resultado inferior já fosse esperado, o resultado obtido foi abaixo das expectativas. Inicialmente, a comparação foi executada com um limite de tempo de 60 segundos, tempo necessário para que um provador no estado da arte resolva cerca de 70% das fórmulas do benchmark. Neste limite de tempo, dos 378 conjuntos de fórmulas pertencentes ao benchmark *lwb*, a implementação do cálculo $\mathcal{CK}_n$ obteve, embora todas corretas, apenas 73 respostas enquanto que o provador KSP obteve 330 respostas, conforme Tabela 4.2.

Tabela 4.2: Famílias de fórmulas modais testadas.

	Satisfatível	Insatisfatível	Total
$\mathcal{CK}_n$	19	54	73
KSP	164	166	330
Total	189	189	378

Como este benchmark é dividido em famílias de fórmulas, é válida a realização de uma análise por família, como forma de identificar pontos fracos e fortes de ambas as implementações. Cada família é composta por 42 fórmulas: 21 fórmulas satisfatíveis e 21 insatisfatíveis. A Tabela 4.3 mostra quantas respostas foram obtidas por cada provador para cada família de fórmulas.

Tabela 4.3: Separação por famílias do benchmark *lwb*.

		Satisfatível	Insatisfatível	Total
branch	$\mathcal{CK}_n$	1	1	2
	KSP	12	14	26
d4	$\mathcal{CK}_n$	1	1	2
	KSP	21	21	42
dum	$\mathcal{CK}_n$	9	5	14
	KSP	21	21	42
grz	$\mathcal{CK}_n$	1	21	22
	KSP	21	21	42
lin	$\mathcal{CK}_n$	1	21	22
	KSP	21	21	42
path	$\mathcal{CK}_n$	1	1	2
	KSP	21	21	42
ph	$\mathcal{CK}_n$	3	3	6
	KSP	5	5	10
poly	$\mathcal{CK}_n$	2	1	3
	KSP	21	21	42
t4p	$\mathcal{CK}_n$	0	0	0
	KSP	21	21	42

Pela Tabela 4.2 já era possível notar que o provador KSP obteve um comportamento equilibrado com relação aos conjuntos de fórmulas tanto satisfatíveis, quanto insatisfatíveis. O que não ocorreu no caso da implementação do cálculo $\mathcal{C}_{\mathsf{K}_n}$, que apresentou melhores resultados para os conjuntos de fórmulas insatisfatíveis. Este comportamento também pode ser observado na Tabela 4.3, onde vê-se que o KSP obteve uma quantidade regular de respostas por família de fórmulas. Já para a implementação do cálculo proposto neste trabalho, para as famílias grz e lin, por exemplo, apesar da implementação ter sido capaz de obter a resposta para os 21 conjuntos insatisfatíveis, para os conjuntos satisfatíveis, obteve apenas uma resposta.

Por outro lado, para a família t4p, a implementação do cálculo não foi capaz de obter resposta nem para as fórmulas satisfatíveis, nem para as fórmulas insatisfatíveis, no limite de tempo estabelecido. Para esta família em específico, foi realizada uma execução para os primeiros conjuntos de fórmulas, sem limite de tempo, para verificar se a resposta obtida estaria correta. Após 38 minutos de execução para o conjunto de fórmulas satisfatíveis e 2 horas e 46 minutos para o conjunto de fórmulas insatisfatíveis, a implementação obteve a resposta correta. A investigação de uma justificativa que explique porque a implementação é mais eficiente em determinadas famílias é deixado como proposta de trabalho futuro.

É fácil imaginar que em uma comparação entre os tempos de execução dos dois provadores, o KSP tenha se mostrado superior. De qualquer forma, a Tabela A.1 do Apêndice A mostra as médias dos tempos obtidos para os conjuntos de fórmulas de cada família que ambos provadores foram capazes de resolver.

Capítulo 5

Considerações Finais

Quando se trata de lógica, a combinação de métodos de prova é de grande interesse teórico por apresentar diversas vantagens quando comparada ao uso de métodos isolados. Um exemplo claro é quando consideramos sistemas computacionais descritos por combinações de diferentes lógicas modais, para uma tal combinação, o método de prova pode ser obtido pela combinação dos métodos para as linguagens componentes.

Este trabalho apresentou a definição do cálculo tableaux clausal $\mathcal{CK}_n$, que combina a forma normal utilizada produzida para resolução clausal e o método refutacional baseado em tableaux. Isto é, trata-se de um cálculo composto por regras tableaux cujos numeradores são conjuntos de cláusulas na forma $\mathsf{SNF}_{\mathrm{K}}$. Apresentou, também, as provas de correção de todas as regras especificadas, portando, provando que o cálculo como um todo é correto. Além disso, uma visão geral de uma implementação deste cálculo foi mostrada, bem como os dados obtidos nas análises experimentais.

Quando comparado com o provador implementado em [24], a implementação do cálculo $\mathcal{CK}_n$ apresentou um baixo desempenho. Entretando, todas as execuções realizadas chegaram à resposta correta, cumprindo os objetivos de testar o cálculo e extrair informações dos conjuntos de cláusulas de entrada, informações que podem ser úteis como ferramenta para auxiliar a efetiva combinação entre os cálculos baseados em tableaux e resolução. Portanto, a implementação pode ser considerada bem sucedida. Essa extração é o ponto forte do cálculo e da implementação, porque uma grande quantidade de informação é gerada a partir dos conjuntos de entrada. Estas informações contêm peculiaridades do conjunto de fórmulas inicial que podem auxiliar na elaboração de estratégias para decidir o melhor método de prova e reduzir o custo de se encontrar uma prova.

5.1 Trabalhos Futuros

Apesar da implementação do cálculo proposto ter sido bem sucedida, ainda existe muito espaço para melhora. O desempenho observado mostrou que a implementação requer esforços práticos com o objetivo de melhorar a eficiência da execução, este é um dos possíveis trabalhos futuros. Seguem esta e outras sugestões de trabalhos futuros:

- Criar uma maior interseção entre as fases do programa para analisar possíveis ganhos de performance. Também com esse objetivo, pode-se focar esforços em análise dos algoritmos implementados;
- Utilizar a informação construída ao longo da execução do cálculo para extrair um modelo no caso em que o conjunto de cláusulas inicial for satisfatível, ou seja, implementar um gerador automático de modelos com base nos conjuntos de literais que não foram excluídos;
- Analisar as peculiaridades das famílias de fórmulas do benchmark *lwb* e, futuramente, outros benchmarks, para classificar o melhor cenário em que o cálculo proposto atua;
- Utilizar as informações geradas pela implementação como ferramenta de suporte teórico para auxiliar nos avanços de conhecimento das fórmulas pertencentes à lógica modal.

Referências

[1] C. Areces, R. Gennari, J. Heguiabehere, and M. D. Rijke. Tree-based heuristics in modal theorem proving. In *Proc. ECAI 2000*, pages 199–203. IOS Press, 2000. 16

[2] C. Areces and J. Heguiabehere. HyLoRes: A hybrid logic prover, Sept. 18 2002. 2

[3] P. Balsiger, A. Heuerding, and S. Schwendimann. A benchmark method for the propositional modal logics K, KT, S4. *J. Autom. Reasoning*, 24(3):297–317, 2000. 31

[4] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Number 53 in Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2001. 16

[5] M. Bratman. *Intention, plans, and practical reason*. Center for the Study of Language and Information, California, USA, 1987. 1

[6] B. F. Chellas. *Modal logic — an introduction*. Press Syndicate of the University of Cambridge, London, 1980. 4, 15

[7] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd annual ACM STOC*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM. 2

[8] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, Cambridge, MA, USA, 1995. 1

[9] M. Fitting. Destructive modal resolution, 1989. 4

[10] M. Fitting and R. L. Mendelsohn. *First order modal logic*. Kluwer Academic Plubishers, New York, 1998. 16, 18, 19, 23

[11] D. Gabbay, A. Kurucz, F. Wolter, and M. Zakharyaschev. Many-dimensional modal logics: theory and applications. *Studies in logic and the foundations of Mathematics*, 148, 2003. 2

[12] H. J. Gensler. *Introduction to logic*. Routledge, 2012. 5, 6

[13] R. Goré and L. A. Nguyen. Clausal tableaux for multimodal logics of belief. *Fundamenta Informaticae*, 94:21–40, 2009. 19, 20

[14] D. Götzmann, M. Kaminski, and G. Smolka. Spartacus: A tableau prover for hybrid logic. *Electr. Notes Theor. Comput. Sci.*, 262:127–139, 2010. 2

[15] B. T. Hailpern. *Verifying Concurrent Processes Using Temporal Logic*, volume 129 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin/New York, 1982. 1

[16] J. Halpern, Z. Manna, and B. Moszkowski. A Hardware Semantics Based on Temporal Intervals. *Lecture Notes in Computer Science*, 154:278–291, 1983. 1

[17] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54(3):319–379, Apr. 1992. 2

[18] J. Hudelmaier. Improved decision procedures for the modal logics K, T and S4. In *International Workshop on Computer Science Logic*, pages 320–334. Springer, 1995. 20

[19] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge university press, 2004. 5

[20] S. C. Kleene. *Mathematical Logic*. Wiley and Sons, 1968. 17

[21] R. E. Ladner. The computational complexity of provability in systems of modal propositional logic. *SIAM J. Comput*, 6, 1977. 2

[22] G. Mints. Gentzen-type systems and resolution rules Part I: Propositional logic. In *COLOG-88*, pages 198–231. Springer, 1990. 20

[23] C. Nalon, U. Hustadt, and C. Dixon. A modal-layered resolution calculus for K. In H. D. Nivelle, editor, *Automated Reasoning with Analytic Tableaux and Related Methods - 24th International Conference, TABLEAUX 2015, Wrocław, Poland, September 21-24, 2015. Proceedings*, volume 9323 of *Lecture Notes in Computer Science*, pages 185–200. Springer, 2015. 2, 17, 18

[24] C. Nalon, U. Hustadt, and C. Dixon. Ksp: A resolution-based prover for multimodal k. In N. Olivetti and A. Tiwari, editors, *Automated Reasoning: 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 – July 2, 2016, Proceedings*, pages 406–415, Cham, 2016. Springer International Publishing. 2, 28, 31, 34

[25] L. A. Nguyen. A new space bound for the modal logics K4, KD4 and S4. In M. Kutyłowski, L. Pacholski, and T. Wierzbicki, editors, *Mathematical Foundations of Computer Science 1999: 24th International Symposium, MFCS'99 Szklarska Poręba, Poland, September 6–10,1999 Proceedings*, pages 321–331. Springer Berlin Heidelberg, Berlin, Heidelberg, 1999. 20

[26] A. S. Rao and M. P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In R. Fikes and E. Sandewall, editors, *Proceedings of Knowledge Representation and Reasoning (KR&R-91)*, pages 473–484, Cambridge, MA, USA, Apr. 1991. Morgan-Kaufmann. 1

[27] W. Rautenberg. Modal tableau calculi and interpolation. *Journal of Philosophical Logic*, 12(4):403–423, 1983. 18

[28] S. Schulz. The E theorem prover, 2013. `http://wwwlehre.dhbw-stuttgart.de/~sschulz/E/E.html`. 1

[29] G. Sutcliffe. The TPTP problem library and associated infrastructure. *Journal of Automated Reasoning*, 43(4):337–362, 2009. 30

[30] The SPASS Team. Automation of logic: Spass, 2010. http://www.spass-prover.org/. 1

[31] A. Voronkov. Vampire. http://www.vprover.org/index.cgi. 1

Apêndice A

Tabelas de Resultados

Tabela A.1: Média de tempos para cada família de fórmulas modais sobre os conjuntos que ambos os provadores foram capazes de resolver.

Família	Média de tempo (segundos)			
	KSP		$\mathcal{CK}_n$	
	Sat	Insat	Sat	Insat
branch	0.00	0.00	0.07	0.05
d4	0.00	0.00	0.88	0.12
dum	0.00	0.00	5.59	5.326
grz	0.00	0.00	7.32	36.73
lin	0.00	0.00	0.01	0.01
path	0.00	0.00	9.07	0.02
ph	0.01	0.00	0.14	11.73
poly	0.00	0.00	2.31	0.02
t4p	-	-	-	-

Tabela A.2: Respostas obtidas na execução da implementação para o benchmark *tptp*.

Satisfatível	Insatisfatível	Não Resolvido
PUZ016-2	GRA001-1	MSC007-1
SYN086-1	HWV003-3	PUZ015-2
SYN087-1	LCL181-2	PUZ016-2
SYN092-1	LCL230-2	SYN302-1
	PUZ004-1	
	PUZ009-1	
	PUZ013-1	
	PUZ014-1	
	PUZ030-2	
	PUZ033-1	
	SYN001-1	
	SYN003-1	
	SYN004-1	
	SYN008-1	
	SYN010-1	
	SYN011-1	
	SYN028-1	
	SYN029-1	
	SYN030-1	
	SYN032-1	
	SYN040-1	
	SYN041-1	
	SYN044-1	
	SYN045-1	
	SYN046-1	
	SYN047-1	
	SYN085-1	
	SYN089-1	
	SYN090-1	
	SYN093-1	
	SYN097-1	
	SYN098-1	
	SYN915-1	
Mais de 0.1 segundos		
tempo (segundos)	tempo (segundos)	
NUM285-1 0.37	SYN094-1 1.98	
SYN091-1 1.12		