UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO – BCC

MIGUEL ALFREDO NUNES

FUSÃO DE LÓGICAS MODAIS NO ASSISTENTE DE PROVAS COQ

JOINVILLE
2023

MIGUEL ALFREDO NUNES

FUSÃO DE LÓGICAS MODAIS NO ASSISTENTE DE PROVAS COQ

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Orientadora: Karina Girardi Roggia
Coorientador: Paulo Henrique Torrens

JOINVILLE
2023

MIGUEL ALFREDO NUNES

FUSÃO DE LÓGICAS MODAIS NO ASSISTENTE DE PROVAS COQ

Trabalho de conclusão de curso submetido à Universidade do Estado de Santa Catarina como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação

Orientadora: Karina Girardi Roggia
Coorientador: Paulo Henrique Torrens

BANCA EXAMINADORA:

Orientadora:

Dra. Karina Girardi Roggia
UDESC

Coorientador:

Me. Paulo Henrique Torrens
University of Kent

Membros:

Dr. Cristiano Damiani Vasconcellos
UDESC

Dr. Kariston Pereira
UDESC

Joinville, Junho de 2023

AGRADECIMENTOS

Sinto que poderia escrever dez vezes mais que escrevi aqui, mas espero que o pouco que escrevi seja o suficiente para expressar minha gratidão.

Primeiro devo agradecer aos muitos amigos que fiz durante a graduação, cuja amizade foi e ainda é muito importante para mim, pela sua amizade e carinho imensuráveis, por sua companhia durante os dois anos de isolamento social, por todas as conversas, gargalhadas, jogatinas até tarde da noite, ajuda nas matérias, almoços juntos, conversas sem começo nem fim no centro acadêmico, noites bebendo, idas ao Kelson após os colóquios e por todos os inesquecíveis e incontáveis bons momentos que passei com vocês. Vocês tornaram meu tempo na Udesc melhor do que eu jamais poderia imaginar e me ajudaram quando mais precisava.

Agradeço aos professores e organizadores do SPLogIC, por me escolherem para participar e por tudo que me ensinaram, e a todos os amigos que fiz durante o evento por toda sua amizade, por terem tornado aquelas duas semanas em Campinas ainda mais maravilhosas que já eram e por tudo que aprendi com vocês, durante e depois do evento.

Agradeço muito aos meus amigos do grupo Função, por todas as conversas nos corredores e no grupo de Telegram, todos os colóquios, todos os cafés, todas as partidas de truco e poker, todas as idas ao Clover, todas as videochamadas durante o isolamento social que me trouxeram imensa alegria, todos os momentos no nosso apertado e bagunçado laboratório, todos os bons momentos que passei junto de vocês e tudo que eu aprendi com vocês. Vocês são incríveis, nunca teria imaginado que encontraria um grupo de amigos como vocês.

Também devo agradecer a minha banca, Professores Cristiano e Kariston, pelo interesse no meu trabalho e no esforço para ler e avaliar ele, por tudo que aprendi com vocês e por todo o tempo que cada um dedicou a mim, tanto como parte deste trabalho quanto antes de chegar aqui.

Agradeço ao meu coorientador, Professor Paulo, por todas as tardes e noites que me fez companhia no laboratório, por todo o tempo que dedicou para me ajudar com problemas da IC e do TCC e por tudo que me ensinou, do seu jeito bem peculiar.

Devo agradecer à minha orientadora, Professora Karina, por me orientar e guiar de pouquinho em pouquinho até onde estou hoje, por todas as folhas de papel que rabiscou para me explicar o que quer que fosse que eu lhe perguntasse, por todo o imensurável tempo que dedicou a mim, por sempre arranjar um momento para me atender independentemente do quão atarefada estivesse ou do que quer que eu quisesse, por todos os cafés, por todos os almoços, todos os abraços, todo o seu carinho e afeto, por me perdoar após ter pronunciado errado seu sobrenome no vídeo do SIC, por ser uma grande inspiração para mim e por, para minha contínua surpresa, não ter enlouquecido com tudo que aconteceu nesses tantos anos que nos conhecemos. Obrigado por tudo. Se não fosse por você, eu nunca teria chegado até aqui.

Por fim, devo agradecer a Udesc por tudo que aprendi e vivenciei aqui, apesar dos pesares, e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq pela bolsa de modalidade ITC-A de número 409707/2022-8 que apoiou este trabalho.

"*We loved our country as much as they; we went courageously into every action; but also we distinguished the false from true, we had suddenly learned to see. And we saw that there was nothing of their world left. We were all at once terribly alone; and alone we must see it through*" (Enrich Maria Remarque – All Quiet on The Western Front, [1928])

RESUMO

É possível modelar componentes de sistemas de *software* em sistemas lógicos, porém sistemas de *software* complexos não são facilmente modelados em linguagens simples. Para obter uma linguagem lógica que seja capaz de expressar propriedades complexas pode-se combinar sistemas lógicos mais simples. Uma das formas de combinar lógicas é a fusão de lógicas modais. Porém, qualquer tipo de combinação de lógicas é uma tarefa complexa, que requer que sistemas lógicos sejam tratados como objetos matemáticos para então serem combinados entre si. Modelar sistemas lógicos em assistentes de provas pode facilitar o processo de combinação de lógicas pois, em assistentes de provas, sistemas lógicos já são tratados como objetos matemáticos. Mais ainda, assistentes como *Coq* e *Lean* possuem grande gama de ferramentas de automação que facilitam o processo de desenvolvimento de provas. Este trabalho propõe modelar de forma paramétrica lógicas multimodais resultantes da fusão de sistemas lógicos, no assistentes de provas Coq, e provar a preservação de algumas propriedades de lógicas pela operação de fusão. Após uma revisão da literatura relevante, foi modelado um estudo de caso de fusões de lógicas modais no Coq e foi implementado de forma paramétrica a fusão de lógicas modais, onde foi possível modelar, de forma paramétrica, a fusão de sistemas sintáticos de lógicas modais quaisquer.

Palavras-chave: Coq. Assistentes de Provas. Lógicas Multimodais. Fusão de Lógicas.

ABSTRACT

It is possible to model software components in logical systems, however complex software systems are not easily modelled in simple logical languages. To obtain a logical language capable of expressing complex properties, combining simpler logical systems is a possibility. One of the ways to combine logical systems is by means of fusions of modal logics. However, any kind of combination of logics is a complex task, that requires logical system to be treated as mathematical objects for they then be able to be combined. Modelling logical systems in proof assistants can make the combination process easier, since, in proof assistants, logical systems are already treated as mathematical objects. Moreover, proof assistants such as *Coq* and *Lean* have a large amount of automation tools that help in the process of developing proofs. As such, this work proposes to parametrically model multimodal logicas resulting from the fusion of logical systems, in the Coq proof assistant, and to prove the preservation of some properties of logics by the operation of fusion. After a review of the relevant literature, a case study of fusion of modal logics in Coq was modeled and the fusion of modal logics was was parametrically implemented, where it was possible to parametrically model the fusion of syntactic systems of any modal logics.

Keywords: Coq. Proof Assistants. Multimodal Logics. Fusion of Logics.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Lógica modal é uma família de lógicas não clássicas com diversas aplicações, tanto na filosofia quanto na computação e na matemática. A primeira formulação da lógica modal surge dentro da filosofia com os trabalhos de Lewis, de acordo com Gabbay (2003). Lewis definiu um conjunto de axiomas com modalidades para representar o conceito de implicação de forma mais próxima àquela da linguagem natural, porém sua interpretação filosófica e seus axiomas para implicação foram substituídos por outras interpretações e conjuntos de axiomas voltados à matemática, como é o caso de Gödel (GÖDEL, 1986), que lidou com a representabilidade da lógica intuicionista dentro da lógica clássica. Apesar disso, dois dos sistemas semânticos definidos por Lewis, os sistemas S4 e S5, ainda são estudados até hoje.

Apenas com os trabalhos de Carnap (CARNAP, 1942) e Kripke (KRIPKE, 1959; KRIPKE, 1963) que uma interpretação semântica da lógica modal foi definida. Carnap definiu uma semântica baseada em descrições de estados, porém, como apresentado por Zalta (1995), a maneira que Carnap definiu a semântica faz com que repetições de modalidades não tenham efeito na interpretação semântica da fórmula. A semântica definida por Kripke é baseada no conceito de mundos possíveis e relações entre mundos, essa é a apresentação semântica mais utilizada atualmente, de acordo com Zalta (1995).

Ainda nos primeiros anos de estudo da lógica modal, outras interpretações de modalidades surgiram, por exemplo, com os trabalhos de Prior (PRIOR, 1957; PRIOR, 1967), que estudou como representar o conceito de temporalidade dentro de linguagens modais e com os trabalhos de von Wright e Hintikka (WRIGHT, 1951; HINTIKKA, 1962), que estudaram como representar os conceito de conhecimento e crença dentro de linguagens modais. Ambas interpretações de lógica modal têm importância na computação: a lógica temporal é muito usada como um formalismo para especificar e verificar corretude de programas, como inicialmente proposto por Pnueli (1977) e mais recentemente estudado por Lamport (2002); já a lógica epistêmica é usada como um formalismo para especificar sistemas multiagentes e sistemas distribuídos, como apresentado em Fagin et al. (1995).

Linguagens modais já eram ferramentas bem estabelecidas quando Thomason (1984) apresentou uma formalização de uma lógica que continha dois operadores modais distintos. Apesar de não ser o primeiro trabalho a tratar deste assunto (o primeiro sendo Fitting (1969) de acordo com Roggia (2012)), o método de Thomason foi o primeiro método genérico de combinações de lógicas, segundo Carnielli e Coniglio (2020), já que o autor apresentou um algoritmo para combinar as linguagens lógicas descritas de forma sintática e semântica.

Paralelo aos desenvolvimentos em lógica modal, a teoria de tipos surgiu como uma alternativa à teoria de conjuntos como uma fundação para a matemática. Inicialmente proposta por Russell (RUSSELL, 1903) como tentativa de lidar com paradoxos da teoria de conjuntos, a teoria de tipos foi adotada pela ciência da computação desde os primórdios da mesma, com trabalhos como Church (1940) que formulou a teoria de tipos com a sintaxe do Cálculo-λ, Curry

e Feys (1958) que apontaram uma correspondência entre axiomas de um fragmento de lógica proposicional e combinadores do Cálculo-λ, e Howard (1980) que generalizou a correspondência de Curry para toda a lógica intuicionista e todo o Cálculo-λ, dando surgimento ao que hoje é chamado de Correspondência de Curry-Howard.

Estes e muitos outros avanços em computação e teoria de tipos proporcionaram, durante os anos de 1960 e 1970, o surgimento de softwares chamados assistentes de provas (HARRISON; URBAN; WIEDIJK, 2014). Estes são sistemas computacionais que permitem usuários definirem, raciocinarem e provarem propriedades sobre teorias e objetos matemáticos, segundo Geuvers (2009). O assistente Automath (BRUIJN, 1980) foi, segundo Harrison, Urban e Wiedijk (2014), o primeiro assistente que usou a Correspondência de Curry-Howard para codificar provas, algo que é feito até os dias de hoje.

Com avanços na teoria de tipos provenientes de trabalhos como os de Martin-Löf (MARTIN-LÖF, 1975; MARTIN-LÖF; SAMBIN, 1984) e sua Teoria de Tipos Intuicionista se tornou possível o desenvolvimento de assistentes de provas mais complexos, como Coq, Lean e Isabelle. Destes sistemas, é de interesse deste trabalho o Coq, que é um assistente desenvolvido pelo instituto francês INRIA desde 1984 e é baseado em um sistema de tipos chamado de Cálculo de Construções Indutivas (TEAM, 2022).

O Coq é comumente usado como uma ferramenta para formalizar e verificar sistemas lógicos e computacionais (PAULIN-MOHRING, 2012). Diversos resultados significativos foram obtidos com Coq, dentre eles podem ser destacados: a certificação de um compilador para a linguagem C (LEROY, 2021), a verificação do *kernel* do sistema operacional seL4 (KLEIN et al., 2010) e uma formalização de uma prova do Teorema das 4 Cores (GONTHIER, 2005).

Com isso, a proposta deste trabalho é continuar a biblioteca de lógica modal em Coq desenvolvida em Silveira (2020) e Silveira et al. (2022). A biblioteca será expandida para representar sistemas lógicos modais resultantes da fusão de sistemas modais mais simples, o código desenvolvido está disponível em um repositório do GitHub no endereço <https://github.com/funcao/LML/tree/Fusao>.

1.1 OBJETIVO GERAL

O objetivo geral deste trabalho é modelar, no assistente Coq, sistemas de lógicas multimodais resultantes da fusão de lógicas modais mais simples, preservando propriedades das lógicas combinadas.

1.2 OBJETIVOS ESPECÍFICOS

Com base no objetivo geral, os seguintes objetivos específicos são definidos:

1. Estudar os principais conceitos de combinações de lógicas, em especial, a fusão;
2. Realizar um estudo de caso de fusões de lógicas modais no Coq;

3. Modelar, de forma paramétrica, sistemas de lógicas multimodais resultantes de fusão de lógicas modais em Coq.

1.3 TRABALHOS RELACIONADOS

A partir de um levantamento da literatura, foram identificados quatro trabalhos semelhantes ao presente trabalho, estes são: Benzmüller (2010) onde o autor apresenta uma modelagem de lógica modal normal com quantificadores, lógica intuicionista, lógica de controle de acesso e lógica de raciocínio espacial em Cálculo-λ Simplesmente Tipado, apresentando também como é possível combinar as lógicas descritas, algumas derivações dentro das lógicas combinadas, algumas propriedades da lógica modal quantificada e uma implementação dos sistemas lógicos discutidos em provadores automáticos de teoremas e assistentes de provas, em específico, em LEO-II[1], TPS[2] e Isabelle[3]. Fuenmayor e Benzmüller (2019) onde os autores utilizam o assistente de provas Isabelle para modelar, por meio de *Shallow Embeddings*[4], linguagens multimodais resultantes de combinações de linguagens modais mais simples, em específico, modelam uma lógica diádica deôntica (um tipo de lógica de obrigatoriedade) com modalidades aléticas e quantificadores e uma semântica bidimensional para a lógica, descrevem exemplos de problemas em linguagem natural e um problema ético que podem ser expressos dentro do sistema lógico modelado. Lescanne e Puisségur (2007) onde os autores utilizam o assistente de provas Coq para modelar a lógica dinâmica do conhecimento comum, que é uma lógica resultante da combinação da lógica dinâmica com a lógica de conhecimento comum (um tipo de lógica epistêmica), descrevem um sistema axiomático de Hilbert para a lógica e apresentam como exemplo de aplicação o problema das crianças enlameadas[5]. Rabe (2017), onde o autor apresenta uma modelagem de diversos sistemas lógicos dentro de uma linguagem de representação de fatos matemáticos baseada em teoria de tipos chamada $\mathrm{M_{MT}}$, lógicas são descritas sintática e semanticamente e uma apresentação categorial e em teoria de tipos de sistemas lógicos é dada, é descrito o conceito de combinação de lógicas a partir de uma perspectiva categorial e são dados alguns exemplos de lógicas combinadas dentro da linguagem descrita.

Os trabalhos levantados não tratam de algum método específico de combinação de lógicas, porém todos exceto Rabe (2017) apresentam combinações de lógicas modais por um método semelhante à fusão, descrita no Capítulo 3, e apenas Rabe (2017) apresenta combinações de quaisquer duas lógicas, porém usando um formalismo que não foi encontrado em outros trabalhos na literatura, exceto em trabalhos publicados pelo mesmo autor.

---

[1] <https://page.mi.fu-berlin.de/cbenzmueller/leo/>

[2] <https://gtps.math.cmu.edu/tps.html>

[3] <https://isabelle.in.tum.de/>

[4] Veja (AZURAT; PRASETYA, 2002) ou o Capítulo 5 deste trabalho

[5] Descrição e solução do problema em inglês: <https://plato.stanford.edu/entries/dynamic-epistemic/appendix-B-solutions.html#muddy>

1.4 METODOLOGIA

O desenvolvimento deste trabalho se iniciou com um levantamento da literatura sobre lógicas modais, combinações de lógicas e sobre o método de fusão de lógicas modais, após, foi feito um levantamento da literatura sobre teoria de tipos e assistentes de provas, com um foco no assistente Coq. Paralelamente a isso, foram buscados trabalhos relacionados ao presente trabalho.

Com a finalização dessa etapa, foi iniciada a escrita do texto e a implementação de lógicas multimodais no Coq. Inicialmente foi implementado o estudo de caso, para verificar a possibilidade de se atingir os objetivos propostos, então foi implementado o sistema multimodal e a fusão de sistemas modais.

1.5 ESTRUTURA DO TRABALHO

O trabalho está organizado da seguinte forma: no Capítulo 2 é apresentada a lógica modal e multimodal e algumas propriedades de ambas são descritas, no Capítulo 3 é apresentado o conceito de fusão de lógicas modais e alguns resultados importantes referentes à preservação de propriedades, no Capítulo 4 é apresentado o conceito de assistentes de provas e o assistente Coq é descrito com alguns detalhes, no Capítulo 5 é descrita a implementação da biblioteca de lógica modal que serve de base para este trabalho, de uma prova de conceito feita para demonstrar a possibilidade de atingir os objetivos deste trabalho e da modelagem paramétrica de lógicas multimodais resultantes de fusão e, por fim, no Capítulo 6 são apresentadas as conclusões deste trabalho.

2 LÓGICA MODAL

Lógica modal é a lógica que trata do conceito de *modalidades*, estes são operadores lógicos que qualificam verdades sobre proposições, segundo Goldblatt (1993). Por exemplo, podemos afirmar que uma certa sentença *deve* ser verdadeira, *pode* ser verdadeira, *sempre será* verdadeira, dentre muitos outros. De acordo com Zalta (1995) os operadores de necessidade e possibilidade são chamados de "modais" pois eles especificam o modo como as subsequentes proposições devem ser interpretadas. Já Blackburn, Rijke e Venema (2001) afirmam que, se perguntar para três lógicos o que é lógica modal, os três darão respostas diferentes.

Segundo Gabbay (2003), o início do estudo de lógicas modais se dá com os estudos de Lewis (LEWIS, 1918; LEWIS; LANGFORD; LAMPRECHT, 1959), na tentativa dos autores de definir o conceito de *implicação estrita*, esta seria uma implicação cuja interpretação é mais próxima da noção de linguagem natural do que da noção matemática. A implicação matemática, ou booleana (chamadas de implicação material pelo autor), "se φ então ψ" seria substituída por "é necessário que, se φ então ψ", seu objetivo com isso era resolver certos paradoxos que a implicação material traz. Por exemplo, a frase "Se a Lua é feita de queijo, então $2 \times 2 = 4$" é uma implicação material correta, visto que 2×2 de fato é igual a 4, apesar da frase "A Lua é feita de queijo" ser absurda. Já a frase "É necessário que, se a Lua é feita de queijo então $2 \times 2 = 4$" não é uma implicação estrita correta, visto que é fato que $2 \times 2 = 4$ apesar da Lua não ser feita de queijo[1].

Para estudar implicação estrita, Lewis definiu cinco sistemas axiomáticos S1,...,S5 independentemente de outros matemáticos contemporâneos. Destes, os sistemas S4 e S5 tornaram-se amplamente conhecidos dentro da lógica modal, não por causa dos trabalhos de Lewis, mas sim pelos trabalhos de Gödel (1986) e Orlov (1928), em que ambos tentavam interpretar a lógica intuicionista dentro da lógica clássica. O estudo da semântica da lógica modal se iniciou com Carnap (CARNAP, 1942), porém, de acordo com Zalta (1995), havia um erro na definição de Carnap, em específico, repetições de modalidades não afetavam a interpretação semântica de uma fórmula. Por exemplo, a fórmula "Necessariamente necessariamente φ" seria semanticamente equivalente a "Necessariamente φ".

Foi apenas com Kripke (KRIPKE, 1959; KRIPKE, 1963) que se obteve uma formulação definitiva da semântica da lógica modal. A semântica definida (conhecida como semântica de Kripke, ou semântica de Mundos Possíveis) lida com mundos e relações de acessibilidade entre eles. Nessa definição semântica, a validade de fórmulas depende de mundos. Em específico, fórmulas que não envolvem modalidades dependem apenas de um mundo "local", já fórmulas que contêm modalidades dependem dos mundos acessíveis pelo mundo local.

A lógica modal que contém apenas as modalidades duais de necessidade (representada por $\Box$) e possibilidade (representada por $\Diamond$) é chamada de lógica modal alética. Essas não são

[1] Podemos representar essa frase na notação que será apresentada à frente da seguinte forma: $\Box(p \to q)$, onde $p \mapsto$ "A lua é feita de queijo", $q \mapsto$ "$2 \times 2 = 4$".

as únicas modalidades que podem ser expressas em lógicas modais, conceitos como tempo, moralidade e conhecimento também podem ser expressos em uma linguagem modal. Caso uma linguagem modal contenha várias modalidades não duais, ela é chamada de lógica multimodal. Por exemplo uma lógica alética com as modalidades $\Box_1$, $\Box_2$ e $\Box_3$ e suas respectivas duais $\Diamond_1$, $\Diamond_2$ e $\Diamond_3$ ou uma lógica que contém as modalidades $\Diamond$ alética e a modalidade **O** de obrigatoriedade da lógica modal deôntica.

Uma lógica pode ser entendida como uma tripla $\mathcal{L} = \langle \mathsf{A}, \vDash_{\mathsf{A}}, \vdash_{\mathsf{A}} \rangle$ onde A é a linguagem (conjunto de todas as fórmulas bem formadas) de $\mathcal{L}$, $\vDash_{\mathsf{A}}$ é uma relação de consequência semântica definida sobre A e $\vdash_{\mathsf{A}}$ é uma relação de consequência sintática definida sobre A. Podemos omitir o subscrito A por simplicidade. Relações de consequência permitem estabelecer uma noção de consequência (ou derivação) entre conjuntos (possivelmente vazios) de fórmulas e fórmulas, com base em certas regras semânticas ou sintáticas. Está definição é semelhante à definição apresentada por Carnielli et al. (2008).

Neste capítulo, será apresentada a lógica modal normal, de forma sucinta, focando na lógica monomodal alética, que será simplesmente chamada de lógica modal. Na Seção 2.1 será definida a linguagem da lógica modal, na Seção 2.2 será definida a semântica de mundos possíveis e algumas propriedades desta serão descritas, na Seção 2.3 será descrita uma axiomatização para a lógica modal, na Seção 2.4 são apresentados outros sistemas de lógica modal, assim como o conceito de correspondência de frames, na Seção 2.5 serão apresentadas algumas meta propriedades de lógicas modais, na Seção 2.6 será apresentado o conceito de lógicas multimodais, sua linguagem, semântica e axiomatização, na Seção 2.7 serão descritas outras lógicas modais e, por fim, na Seção 2.8 será apresentado o conceito de lógica modal visto como um conjunto de fórmulas.

No que segue, usaremos letras gregas minúsculas $\gamma, \varphi, \psi, \ldots$ para representar fórmulas bem formadas de uma linguagem e letras gregas maiúsculas $\Gamma, \Delta, \Sigma, \ldots$ para representar conjuntos de fórmulas.

2.1 LINGUAGEM

A linguagem LM da lógica modal apresentada é baseada nas definições de Gabbay (2003) e Chellas (1980), onde $\mathbb{P} = \{p_0, p_1, \ldots\}$ é um conjunto contável de átomos, $\top$ é a constante lógica de verdade, $\bot$ é a constante lógica de falsidade e a linguagem é definida sobre a assinatura (conjunto de conectivos) $\mathsf{C} = \{\Box, \Diamond, \neg, \wedge, \vee, \rightarrow\}$. Temos então a seguinte definição:

Definição 1 (Linguagem da Lógica Modal)**.** A *linguagem da lógica modal* é definida indutiva-

mente como o menor conjunto que satisfaz as seguintes regras:

$\top, \bot \in \mathsf{LM}$

$\mathbb{P} \subseteq \mathsf{LM}$

Se $\varphi \in \mathsf{LM}$, então $\circ\varphi \in \mathsf{LM}$, sendo $\circ \in \{\Box, \Diamond, \neg\}$

Se $\varphi, \psi \in \mathsf{LM}$, então $\varphi \circ \psi \in \mathsf{LM}$, sendo $\circ \in \{\wedge, \vee, \rightarrow\}$ ■

Podemos utilizar uma assinatura mais simples para a lógica, onde outros conectivos podem ser representados a partir da seguintes dualidades:

Definição 2 (Dualidades entre Conectivos)**.** Os conectivos apresentados anteriormente respeitam as seguintes dualidades:

$$\begin{array}{lcll} \Diamond\varphi & \stackrel{\text{def}}{=} & \neg\Box\neg\varphi & \text{Dualidade } \Box \text{ e } \Diamond \\ \varphi \rightarrow \psi & \stackrel{\text{def}}{=} & \neg\varphi \vee \psi & \text{Dualidade } \rightarrow \text{ e } \vee \\ \neg\varphi & \stackrel{\text{def}}{=} & \varphi \rightarrow \bot & \text{Dualidade } \neg \text{ e } \bot \\ \varphi \wedge \psi & \stackrel{\text{def}}{=} & \neg(\neg\varphi \vee \neg\psi) & \text{Dualidade } \wedge \text{ e } \vee \end{array}$$ ■

Segundo Chellas (1980), o conjunto de subfórmulas de uma dada fórmula φ é o conjunto de todas as fórmulas ψ que compõe φ. Temos então a seguinte definição:

Definição 3 (Subfórmulas)**.** O conjunto de *subfórmulas* de uma fórmula φ, denotado por $\text{Sub}(\varphi)$, é definido indutivamente por:

$\text{Sub}(p_i) = \{p_i\}, p_i \in \mathbb{P}$

$\text{Sub}(\top) = \{\top\}$

$\text{Sub}(\bot) = \{\bot\}$

$\text{Sub}(\circ\varphi) = \{\circ\varphi\} \cup \text{Sub}(\varphi), \circ \in \{\Box, \Diamond, \neg\}$

$\text{Sub}(\varphi \circ \psi) = \{\varphi \circ \psi\} \cup \text{Sub}(\varphi) \cup \text{Sub}(\psi), \circ \in \{\wedge, \vee, \rightarrow\}$ ■

2.2 SEMÂNTICA

Semanticamente, a lógica modal pode ser apresentada de forma algébrica, como em Roggia (2012) e Blackburn, Rijke e Venema (2001), ou de forma relacional pela Semântica de Mundos Possíveis, como em Chellas (1980), Zalta (1995) e Gabbay (2003). No presente trabalho usaremos a semântica de mundos possíveis. De acordo com Wind (2001), o conceito de mundos possíveis que Kripke definiu vêm de ideias de Leibniz, que distinguia verdades em nosso mundo e verdades em todos os mundos possíveis de serem criados. Nessa linha de pensamento, para alguma proposição ser dita necessariamente verdadeira ela deve ser verdadeira em *todos* os mundos possíveis.

Formalmente, sendo $\mathcal{W}$ um conjunto de mundos e $\mathcal{R}$ uma relação binária definida sobre $\mathcal{W}$ denominada relação de acessibilidade, isto é $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$. Dados $w_0, w_1 \in \mathcal{W}$, a expressão

$w_0 \mathcal{R} w_1$ é lida como "w_0 está relacionado com w_1". Outras interpretações podem chamar $\mathcal{R}$ de relação de alternatividade, relatividade ou vizinhança.

Definição 4 (Frames)**.** *Um frame é um par* $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$, onde $\mathcal{W} \neq \varnothing$ e $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$. ■

Para atribuir valores verdade a fórmulas em mundos, devemos definir outra estrutura chamada de modelo[2], que associa frames a funções de valoração.

Definição 5 (Modelos)**.** *Modelos são pares de frames e funções de valoração* da forma $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$, onde $\mathcal{F}$ é um frame e $\mathcal{V}$ é uma função total binária definida por $\mathcal{V} : \mathbb{P} \to 2^{\mathcal{W}}$, sendo $2^{\mathcal{W}}$ o conjunto das partes de $\mathcal{W}$. Caso não seja necessário explicitar o frame de um modelo, podemos apresentar modelos como $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$. ■

A função $\mathcal{V}$ descreve o conjunto de mundos onde um dado átomo é interpretado como verdadeiro. Por exemplo, sendo $\mathcal{W} = \{w_0, \ldots, w_{10}\}$, $\mathcal{V}(p_0) = \{w_0, w_1, w_{10}\}$ indica que o átomo p_0 é verdadeiro nos mundos w_0, w_1 e w_{10}, já $\mathcal{V}(p_1) = \varnothing$ indica que o átomo p_1 não é verdadeiro em qualquer mundo.

Exemplo 1 (Frames, Modelos e Classes)**.** Sendo $\mathcal{W} = \langle w_0, w_1 \rangle$ um conjunto de mundos e $\mathcal{R} = \{\langle w_0, w_0 \rangle, \langle w_0, w_1 \rangle\}$ uma relação de acessibilidade definida sobre $\mathcal{W}$. Podemos definir um frame $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$ e uma função de valoração $\mathcal{V}$ como $\mathcal{V}(p_0) = \{w_0, w_1\}, \mathcal{V}(p_1) = \{w_1\}, \mathcal{V}(p_i) = \varnothing, i \geq 3$, logo, podemos construir um modelo $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$.

Podemos definir uma classe de frames $\mathfrak{F}^2$ como a classe de todos os frames $\langle \mathcal{W}, \mathcal{R} \rangle$ onde $|\mathcal{W}| = 2$, logo $\mathcal{F} \in \mathfrak{F}^2$. Também podemos definir uma classe de modelos $\mathfrak{M}^2$ como a classe de todos os modelos $\langle \mathcal{F}', \mathcal{V}' \rangle$ onde $\mathcal{F}' \in \mathfrak{F}^2$, logo $\mathcal{M} \in \mathfrak{M}^2$. ■

Podemos então definir valoração de fórmulas em mundos de um dado modelo, como apresentado em Chellas (1980).

Definição 6 (Valoração de Fórmulas)**.** Sendo $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$ e $w_0 \in \mathcal{W}$, denotaremos por $\mathcal{M}, w_0 \Vdash \varphi$ ou $w_0 \Vdash_{\mathcal{M}} \varphi$ o fato que a fórmula φ é válida no mundo w_0 do modelo $\mathcal{M}$ e definiremos por:

$\mathcal{M}, w_0 \Vdash \top$

$\mathcal{M}, w_0 \nVdash \bot$

$\mathcal{M}, w_0 \Vdash p_i$ sse $w_0 \in \mathcal{V}(p_i)$, para $p_i \in \mathbb{P}$

$\mathcal{M}, w_0 \Vdash \neg\varphi$ sse $\mathcal{M}, w_0 \nVdash \varphi$

$\mathcal{M}, w_0 \Vdash \varphi \wedge \psi$ sse $(\mathcal{M}, w_0 \Vdash \varphi$ e $\mathcal{M}, w_0 \Vdash \psi)$

$\mathcal{M}, w_0 \Vdash \varphi \vee \psi$ sse $(\mathcal{M}, w_0 \Vdash \varphi$ ou $\mathcal{M}, w_0 \Vdash \psi)$

$\mathcal{M}, w_0 \Vdash \varphi \to \psi$ sse $(\mathcal{M}, w_0 \nVdash \varphi$ ou $\mathcal{M}, w_0 \Vdash \psi)$

[2] Alguns autores chama essa estrutura de "Modelos de Kripke".

$\mathcal{M}, w_0 \Vdash \Box\varphi$ sse $\forall w_1 \in \mathcal{W}, (w_0 \mathcal{R} w_1 \rightarrow \mathcal{M}, w_1 \Vdash \varphi)$

$\mathcal{M}, w_0 \Vdash \Diamond\varphi$ sse $\exists w_1 \in \mathcal{W}, (w_0 \mathcal{R} w_1 \wedge \mathcal{M}, w_1 \Vdash \varphi)$ ■

Podemos estender a noção de validade de fórmulas para (classes de) frames e modelos, como definido em Wind (2001).

Definição 7 (Validade em Modelo)**.** Uma fórmula φ é dita *válida em um modelo* $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V} \rangle$, denotado por $\mathcal{M} \Vdash \varphi$, se, e somente se, $\forall w \in \mathcal{W},\ \mathcal{M}, w \Vdash \varphi$. ■

Definição 8 (Validade em Classes de Modelos)**.** Uma fórmula φ é dita *válida em uma classe de modelos* $\mathfrak{M}$, denotado por $\mathfrak{M} \Vdash \varphi$, se, e somente se, $\forall \mathcal{M} \in \mathfrak{M},\ \mathcal{M} \Vdash \varphi$. ■

Definição 9 (Validade em Frame)**.** Uma fórmula φ é dita *válida em um frame* $\mathcal{F} = \langle \mathcal{W}, \mathcal{R} \rangle$, denotado por $\mathcal{F} \Vdash \varphi$, se, e somente se, $\forall \mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle,\ \mathcal{M} \Vdash \varphi$. ■

Definição 10 (Validade em Classes de Frames)**.** Uma fórmula φ é dita *válida em uma classe de frames* $\mathfrak{F}$, denotado por $\mathfrak{F} \Vdash \varphi$, se, e somente se, $\forall \mathcal{F} \in \mathfrak{F},\ \mathcal{F} \Vdash \varphi$. ■

As definições de satisfazibilidade de fórmulas em estruturas são análogas as definições de validade, mas com alterações na quantificação, então, por exemplo, uma fórmula é dita satisfeita em um modelo se ela é verdadeira em algum mundo do modelo, já uma fórmula é dita satisfeita numa classe de modelos se é satisfeita em algum modelo da classe. Um conjunto de fórmulas é dito satisfeito/válido em alguma estrutura se toda fórmula deste conjunto é satisfeita/válida nesta estrutura.

Definição 11 (Fórmulas Válidas)**.** Uma *fórmula* φ *é dita válida*, denotado por $\Vdash \varphi$, se, e somente se, $\mathcal{F} \Vdash \varphi$ para qualquer frame $\mathcal{F}$. ■

Por fim, temos a consequência semântica, como descrito em Wind (2001):

Definição 12 (Relação de Consequência Semântica)**.** Sendo $\mathcal{S}$ uma relação definida sobre $2^{\mathsf{LM}} \times \mathsf{LM}$ e sendo $\mathfrak{M}$ uma classe de modelos, $\mathcal{S}$ será dita uma relação de consequência semântica onde, para todo $\Gamma \subseteq \mathsf{LM}$ e $\varphi \in \mathsf{LM}$, $\langle \Gamma, \varphi \rangle \in \mathcal{S}$ se, e somente se, se para todo $\gamma \in \Gamma, \mathcal{M} \Vdash \gamma$, então $\mathcal{M} \Vdash \varphi$, em qualquer modelo $\mathcal{M} \in \mathfrak{M}$. Denotaremos consequências por $\Gamma \vDash \varphi$ ou $\Gamma \vDash_{\mathfrak{M}} \varphi$. Caso $\Gamma = \varnothing$ escreveremos $\vDash \varphi$ ou $\vDash_{\mathfrak{M}} \varphi$ e φ será dita uma tautologia na classe $\mathfrak{M}$. ■

2.3 AXIOMATIZAÇÃO

Existem, na literatura, diversos sistemas dedutivos para a lógica modal, dentre eles sistemas de dedução natural, como é o caso de Wind (2001), sistemas de *tableaux*, como é o caso de Priest (2008) ou sistemas axiomáticos de Hilbert, como é o caso de Chellas (1980), Gabbay (2003) e Zalta (1995). O presente trabalho apresentará um sistema axiomático no modelo de Hilbert.

De acordo com Gabbay (2003), um sistema axiomático de Hilbert é composto de um conjunto possivelmente finito de fórmulas e de um conjunto finito de regras de inferência. Este conjunto de fórmulas é chamado de conjunto de axiomas da lógica em questão e a lógica é dita axiomatizável pelo conjunto.

Uma derivação de uma fórmula φ em um sistema de Hilbert é uma sequência finita de fórmulas $\psi_0, \ldots, \psi_n, \varphi$, onde toda fórmula é uma instância de um axioma ou o resultado da aplicação de regras de inferência em fórmulas anteriores na sequência. Instâncias de fórmulas são obtidas a partir da substituição de proposições em fórmulas, como descrito abaixo:

Definição 13 (Substituição)**.** Sendo φ e ψ fórmulas, a substituição em φ de p_i por ψ, denotada por $\varphi\{p_i \mapsto \psi\}$ é definida como:

$$
\begin{aligned}
&p_i\{p_i \mapsto \psi\} = \psi \\
&p_j\{p_i \mapsto \psi\} = p_j, \text{ caso } i \neq j \\
&(\circ\varphi)\{p_i \mapsto \psi\} = \circ(\varphi\{p_i \mapsto \psi\}), \circ \in \{\Box\ \Diamond, \neg\} \\
&(\varphi_0 \circ \varphi_1)\{p_i \mapsto \psi\} = (\varphi_0\{p_i \mapsto \psi\} \circ \varphi_1\{p_i \mapsto \psi\}), \circ \in \{\wedge, \vee, \rightarrow\}
\end{aligned}
$$

■

Temos o seguinte conjunto de regras de inferência:

Definição 14 (Regras de Inferência)**.** A lógica modal pode ser descrita pelas regras de Necessitação (Nec) e Modus Ponens MP:

$$\frac{\Gamma \vdash \varphi}{\Gamma \vdash \Box\varphi}\ \text{Nec} \qquad\qquad \frac{\Gamma \vdash \varphi \rightarrow \psi \qquad \Gamma \vdash \varphi}{\Gamma \vdash \psi}\ \text{MP}$$

■

E o seguinte conjunto de axiomas, como apresentado em Silveira (2020). É de interesse ressaltar que este conjunto não é mínimo nem máximo, é possível apresentar uma axiomatização com menos axiomas, assim como com mais axiomas:

Definição 15 (Axiomas da Lógica Modal)**.** A lógica modal pode ser descrita pelo seguinte conjunto Λ_{LM} de axiomas:

$$p_0 \rightarrow (p_1 \rightarrow p_0) \tag{Ax1}$$
$$(p_0 \rightarrow (p_1 \rightarrow p_2)) \rightarrow ((p_0 \rightarrow p_1) \rightarrow (p_0 \rightarrow p_2)) \tag{Ax2}$$
$$(\neg p_1 \rightarrow \neg p_0) \rightarrow (p_0 \rightarrow p_1) \tag{Ax3}$$
$$p_0 \rightarrow (p_1 \rightarrow (p_0 \wedge p_1)) \tag{Ax4}$$
$$(p_0 \wedge p_1) \rightarrow p_0 \tag{Ax5}$$
$$(p_0 \wedge p_1) \rightarrow p_1 \tag{Ax6}$$
$$p_0 \rightarrow (p_0 \vee p_1) \tag{Ax7}$$
$$p_1 \rightarrow (p_0 \vee p_1) \tag{Ax8}$$

$$(p_0 \to p_2) \to ((p_1 \to p_2) \to (p_0 \vee p_1) \to p_2) \quad \text{(Ax9)}$$

$$\neg\neg p_0 \to p_0 \quad \text{(Ax10)}$$

$$\Box(p_0 \to p_1) \to (\Box p_0 \to \Box p_1) \quad \text{(K)}$$

$$\Diamond(p_0 \vee p_1) \to (\Diamond p_0 \vee \Diamond p_1) \quad \text{(Possibilidade)} \ \blacksquare$$

Dados dois axiomas φ e ψ, caso $\varphi \vdash \psi$ ou $\psi \vdash \varphi$, φ e ψ são ditos dependentes, caso contrário são ditos independentes. Dois conjuntos de axiomas Φ e Ψ são dito dependentes caso exista $\varphi_i \in \Phi$ e $\psi_i \in \Psi$ onde φ_i e ψ_i são dependentes, e são ditos independentes caso contrário. Dados conjuntos de axiomas Γ e Σ, escreveremos $\Gamma \oplus \Sigma$ para denotar o menor conjunto de axiomas independentes definido sobre $\Gamma \cup \Sigma$, como apresentado em Gabbay (2003). Caso $\Sigma = \{\varphi\}$, escreveremos $\Gamma \oplus \varphi$.

Com essas definições, podemos então apresentar o conceito de axiomatização para a lógica modal, como apresentado por Chellas (1980):

Definição 16 (Axiomatização da Lógica Modal)**.** Uma axiomatização é um par $\langle \Sigma, \mathfrak{R} \rangle$, onde Σ é um conjunto de axiomas e $\mathfrak{R}$ é um conjunto de regras de derivação. Caso Σ seja finito, então a lógica axiomatizada por Σ é dita *finitamente axiomatizável*. ■

Toda lógica monomodal normal pode ser axiomatizada pelo par $\langle \Lambda_{\mathsf{LM}} \oplus \Gamma, \{\text{Nec}, \text{MP}\} \rangle$, onde Γ é um conjunto possivelmente vazio de axiomas adicionais. Caso $\mathfrak{R} = \{\text{Nec}, \text{MP}\}$, podemos omitir $\mathfrak{R}$ da apresentação da axiomatização desta lógica.

Definição 17 (Relação de Consequência Sintática)**.** Sendo $\mathcal{S}$ uma relação definida sobre $2^{\mathsf{LM}} \times \mathsf{LM}$ e sendo $\langle \Sigma, \mathfrak{R} \rangle$ uma axiomatização, $\mathcal{S}$ será dita uma relação de consequência sintática onde, para todo $\Gamma \subseteq \mathsf{LM}$ e $\varphi \in \mathsf{LM}$, $\langle \Gamma, \varphi \rangle \in \mathcal{S}$ se, e somente se, exista uma derivação de φ a partir das fórmulas do conjunto $\Gamma \oplus \Sigma$ usando as regras em $\mathfrak{R}$. Denotaremos consequências por $\Gamma \vdash \varphi$. Caso $\Gamma = \varnothing$ escrevemos $\vdash \varphi$ e φ será dito um teorema. ■

2.4 SISTEMAS DE LÓGICA MODAL E CORRESPONDÊNCIA DE FRAMES

Como apresentado em Chellas (1980) e Gabbay (2003), a imposição de restrições sobre a relação de acessibilidade de um frame torna certos esquemas de fórmulas válidos nestes frames. Estes esquemas de fórmulas são ditos correspondentes aos frames que respeitam a restrição, por exemplo, o esquema $\Box\varphi \to \varphi$ corresponde a frames cuja relação de acessibilidade respeita a propriedade de reflexividade, como apresentado na Tabela 1. O axioma K, apresentado na Seção 2.3, é correspondente a todos os frames. Se o frame de um modelo respeita uma dada restrição, então é dito que este modelo também respeita esta restrição, por exemplo, modelos de frames reflexivos são ditos (modelos) reflexivos. Algumas restrições notáveis e suas correspondentes fórmulas encontram-se na Tabela 1.

O sistema de lógica modal mais simples (também pode ser chamado de sistema básico da lógica modal) chama-se K, em homenagem à Saul Kripke. O sistema K é definido sobre todos

Tipo de Relação	Condição no Frame	Esquema
Reflexiva	$\forall w \in \mathcal{W}, w\mathcal{R}w$	*T*: $\Box\varphi \to \varphi$
Transitiva	$\forall w_0, w_1, w_2 \in \mathcal{W}$, $(w_0\mathcal{R}w_1 \wedge w_1\mathcal{R}w_2) \to w_0\mathcal{R}w_2$	*4*: $\Box\varphi \to \Box\Box\varphi$
Euclidiana	$\forall w_0, w_1, w_2 \in \mathcal{W}$, $(w_0\mathcal{R}w_1 \wedge w_0\mathcal{R}w_2) \to w_1\mathcal{R}w_2$	*5*: $\Diamond\varphi \to \Box\Diamond\varphi$
Simétrica	$\forall w_0, w_1 \in \mathcal{W}$, $w_0\mathcal{R}w_1 \to w_1\mathcal{R}w_0$	*B*: $\varphi \to \Box\Diamond\varphi$
Convergente	$\forall w_0, w_1, w_2, \exists w_3 \in \mathcal{W}$, $(w_0\mathcal{R}w_1 \wedge w_0\mathcal{R}w_2) \to (w_1\mathcal{R}w_3 \wedge w_2\mathcal{R}w_3)$	*C*: $\Diamond\Box\varphi \to \Box\Diamond\varphi$
Serial	$\forall w_0, \exists w_1 \in \mathcal{W}, w_0\mathcal{R}w_1$	*D*: $\Box\varphi \to \Diamond\varphi$
Funcional	$\forall w_0, w_1, w_2 \in \mathcal{W}$, $(w_0\mathcal{R}w_1 \wedge w_0\mathcal{R}w_2) \to w_1 = w_2$	*CD*: $\Diamond\varphi \to \Box\varphi$
Densa	$\forall w_0, w_1, \exists w_2 \in \mathcal{W}$, $w_0\mathcal{R}w_1 \to (w_0\mathcal{R}w_2 \wedge w_2\mathcal{R}w_1)$	*C4*: $\Box\Box\varphi \to \Box\varphi$

Tabela 1 – Relações e axiomas correspondentes
Fonte: Adaptado de Silveira (2020).

os frames $\langle \mathcal{W}, \mathcal{R} \rangle$, sem qualquer restrição sobre $\mathcal{R}$. Sua relação de consequência semântica é descrita por uma relação $\vDash$ definida sobre a classe $\mathfrak{M}$ de todos os modelos cuja relação de acessibilidade do frame não é restrita, e sua relação de consequência sintática é descrita por uma relação $\vdash$ definida sobre a axiomatização $\langle \Lambda_{\mathsf{LM}}, \{\text{Nec}, \text{MP}\} \rangle$. Representaremos a lógica definida sobre o sistema K por $\mathcal{L}_{\mathbf{K}} = \langle \mathsf{LM}, \vDash_{\mathfrak{M}}, \vdash_{\text{K}} \rangle$, ou simplesmente por $\mathbf{K} = \langle \mathsf{LM}, \vDash, \vdash \rangle$.

A imposição de restrições sobre a relação de acessibilidade de frames dá surgimento a outros sistemas de lógica modal, distintos do sistema básico **K**. Sendo $\mathcal{F}_x = \langle \mathcal{W}, \mathcal{R} \rangle$ um frame qualquer pertencente a uma classe de frames $\mathfrak{F}_x$, onde a relação $\mathcal{R}$ de todo frame desta classe respeita alguma restrição *X*. Como $\mathcal{R}$ é restrito, cada frame $\mathcal{F}_x$ de $\mathfrak{F}_x$ descreve uma classe de modelos $\mathfrak{M}_x$ menor que a classe de todos os modelos sem restrição. Já uma axiomatização para $\mathcal{F}_x$ será definida pelo par $\langle \Lambda_{\mathsf{LM}} \oplus \Xi, \{\text{Nec}, \text{MP}\} \rangle$, onde Ξ é o conjunto de axiomas correspondentes à *X*. Portanto, a classe $\mathfrak{F}_x$ gera o sistema $\mathcal{L}_{\mathfrak{F}_x} = \langle \mathsf{LM}, \vDash_{\mathfrak{M}_x}, \vdash_{\Xi} \rangle$, ou simplesmente $\mathbf{F}_x = \langle \mathsf{LM}, \vDash_{\mathbf{F}_x}, \vdash_{\mathbf{F}_x} \rangle$.

Sistemas de lógica modal geralmente são nomeados de acordo com os nomes de seus axiomas correspondentes (apresentados na Tabela 1), por exemplo, é chamado de sistema **KT** o sistema definido sobre a classe de modelos reflexivos[3], já o sistema definido sobre a classe de modelos reflexivos e transitivos é chamado de **KT4**. Podemos omitir o **K** em nomes de sistemas. São exceções à essa regra os sistemas **S4** e **S5**, descritos no exemplo a seguir.

Exemplo 2 (Sistemas de Lógica Modal)**.** A partir do que foi apresentado na Tabela 1, podemos descrever outros sistemas lógicos, como apresentado abaixo (para evitar repetições, a linguagem LM foi omitida das definições):

- $\mathbf{T} = \langle \vDash_{\mathfrak{M}_{\mathcal{T}}}, \vdash_{T} \rangle$, onde $\mathfrak{M}_{\mathcal{T}}$ é a classe de modelos reflexivos;

[3] Lembre que o axioma *K* é correspondente a todos os frames.

- $\mathbf{4} = \langle \vDash_{\mathfrak{M}_4}, \vdash_{4} \rangle$, onde $\mathfrak{M}_4$ é a classe de modelos transitivos;
- $\mathbf{S4} = \langle \vDash_{\mathfrak{M}_{\mathcal{T}4}}, \vdash_{T \oplus 4} \rangle$, onde $\mathfrak{M}_{\mathcal{T}4}$ é a classe de modelos reflexivos e transitivos;
- $\mathbf{S5} = \langle \vDash_{\mathfrak{M}_{\mathcal{T}\mathcal{B}4}}, \vdash_{T \oplus B \oplus 4} \rangle$, onde $\mathfrak{M}_{\mathcal{T}\mathcal{B}4}$ é a classe de modelos reflexivos, simétricos e transitivos. Alternativamente, podemos definir **S5** por:
  - $\mathbf{S5} = \langle \vDash_{\mathfrak{M}_{\mathcal{T}\mathcal{B}4}}, \vdash_{\mathbf{S4} \oplus B} \rangle$, onde $\mathbf{S4} \oplus B$ denota o menor conjunto de axiomas definido pelo conjunto de axiomas do sistema **S4** e o axioma B;
  - $\mathbf{S5} = \langle \vDash_{\mathfrak{M}_{\mathcal{T}5}}, \vdash_{T \oplus 5} \rangle$, onde $\mathfrak{M}_{\mathcal{T}5}$ é a classe de modelos reflexivos e euclideanos. ■

Para denotar valoração de fórmulas ou validade em sistemas específicos de lógica modal usaremos a notação $\mathcal{S}, w \Vdash_{\mathbf{X}} \varphi$ ou $w \Vdash_{\mathbf{X},\mathcal{S}} \varphi$, onde $\mathcal{S}$ é alguma estrutura (modelo, frame, classe de frames, classe de modelos) e $\mathbf{X}$ é um sistema. Omitiremos o $\mathbf{X}$ subscrito caso não seja necessário explicitar o sistema que está sendo discutido. Análogo para $\vDash$ e $\vdash$.

Por fim, usaremos os termos "sistema de lógica modal", "lógica modal" e "lógica" como sinônimos no restante do texto.

2.5 ALGUMAS META PROPRIEDADES

Sistemas de lógica modal são, assim como muitos outros sistemas lógicos, corretos e completos como foi demonstrado por Silveira (2020), Chellas (1980) e Blackburn, Rijke e Venema (2001), e também possuem outras propriedades interessantes, como decidibilidade e propriedade de modelos finitos, como demonstrado por Gabbay (2003). Provar essas propriedades é uma tarefa grande, complexa e que se encontra fora do escopo do presente trabalho. Apesar disso, estas propriedades serão enunciadas, brevemente explicadas e provas delas na literatura serão indicadas.

O problema de demonstrar corretude e completude de sistemas lógicos se resume a demonstrar que há uma correspondência entre suas relações de consequência sintática e semântica. Caso uma lógica seja correta então toda fórmula sintaticamente derivável também é semanticamente derivável, caso uma lógica seja completa toda fórmula semanticamente derivável também é sintaticamente derivável. Logo, as relações de consequência sintática e semântica de uma lógica correta e completa são igualmente expressivas.

As definições de completude e corretude abaixo foram retiradas de Blackburn, Rijke e Venema (2001).

2.5.1 Corretude

Definição 18 (Corretude)**.** Uma lógica modal $\mathcal{L}$ é dita *correta com respeito a* uma classe de frames (modelos) $\mathfrak{S}$ se, para qualquer fórmula φ e qualquer frame (modelo) $\mathcal{S} \in \mathfrak{S}$, se $\vdash_{\mathcal{L}} \varphi$ então $\mathcal{S} \vDash_{\mathcal{L}} \varphi$. ■

Teorema 1 (Corretude da Lógica Modal **K**)**.** *A lógica modal **K** é correta para a classe de todos os frames.* ■

Ideia da Prova do Teorema 1. Provas de corretude de lógicas modais se resumem a provar que os axiomas que axiomatizam uma lógica são corretos, isto é, são fórmulas válidas, e que as regras de derivação preservam validade, isto é, demonstrar que, ao aplicar uma regra de derivação em um conjunto de fórmulas válidas, teremos como resultado outra fórmula válida.

Em particular, para provar a corretude da lógica modal **K** como apresentada neste trabalho, basta provar que os axiomas descritos na Seção 2.3 são válidos e que as regras de derivação de Necessitação e Modus Ponens preservam validade. Como **K** é correta para a classe de todos os frames, qualquer outro sistema da lógica modal "herda" a corretude das regras de derivação e axiomas que axiomatizam **K**. Para provar a corretude de outros sistemas modais, basta provar que seu(s) axioma(s) correspondente(s) é(são) válido(s) na sua respectiva classe de frames.

Provas de corretude podem ser encontradas em Silveira (2020), Chellas (1980) e Zalta (1995). ■

2.5.2 Completude

Definição 19 (Completude)**.** Sendo $\mathfrak{S}$ uma classe de frames (modelos), uma lógica $\mathcal{L}$ é dita *fortemente completa* com respeito a $\mathfrak{S}$ se, para qualquer conjunto de fórmulas Γ e qualquer fórmula φ, se $\Gamma \vDash_{\mathfrak{S},\mathcal{L}} \varphi$ então $\Gamma \vdash_{\mathcal{L}} \varphi$.

Sendo $\mathfrak{S}$ uma classe de frames (modelos), uma lógica $\mathcal{L}$ é dita *fracamente completa* com respeito à $\mathfrak{S}$ se, para qualquer fórmula φ, se $\mathfrak{S} \vDash_{\mathcal{L}} \varphi$ então $\vdash_{\mathcal{L}} \varphi$. ■

Teorema 2 (Completude da Lógica Modal **K**)**.** *A lógica modal **K** é fortemente completa para a classe de todos os frames.* ■

Antes de apresentar a ideia da prova, é necessário definir o conceito de consistência de um conjunto de fórmulas:

Definição 20 (Consistência)**.** Sendo Γ um conjunto de fórmulas, Γ é dito consistente se não existe fórmula φ tal que $\varphi \in \Gamma$ e $\neg\varphi \in \Gamma$. ■

Ideia da Prova do Teorema 2. Ao contrário da corretude, provas de completude para lógica modal são consideravelmente complexas e, de acordo com Blackburn, Rijke e Venema (2001), são essencialmente teoremas sobre a existência de modelos, pois para demonstrar completude de uma dada lógica modal $\mathcal{L}$ com relação à uma dada classe de estruturas $\mathfrak{S}$, é demonstrado que todo subconjunto consistente do conjunto de todos os teoremas de $\mathcal{L}$ é satisfazível em algum modelo.

Então o problema se torna: "como construir estes modelos?". A resposta é: construindo modelos canônicos. Modelos canônicos são uma classe específica de modelos que estão intimamente relacionados com o conceito de conjuntos maximais consistentes de fórmulas, estes são os

"maiores" conjuntos de fórmulas consistentes. Este método de prova é bastante complexo e não há muito mais que possa ser explicado sobre sem antes abordar detalhes técnicos fora do escopo deste trabalho. Blackburn, Rijke e Venema (2001), Chellas (1980) e Silveira (2020) apresentam provas de completude. ■

2.6 LÓGICAS MULTIMODAIS

Lógicas multimodais são uma extensão intuitiva do conceito de lógicas modais com apenas uma modalidade (ou um par de modalidades duais, como é o caso da lógica alética apresentada), que contém diversas modalidades distintas.

2.6.1 Linguagem

A linguagem LM_n da lógica *n*-modal é uma simples extensão da linguagem da lógica modal apresentada na Seção 2.1, onde, ao invés de termos apenas a modalidade $\Box$ e sua dual $\Diamond$, temos *n* modalidades $\Box_1,\ldots,\Box_n$ e suas duais $\Diamond_1,\ldots,\Diamond_n$. O conceito de subfórmulas é estendido para lidar com as novas modalidades. Temos a seguinte definição de *profundidade modal*, como apresentado por Gabbay (2003), que será utilizado posteriormente na prova de transferência de completude pela fusão de lógicas modais:

Definição 21 (Profundidade Modal)**.** A *profundidade de uma fórmula* φ, denotada por $\mathrm{d}(\varphi)$, é definida indutivamente por:

$$\mathrm{d}(\varphi) = 0, \text{ caso } \varphi \in \mathbb{P} \cup \{\top, \bot\}$$
$$\mathrm{d}(\neg\varphi) = \mathrm{d}(\varphi)$$
$$\mathrm{d}(\varphi \circ \psi) = \max(\mathrm{d}(\varphi), \mathrm{d}(\psi)), \circ \in \{\wedge, \vee, \rightarrow\}$$
$$\mathrm{d}(\circ\varphi) = \mathrm{d}(\varphi) + 1, \circ \in \{\Box_1, \ldots, \Box_n, \Diamond_1, \ldots, \Diamond_n\}$$

Onde max é a função de máximo entre dois números. ■

2.6.2 Semântica e Axiomatização

Estender a semântica de mundos possíveis para lidar com diversas modalidades é também intuitivo, basta estender o conceito de frames para *n-frames*, como apresentado por Gabbay (2003).

Definição 22 (n-frames)**.** Um *n-frame é uma n-upla* $\mathcal{F}_n = \langle \mathcal{W}, \mathcal{R}_1, \ldots, \mathcal{R}_n \rangle$, onde $\mathcal{W} \neq \varnothing$ e $\mathcal{R}_i \subseteq \mathcal{W} \times \mathcal{W}, 1 \leq i \leq n$. ■

Definição 23 (n-modelos)**.** Um *n-modelo é um par de n-frame e uma função de valoração* da forma $\mathcal{M}_n = \langle \mathcal{F}_n, \mathcal{V} \rangle$, onde $\mathcal{V}$ é uma função total binária definida por $\mathcal{V} : \mathbb{P} \rightarrow 2^{\mathcal{W}}$. Os n-modelos também podem ser apresentados como $\mathcal{M}_n = \langle \mathcal{W}, \mathcal{R}_1, \ldots, \mathcal{R}_n, \mathcal{V} \rangle$. ■

A definição de valoração de fórmulas em lógicas *n*-modais é uma generalização do caso monomodal, onde os casos para $\Box$ e $\Diamond$ são substituídos por:

$$\mathcal{M}_n, w_0 \Vdash \Box_i \varphi \text{ sse } \forall w_1 \in \mathcal{W}, (w_0 \mathcal{R}_i w_1 \rightarrow \mathcal{M}_n, w_1 \Vdash \varphi), 1 \leq i \leq n$$
$$\mathcal{M}_n, w_0 \Vdash \Diamond_i \varphi \text{ sse } \exists w_1 \in \mathcal{W}, (w_0 \mathcal{R}_i w_1 \wedge \mathcal{M}_n, w_1 \Vdash \varphi), 1 \leq i \leq n$$

As definições de outros conceitos semânticos são estendidas da maneira intuitiva.

Definição 24 (Axiomatização Multimodal)**.** O conjunto de axiomas Λ_{LM_n} de uma lógica *n*-modal é uma generalização do conjunto Λ_{LM} monomodal, onde o axioma *K* é substituído por *n* instâncias de:

$$\Box_i(p_0 \rightarrow p_1) \rightarrow (\Box_i p_0 \rightarrow \Box_i p_1), 1 \leq i \leq n \qquad (\mathrm{K_i})$$

e o conjunto de regras de derivação é uma generalização do caso monomodal, com *n* instâncias da regra Nec, ou seja, $\mathfrak{R} = \{\mathrm{Nec}_1, \ldots, \mathrm{Nec}_n, \mathrm{MP}\}$. Logo, uma lógica n-modal pode ser axiomatizada pelo par $\langle \Lambda_{\mathsf{LM}_n} \oplus \Gamma, \{\mathrm{Nec}_1, \ldots, \mathrm{Nec}_n, \mathrm{MP}\}\rangle$, onde Γ é um conjunto possivelmente vazio de axiomas adicionais. ■

A mesma generalização pode ser aplicada para outros axiomas, como apresentado no exemplo abaixo.

Exemplo 3 (Esquemas Multimodais)**.** Considerando os esquemas de fórmulas descritos na Tabela 1, temos as seguintes generalizações multimodais para alguns deles:

$$T_i: \ \Box_i \varphi \rightarrow \varphi$$
$$4_i: \ \Box_i \varphi \rightarrow \Box_i \Box_i \varphi$$
$$5_i: \ \Diamond_i \varphi \rightarrow \Box_i \Diamond_i \varphi$$
$$B_i: \ \varphi \rightarrow \Box_i \Diamond_i \varphi$$ ■

Relações de consequência sintática e semântica são estendidos de modo intuitivo. O sistema *n*-modal básico $\mathbf{K}_n$ é definido sobre todos os n-frames sem restrições, de forma análoga ao sistema **K** monomodal.

Exemplo 4 (Sistemas Multimodais)**.** Apresentaremos aqui as versões multimodais dos sistemas de lógica modal apresentados no Exemplo 2. Para evitar repetições, a linguagem LM_n foi omitida das definições:

- $\mathbf{T}_n = \langle \vDash_{\mathfrak{M}_{\mathcal{T}_n}}, \vdash_{T_i} \rangle$, onde $\mathfrak{M}_{\mathcal{T}n}$ é a classe de n-modelos reflexivos;
- $\mathbf{4}_n = \langle \vDash_{\mathfrak{M}_{4_n}}, \vdash_{4_i} \rangle$, onde $\mathfrak{M}_{4n}$ é a classe dos n-modelos transitivos;
- $\mathbf{S4}_n = \langle \vDash_{\mathfrak{M}_{\mathcal{T}4_n}}, \vdash_{T_i \oplus 4_i} \rangle$, onde $\mathfrak{M}_{\mathcal{T}4n}$ é a classe dos n-modelos reflexivos e transitivos;

- $\mathbf{S5}_n = \langle \vDash_{\mathfrak{M}_{\mathcal{TB}4n}}, \vdash_{T_i \oplus B_i \oplus 4_i} \rangle$, onde $\mathfrak{M}_{\mathcal{TB}4n}$ é a classe dos n-modelos reflexivos, simétricos e transitivos. Alternativamente, podemos definir $\mathbf{S5}_\mathrm{n}$ por:
  - $\mathbf{S5}_n = \langle \vDash_{\mathfrak{M}_{\mathcal{TB}4n}}, \vdash_{\mathbf{S4}_n \oplus B_i} \rangle$, onde $\mathbf{S4}_n \oplus B_i$ denota o menor conjunto de axiomas definido pelo conjunto de axiomas do sistema $\mathbf{S4}_\mathrm{n}$ e o axioma B_i;
  - $\mathbf{S5}_n = \langle \vDash_{\mathfrak{M}_{\mathcal{T}5n}}, \vdash_{T_i \oplus 5_i} \rangle$, onde $\mathfrak{M}_{\mathcal{T}5n}$ é a classe dos n-modelos reflexivos e euclideanos. ■

Um n-modelo $\mathcal{M}_n = \langle \mathcal{W}, \mathcal{R}_1, \ldots, \mathcal{R}_n, \mathcal{V} \rangle$ é dito respeitar uma restrição se cada $\mathcal{R}_i$ respeita a restrição, por exemplo, se toda $\mathcal{R}_i$ é uma relação reflexiva, então o n-modelo é dito reflexivo. Alternativamente, um sistema descrito pela classe de n-modelos que respeitam uma dada propriedade $\mathcal{X}$ pode ser visto como o sistema onde cada modalidade $\Box_i$ se comporta como a modalidade $\Box$ do sistema monomodal que respeita a mesma propriedade $\mathcal{X}$, como apresentado por Gabbay (2003).

2.7 OUTROS TIPOS DE LÓGICAS MODAIS

Como afirmado no início deste capítulo, a lógica modal não lida apenas com conceito de *necessidade* e *possibilidade*, de fato, linguagens modais são uma excelente ferramenta para raciocinar sobre estruturas relacionais, de acordo com Blackburn, Rijke e Venema (2001). Estruturas relacionais estão presentes em diversas áreas do conhecimento, por exemplo, sistemas de transição para programas de computador, redes semânticas para representação de conhecimento ou estruturas linguísticas de atributo e valor, de acordo com Gabbay (2003).

Para raciocinar sobre estas e outras estruturas relacionais, é necessário especializar a lógica modal que foi apresentada até então, onde os conectivos $\Box$ e $\Diamond$ de necessidade e possibilidade são substituídos por outros conectivos modais que representam os conceitos desejados de modo mais preciso. Ademais, sistemas modais mais complexos podem ser construídos a partir da *combinação* de sistemas modais mais simples (ROGGIA, 2012), algo que será visto mais à frente.

Nesta seção, serão brevemente apresentadas as lógicas modais temporal, epistêmica e temporal epistêmica, com alguns exemplos de suas aplicações em computação.

2.7.1 Lógica Temporal

De acordo com Emerson (1990), lógica temporal é um tipo especial de lógica modal que define um sistema para descrever e raciocinar sobre como a veracidade de afirmações mudam com o tempo. Já Gabbay (2003) define como uma aplicação natural e intuitiva da lógica modal. Mais ainda, Emerson (1990) apresenta uma classificação de lógicas temporais, onde estas se dividem em: proposicional ou primeira ordem, global ou composicional, tempo ramificável ou linear, pontos de tempo ou intervalos de tempo, tempo discreto ou contínuo, passado ou futuro.

Gabbay (2003) apresenta cinco modalidades para lógica temporal, “sempre no passado/futuro”, “desde”, “até” e “no próximo instante”. Já Emerson (1990) apresenta quatro modalidades para a lógica temporal linear, “eventualmente”, “sempre”, “no próximo instante” e “até”.

De acordo com Emerson (1990), Pnueli (1977) foi o primeiro a descrever a lógica temporal como formalismo para verificação de programas sequenciais e concorrentes. Extensões da lógica temporal também são muito usadas para formalizar e verificar programas, por exemplo Lamport (1983) descreve um formalismo baseado em lógica temporal para especificar programas concorrentes, este foi estendido em Lamport (1994) com a descrição de um sistema lógico, baseado em lógica temporal, chamado de TLA (*Temporal Logic of Actions*[4]) para formalizar e raciocinar sobre programas concorrentes, por fim Lamport (2002) estende o sistema TLA gerando então o sistema TLA$^+$.

2.7.2 Lógica Epistêmica

Segundo Fagin et al. (1995), a lógica epistêmica é a lógica que estuda o conhecimento, de um ou mais agentes. Gabbay (2003) afirma que a lógica epistêmica é relevante para diversas disciplinas, dentre elas: teoria dos jogos, inteligência artificial e sistemas multiagentes.

Gabbay (2003) apresenta uma formalização da lógica epistêmica como uma lógica multimodal com diversas modalidades $\Box$, onde cada $\Box_i$ indica o conhecimento de um agente i sobre uma dada fórmula ou proposição. Já Lescanne (2004) apresenta como uma lógica multimodal com três modalidades, K_{i}, E_{G} e C que representam, respectivamente, conhecimento de um agente i, conhecimento de todos os agentes num grupo G e conhecimento comum a todos os agentes.

Por fim, Gabbay (2003) define que a lógica epistêmica pode ser interpretada em sistemas multimodais aléticos, caso o operador $\Box$ seja interpretado de forma epistêmica, onde cada sistema impõe restrições sobre o conhecimento de um agente. Por exemplo, em um interpretação epistêmica de $\mathbf{T}_{\mathrm{n}}$, é imposta a restrição que tudo que um agente sabe é verdadeiro.

Gabbay (2003) descreve uma aplicação de lógica epistêmica no problema dos três homens sábios e os chapéus, apresentado como o problema pode ser modelado dentro de um sistema lógico epistêmico e descrevendo como o problema é usualmente resolvido.

2.7.3 Lógica Temporal Epistêmica

Lógica Temporal Epistêmica é uma família de lógicas que foi construída com o intuito de formalizar o comportamento de agentes em sistemas multiagentes (GABBAY, 2003). Fagin et al. (1995) afirma que a combinação de operadores epistêmicos e temporais numa única linguagem possibilita que afirmações sobre o desenvolvimento do conhecimento no sistema sejam feitas.

A sua linguagem incorpora elementos da linguagem da lógica temporal e da lógica epistêmica para poder representar afirmações sobre o conhecimento de agentes com relação ao

[4] Lógica Temporal das Ações.

tempo, por exemplo "O agente i sempre soube φ" (FAGIN et al., 1995).

Em Fagin et al. (1995) é apresentada, como exemplo de uma lógica temporal epistêmica para sistemas multiagentes, a formalização de um sistema de transmissão de bits entre agentes.

2.8 OUTRA FORMA DE DEFINIR LÓGICAS

A interpretação de lógicas (modais) como sistemas que contém uma linguagem e uma (ou mais) relação(ões) de consequência definida(s) sobre essa linguagem apresentada no começo deste capítulo não é a única definição possível do conceito de lógica. A forma utilizada por Blackburn, Rijke e Venema (2001), Gabbay (2003) e Fine e Schurz (1996) por exemplo, é apresentando lógicas como conjuntos (usualmente infinitos) de fórmulas fechados para algum operador de consequência. Esta apresentação de lógica é útil para demonstrar meta propriedades de sistemas lógicos, como completude e corretude.

Nesta interpretação, é definido um conjunto base de fórmulas, usualmente um conjunto de axiomas, como o apresentado na Definição 15, ou o conjunto de todas as tautologias, e é definida uma operação de consequência (sintática ou semântica) sob a qual este conjunto deve ser fechado. O conceito de teoremas/tautologias/consequências é definido a partir do fato da fórmula pertencer ao conjunto, dependendo da relação sobre a qual o conjunto é construído.

Diferentes autores apresentam definições levemente diferentes uma da outra, variando a assinatura utilizada ou o conjunto de axiomas base, porém todas seguem a forma geral descrita anteriormente. Por exemplo Blackburn, Rijke e Venema (2001) define lógica modal sobre a assinatura $\{\neg, \Diamond, \vee\}$ utilizando o conjunto de todas as tautologias proposicionais juntamente com o axioma K e a fórmula $\Diamond\varphi \leftrightarrow \neg\Box\neg\varphi$, já Gabbay (2003) define lógica modal sobre a assinatura $\{\neg, \Box, \Diamond, \wedge, \vee, \rightarrow\}$ utilizando um conjunto de axiomas semelhante ao apresentado na Definição 15.

Para a definição abaixo, vamos considerar a linguagem LM apresentada na Definição 1, o conjunto Λ_{LM} apresentado na Definição 15, a regra de Substituição apresentada na Definição 13 e as regras de *Modus Ponens* e Necessitação apresentadas na Definição 14.

Definição 25 (Lógica Modal como Conjunto)**.** Uma lógica modal definida sobre uma teoria Γ é o menor conjunto $\mathcal{L}$ tal que $\Gamma \subseteq \mathcal{L}$, $\Lambda_{\mathsf{LM}} \subseteq \mathcal{L}$ e $\mathcal{L}$ é fechado para a regras de Substituição, *Modus Ponens* e Necessitação. Se $\varphi \in \mathcal{L}$ então φ é dito um *teorema* e escrevemos $\vdash_{\mathcal{L}} \varphi$, caso contrário escrevemos $\nvdash_{\mathcal{L}} \varphi$. ■

Os sistemas modais apresentados na Seção 2.4 podem também ser descritos com essa definição de lógica. O sistema **K** é definido a partir de $\Gamma = \varnothing$, o Sistema **T** é definido a partir de $\Gamma = \{\Box\varphi \rightarrow \varphi\}$, já o Sistema **S5** pode ser definido a partir de $\Gamma = \{\Box\varphi \rightarrow \varphi, \Box\varphi \rightarrow \Box\Box\varphi, \varphi \rightarrow \Box\Diamond\varphi\}$ ou $\Gamma = \{\Box\varphi \rightarrow \varphi, \Diamond\varphi \rightarrow \Box\Diamond\varphi\}$.

Lógicas multimodais são definidas de forma análoga. Podemos usar a notação $\vdash_{\mathcal{L}} \varphi$ para indicar que $\varphi \in \mathcal{L}$, e $\nvdash_{\mathcal{L}} \varphi$ para indicar que $\varphi \notin \mathcal{L}$. Tendo apresentado o conceito de lógica como

conjunto, podemos apresentar o conceito de dedutibilidade em lógicas vistas como conjuntos, como apresentado em Blackburn, Rijke e Venema (2001):

Definição 26 (Dedutibilidade)**.** Sendo Γ um conjunto de fórmulas, φ uma fórmula e $\mathcal{L}$ uma lógica, é dito que φ *é dedutível em* $\mathcal{L}$ *a partir de* Γ se $\vdash_{\mathcal{L}} \varphi$, ou existem fórmulas $\gamma_0, \ldots, \gamma_n \in \Gamma$ tal que:

$$\vdash_{\mathcal{L}} (\gamma_0 \wedge \cdots \wedge \gamma_n) \rightarrow \varphi$$

Caso φ seja dedutível em $\mathcal{L}$ a partir de Γ, escrevemos $\Gamma \vdash_{\mathcal{L}} \varphi$, caso não seja, escrevemos $\Gamma \nvdash_{\mathcal{L}} \varphi$. ■

Segundo Blackburn, Rijke e Venema (2001), todas as lógicas são fechadas sob a operação de dedução: se φ é dedutível a partir das premissas $\psi_0, \ldots, \psi_n$, então $\vdash \psi_0$ e $\ldots$ e $\vdash \psi_n$ implica que $\vdash \varphi$.

Note que esta é uma apresentação sintática do conceito de lógica, porém ela não é muito diferente de uma apresentação semântica do conceito: considere $\mathfrak{F}$ uma classe de frames, a lógica definida por essa classe será dada pelo conjunto $\mathcal{L}_{\mathfrak{F}} = \{\varphi \mid \varphi \in \mathsf{LM} \text{ e } \mathfrak{F} \Vdash \varphi\}$.

Não é difícil de observar que essa definição de lógicas modais é equivalente à definição apresentada antes neste capítulo. Considere, por exemplo, a lógica modal **K** apresentada na Seção 2.4, ela pode ser igualmente definida pelos conjuntos $T_{\vdash} = \{\varphi \mid \varphi \in \mathsf{LM} \text{ e } \vdash_{\mathrm{K}} \varphi\} \cup \{(\gamma_0 \wedge \cdots \wedge \gamma_n) \rightarrow \varphi \mid \varphi, \gamma_0, \ldots, \gamma_n \in \mathsf{LM} \text{ e } \gamma_0, \ldots, \gamma_n \vdash_{\mathrm{K}} \varphi\}$ e $T_{\models} = \{\varphi \mid \varphi \in \mathsf{LM} \text{ e } \mathfrak{F} \Vdash \varphi\} \cup \{(\gamma_0 \wedge \cdots \wedge \gamma_n) \rightarrow \varphi \mid \varphi, \gamma_0, \ldots, \gamma_n \in \mathsf{LM} \text{ e } \gamma_0, \ldots, \gamma_n \models_{\mathfrak{M}} \varphi\}$ (sendo $\mathfrak{F}$ a classe de frames irrestritos e $\mathfrak{M}$ a classe de modelos construídos com frames em $\mathfrak{F}$), que representam, respectivamente, todas as fórmulas semântica e sintaticamente deriváveis no sistema **K**.

Naturalmente, é possível expressar meta propriedades de lógicas apresentadas como conjuntos tal e qual é possível expressar para lógicas apresentadas como sistemas. As definições abaixo foram retiradas de Blackburn, Rijke e Venema (2001).

Definição 27 (Corretude como Conjunto)**.** Uma lógica $\mathcal{L}$ é dita *correta com relação a uma classe de frames (modelos)* $\mathfrak{S}$ se, e somente se, se $\varphi \in \mathcal{L}$ então $\mathfrak{S} \Vdash \varphi$. Ou seja, $\mathcal{L} \subseteq \mathcal{L}_{\mathfrak{S}}$ ■

Definição 28 (Completude como Conjunto)**.** Sendo $\mathfrak{S}$ uma classe de frames (modelos), uma lógica $\mathcal{L}$ é dita *fortemente completa* com respeito a $\mathfrak{S}$ se, para qualquer conjunto de fórmulas Γ e qualquer fórmula φ, se $\Gamma \models_{\mathfrak{S}} \varphi$ então $\Gamma \vdash \varphi$.

Sendo $\mathfrak{S}$ uma classe de frames (modelos), uma lógica $\mathcal{L}$ é dita *fracamente completa* com respeito à $\mathfrak{S}$ se, para qualquer fórmula φ, se $\mathfrak{S} \Vdash \varphi$ então $\vdash_{\mathcal{L}} \varphi$. Ou seja, $\mathcal{L}_{\mathfrak{S}} \subseteq \mathcal{L}$. ■

Provar a completude e a corretude para lógicas vistas como conjuntos pode ser entendido como provar que as definições sintáticas e semânticas desta lógica são equivalentes, como afirmado por Blackburn, Rijke e Venema (2001). Considerando $\mathcal{L}$ uma lógica descrita sintaticamente, como apresentado anteriormente, e $\mathcal{L}_{\mathfrak{F}}$ sua correspondente semântica com relação à alguma

classe de frames $\mathfrak{F}$. Caso $\mathcal{L}$ seja correta para a classe $\mathfrak{F}$, então $\mathcal{L} \subseteq \mathcal{L}_{\mathfrak{F}}$, caso $\mathcal{L}$ seja completa, então $\mathcal{L}_{\mathfrak{F}} \subseteq \mathcal{L}$, portanto, caso $\mathcal{L}$ seja correta e completa temos que $\mathcal{L} = \mathcal{L}_{\mathfrak{F}}$.

Podemos então apresentar a seguinte definição, retirada de Blackburn, Rijke e Venema (2001), cuja importância será vista a frente.

Definição 29 (Consistência com Respeito a Conjunto)**.** Sendo Γ e Δ conjuntos de fórmulas, Γ *é dito consistente com respeito a* Δ*, ou* Δ*-consistente*, se, e somente se, não existe γ tal que $\gamma \in \Gamma$ e $\neg\gamma \in \Delta$. Uma *fórmula* φ *é dita* Δ*-consistente* se $\{\varphi\}$ é Δ-consistente. ■

Na definição anterior, caso Δ seja uma lógica, Γ será dito consistente com respeito a Δ se $\Gamma \nvdash_{\Delta} \bot$, pois $\Gamma \vdash_{\Delta} \bot$ implica que existem fórmulas $\gamma_0, \ldots, \gamma_n \in \Gamma$ tal que $\vdash_{\Delta} (\gamma_0 \wedge \cdots \wedge \gamma_n) \rightarrow \bot$, ou seja, $\vdash_{\Delta} \neg(\gamma_0 \wedge \cdots \wedge \gamma_n) \vee \bot$. Isto é, existe pelo menos um γ_i que pertence à Γ mas não pertence à Δ ou $\bot \in \Delta$. Caso $\bot \in \Delta$, Δ é dita uma lógica inconsistente e todo conjunto é inconsistente com respeito à Δ.

Por fim, temos o seguinte resultado auxiliar referente à completude, retirado de Blackburn, Rijke e Venema (2001), cuja importância também será vista a frente:

Proposição 1. *Uma lógica $\mathcal{L}$ é dita fortemente completa para uma classe de frames $\mathfrak{F}$, se, e somente se, todo conjunto de fórmulas $\mathcal{L}$-consistente é satisfazível em algum frame de $\mathfrak{F}$. Uma lógica $\mathcal{L}$ é dita fracamente completa para uma classe de frames $\mathfrak{F}$, se, e somente se, toda fórmula $\mathcal{L}$-consistente é satisfazível em algum frame de $\mathfrak{F}$.* ■

Satisfazibilidade em um frame pode ser entendida como satisfazibilidade em ao menos um modelo construído com aquele frame. Analogamente para classes de frames.

Prova da Proposição 1. A prova para completude fraca é um caso particular para o caso da completude forte, onde $\Gamma = \varnothing$, logo, provaremos apenas o caso da completude forte. A prova é dividida em dois casos; no primeiro caso é provado que se uma lógica é fortemente completa então todo conjunto consistente é satisfazível, já no segundo caso é provado que se todo conjunto consistente é satisfazível então uma lógica é fortemente completa. Ambos os casos serão provados por contraposição.

Caso 1 Supondo algum conjunto de fórmulas Γ e alguma fórmula φ tal que $\Gamma \cup \{\varphi\}$ é $\mathcal{L}$-consistente e insatisfazível em $\mathfrak{F}$. Como $\Gamma \cup \{\varphi\}$ é insatisfazível, temos, pela definição de $\vDash$, que $\Gamma \cup \{\varphi\} \vDash_{\mathcal{M}} \bot$[5], em algum modelo $\mathcal{M} = \langle \mathcal{F}, \mathcal{V} \rangle$ onde $\mathcal{F} \in \mathfrak{F}$. Como $\Gamma \cup \{\varphi\}$ é $\mathcal{L}$-consistente, temos que $\Gamma \cup \{\varphi\} \nvdash_{\mathcal{L}} \bot$. Portanto, $\mathcal{L}$ não é fortemente completo para $\mathfrak{F}$.

Caso 2 Supondo que $\mathcal{L}$ não é fortemente completa com relação à $\mathfrak{F}$. Logo, existe um conjunto de fórmulas Γ e uma fórmula φ tal que $\Gamma \vDash_{\mathfrak{F}} \varphi$, porém $\Gamma \nvdash_{\mathcal{L}} \varphi$. Logo, o conjunto $\Gamma \cup \{\neg\varphi\}$ é $\mathcal{L}$-consistente, porém não é satisfazível em qualquer frame da classe $\mathfrak{F}$. ■

[5] A definição de $\vDash$ nos diz que "Caso o antecedente é satisfazível em todos os mundos de algum modelo $\mathcal{M}$, então o consequente também será" Como $\Gamma \cup \{\varphi\}$ é insatisfazível em todos os mundos de qualquer modelo $\mathcal{M}$ construído com qualquer frame da classe $\mathfrak{F}$, temos que qualquer fórmula é consequência semântica de $\Gamma \cup \{\varphi\}$.

Note que a Proposição 1 indica que há uma correspondência entre o conceito de completude com alguma classe de frames e satisfazibilidade de conjuntos $\mathcal{L}$-consistente nesta classe.

3 FUSÕES DE LÓGICAS MODAIS

A fusão de lógicas modais é o método mais simples de combinar duas lógicas modais, segundo Gabbay (2003). Combinações de lógicas é um tópico relativamente recente no estudo da lógica moderna, de acordo com Carnielli e Coniglio (2020). Consiste de métodos para sintetizar novos sistemas lógicos a partir de sistemas já existentes, seja por meio da junção de diversos sistemas em um único ou pela separação de um sistema lógico em diversos sistemas componentes (CARNIELLI et al., 2008). A ideia fundamental por trás da combinação de lógicas é que sistemas lógicos podem ser "reutilizados", gerando sistemas mais expressivos mas que possuam as mesmas propriedades úteis dos sistemas lógicos base (ROGGIA, 2012).

A combinação de lógicas é uma ferramenta para tratar problemas tanto práticos quanto filosóficos. Por exemplo, linguagens lógicas que tratam de espaço, tempo e conhecimento podem ser combinadas num sistema lógico ideal para raciocinar sobre sistemas multiagentes. Já linguagens lógicas que tratam de obrigatoriedade e possibilidade podem ser combinadas para tratar do problema da "Obrigação implica Possibilidade", onde o fato que algo é obrigatório implica que é logicamente possível de ser feito ou ocorrer (CARNIELLI et al., 2008).

Segundo Carnielli et al. (2008), a fusão de lógicas modais pode ser vista como um mecanismo de combinação de lógicas $\odot$, tal que, dadas duas lógicas $\mathcal{L}_1$ e $\mathcal{L}_2$, gera uma nova lógica $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$. A lógica $\mathcal{L}_{12}$ é então definida como uma extensão (máxima ou mínima) das lógicas base, isto é, estende a linguagem, semântica e sintaxe de $\mathcal{L}_1$ e $\mathcal{L}_2$ de maneira controlada, de forma que nenhuma característica indesejada seja inclusa em $\mathcal{L}_{12}$. Mais ainda, as lógicas base devem ser apresentadas da mesma forma, por exemplo, duas lógicas modais apresentadas semanticamente por uma semântica de mundos possíveis e sintaticamente por uma axiomatização de Hilbert.

O estudo de combinações de lógicas modais se iniciou, segundo Roggia (2012), com Fitting (1969), onde o autor apresenta sistemas lógicos resultantes de combinações de S4, S5 e sistemas lógicos deônticos por um método semelhante à fusão, já Thomason (1984) introduziu o termo "fusão" e definiu sistematicamente a fusão de lógicas modais. Ademais, segundo Carnielli e Coniglio (2020), a fusão definida por Thomason foi o primeiro método genérico de combinações de lógicas.

No que segue, será apresentada a fusão de lógicas modais e suas propriedades de acordo com a formulação de Gabbay (2003).

Definição 30 (Fusão de Lógicas Modais)**.** Sendo $\mathcal{L}_1$ e $\mathcal{L}_2$ lógicas monomodais, cujas linguagens LM_1 e LM_2 são definidas sobre as assinaturas C_1 e C_2 com modalidades $\Box_1$ e $\Box_2$ distintas entre si[1], correspondentes às classes de frames $\mathfrak{F}_1$ e $\mathfrak{F}_2$, sendo $\langle \mathcal{W}, \mathcal{R} \rangle$ e $\langle \mathcal{W}, \mathcal{S} \rangle$ seus respectivos representantes[2], e axiomatizadas por $\langle \Lambda_{\mathsf{LM}_1} \oplus \Xi_1, \{\mathrm{Nec}_1, \mathrm{MP}\} \rangle$ e $\langle \Lambda_{\mathsf{LM}_2} \oplus \Xi_2, \{\mathrm{Nec}_2, \mathrm{MP}\} \rangle$.

A lógica $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$ resultante da fusão de $\mathcal{L}_1$ com $\mathcal{L}_2$ terá linguagem LM_{12} definida

[1] Isto é, $\Box_1$ e $\Box_2$ são interpretadas de maneira diferente.

[2] Note que o conjunto de mundos $\mathcal{W}$ é o mesmo em todos os frames de ambas as classes.

sobre a assinatura $\mathsf{C}_{12} = \mathsf{C}_1 \cup \mathsf{C}_2$, será correspondente à classe $\mathfrak{F}_{12}$ de 2-frames da forma $\langle \mathcal{W}, \mathcal{R}, \mathcal{S} \rangle$ e será axiomatizada por $\langle \Lambda_{\mathsf{LM}_{12}} \oplus (\Xi_1 \oplus \Xi_2), \{\text{Nec}_1, \text{Nec}_2, \text{MP}\} \rangle$. ■

Como LM_1 e LM_2 não compartilham modalidades, $\mathcal{L}_{12}$ será dita *independentemente axiomatizável* como apresentado por Kracht e Wolter (1991). A operação de fusão é associativa e comutativa (GABBAY, 2003), e a fusão de lógicas modais pode ser vista como uma forma de obter as lógicas multimodais descritas na Seção 2.6. O caso de fusões de lógicas multimodais é uma generalização do caso acima, onde LM_1 e LM_2 são linguagens multimodais e os demais conceitos são generalizados de acordo.

Exemplo 5 (Lógica Epistêmica como Fusão)**.** Como apresentado em Gabbay (2003), considerar lógicas epistêmicas como fusões de lógicas aléticas é algo simples. Considere $\mathcal{L}_A$, $\mathcal{L}_B$ e $\mathcal{L}_C$ lógicas monomodais aléticas cujas modalidades representam o conhecimento de agentes A, B e C respectivamente. Caso seja desejado um sistema lógico que modele interação entre os agentes, é necessário descrever um formalismo que seja capaz de expressar tanto o conhecimento de cada agente sobre o mundo ao seu redor, quanto o conhecimento de cada agente sobre o conhecimento *dos outros* agentes. Esse sistema é obtido pela fusão das lógicas $\mathcal{L}_A$, $\mathcal{L}_B$ e $\mathcal{L}_C$ em um único sistema lógico $\mathcal{L}_{ABC}$. Os axiomas deste sistema que contém alguma modalidade descrevem o conhecimento do seu respectivo agente, mas, por definição, todos os axiomas contém apenas uma modalidade, logo não há nenhum axioma que explicitamente relaciona o conhecimento de dois agentes distintos.

Porém, não é difícil imaginar situações onde interações entre os agente são necessárias, por exemplo caso um agente saiba tudo o que o outro sabe. Para lidar com essas restrições, basta adicionar novos axiomas na axiomatização de $\mathcal{L}_{ABC}$. Por exemplo, caso seja necessário representar a afirmação "o agente A sabe tudo o que o agente B sabe" basta adicionar o axioma $\Box_B p_0 \rightarrow \Box_A p_0$ na axiomatização de $\mathcal{L}_{ABC}$[3]. ■

Este capítulo é organizado da seguinte forma: na Seção 3.1 será apresentado o conceito de preservação de propriedades pela fusão de lógicas modais e será provada a transferência de algumas propriedades pela fusão e na Seção 3.2 será apresentado o sistema bimodal $\mathbf{KT} \odot \mathbf{K4}$, que será utilizado posteriormente para a prova de conceito de fusão de lógicas modais em Coq.

3.1 PRESERVAÇÃO DE PROPRIEDADES

Uma das mais importantes características da fusão de lógicas modais é a preservação de propriedades, também chamada de transferência de teoremas por (FINE; SCHURZ, 1996). Segundo (KRACHT; WOLTER, 1991), uma propriedade é dita preservada pela fusão se, sendo $\mathcal{L}_{12}$ uma lógica multimodal resultante da fusão de duas lógicas (multi)modais $\mathcal{L}_1$ e $\mathcal{L}_2$, caso $\mathcal{L}_1$ e $\mathcal{L}_2$ satisfaçam alguma propriedade P, então $\mathcal{L}_{12}$ também ira satisfazer P.

[3] Obtendo então um sistema $\mathcal{L}'_{ABC}$.

Dentre as propriedades que são preservadas pela fusão de lógicas modais, têm-se as descritas na Seção 2.5. No restante da seção, serão apresentados conceitos e provas referentes à preservação das propriedades de corretude e completude, retirados de Fine e Schurz (1996).

Em Fine e Schurz (1996), os autores consideram uma lógica monomodal como um conjunto de fórmulas, como foi descrito na Seção 2.8. Para manter consistência com a prova original, usaremos esta definição de lógica pelo restante da seção. Conceitos semânticos, como frames, modelos e classes destes não são alterados.

Definição 31 (Fusão de Lógicas como Conjuntos)**.** Sendo $\mathcal{L}_1$ e $\mathcal{L}_2$ lógicas monomodais cujas linguagens são LM_1 e LM_2 e são definidas com base nos conjuntos Λ_{LM_1} e Λ_{LM_2}. A lógica $\mathcal{L}_{12}$ resultante da fusão de $\mathcal{L}_1$ e $\mathcal{L}_2$ terá linguagem LM_{12} (obtida da forma descrita na Definição 30) e será definida (remeta à Definição 25) com base no conjunto $\Lambda_{\mathsf{LM}_1} \cup \Lambda_{\mathsf{LM}_2}$.

Definição 32 (Frames e Modelos para Lógicas)**.** Sendo $\mathcal{L}$ uma lógica, um frame (modelo) $\mathcal{S}$ é dito um frame (modelo) *para $\mathcal{L}$ se, e somente se, $\mathcal{L}$ é válida em $\mathcal{S}$*[4]. ■

No restante desta sessão, serão analisadas duas lógicas monomodais $\mathcal{L}_1$ e $\mathcal{L}_2$ cujas linguagens são LM_1 e LM_2, estas são combinadas numa lógica bimodal $\mathcal{L}_{12}$ cuja linguagem é LM_{12}. As linguagens $\mathsf{LM}_1, \mathsf{LM}_2$ e LM_{12} são definidas de forma análoga à linguagem LM apresentada na Definição 1, porém são baseadas nas assinaturas minímas $\mathsf{C}_1 = \{\vee, \neg, \Box_1\}$, $\mathsf{C}_2 = \{\vee, \neg, \Box_2\}$ e $\mathsf{C}_{12} = \{\vee, \neg, \Box_1, \Box_2\}$ respectivamente. Os conectivos $\wedge, \rightarrow, \Diamond_1$ e $\Diamond_2$ são abreviações definidas a partir das dualidades apresentadas na Definição 2.

Denotaremos por $\mathfrak{F}_1$ e $\mathfrak{F}_2$ as classes de frames de $\mathcal{L}_1$ e $\mathcal{L}_2$ e denotaremos por $\mathfrak{F}_{12} = \mathfrak{F}_1 \otimes \mathfrak{F}_2$ a classe de 2-frames $\{\langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2 \rangle \mid \langle \mathcal{W}, \mathcal{R}_1 \rangle \in \mathfrak{F}_1 \text{ e } \langle \mathcal{W}, \mathcal{R}_2 \rangle \in \mathfrak{F}_2\}$, da lógica $\mathcal{L}_{12}$. Análogo para modelos e classes de modelos.

3.1.1 Preservação de Corretude

Considerando a Definição 27 que apresenta o conceito de completude para lógicas descritas como conjuntos, temos o seguinte teorema:

Teorema 3 (Transferência de Corretude)**.** *Sendo $\mathcal{L}_1$ e $\mathcal{L}_2$ lógicas modais corretas com relação às classes de frames $\mathfrak{F}_1$ e $\mathfrak{F}_2$, $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$ será correta com relação a classe de frames $\mathfrak{F}_{12} = \mathfrak{F}_1 \otimes \mathfrak{F}_2$.* ■

Prova do Teorema 3. A prova decorre diretamente da definição de $\mathfrak{F}_{12}$ e da corretude de $\mathcal{L}_1$ e $\mathcal{L}_2$. Como $\mathfrak{F}_{12}$ é uma classe de frames forma $\{\langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2 \rangle \mid \langle \mathcal{W}, \mathcal{R}_1 \rangle \in \mathfrak{F}_1 \text{ e } \langle \mathcal{W}, \mathcal{R}_2 \rangle \in \mathfrak{F}_2\}$ e ambas as lógicas são corretas, sabemos que todos os axiomas proposicionais (Definição 15) são válidos em todos os mundos $w \in \mathcal{W}$ de todo frame $\mathcal{F}_x \in \mathfrak{F}_x, x \in \{1,2\}$, mais ainda, temos que a regra de Modus Ponens (Definição 14) e a propriedade de Substituição (Definição 13) preservam validade.

[4] Veja as definições 7 e 9.

Como $\mathcal{L}_1$ é correta para $\mathfrak{F}_1$, sabemos que toda instância de K_1 é válida em todos os mundos $w \in \mathcal{W}$ de todo frame $\mathcal{F}_1 \in \mathfrak{F}_1$ e que a regra de Necessitação para $\Box_1$ (Definição 24) preserva validade.

Como $\mathcal{L}_2$ é correta para $\mathfrak{F}_2$, sabemos que toda instância de K_2 é válida em todos os mundos $w \in \mathcal{W}$ de todo frame $\mathcal{F}_2 \in \mathfrak{F}_2$ e que a regra de Necessitação para $\Box_2$ (Definição 24) preserva validade. ■

3.1.2 Preservação de Completude

Uma lógica é dita (forte ou fracamente) completa se ela é (forte ou fracamente) completa para alguma classe de frames. Ademais, toda lógica é (forte ou fracamente) completa para a sua classe de frames correspondentes[5]. Logo, para provar que a lógica $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$ é completa, dada a completude de $\mathcal{L}_1$ para $\mathfrak{F}_1$ e $\mathcal{L}_2$ para $\mathfrak{F}_2$, basta provar que ela é completa para a classe $\mathfrak{F}_{12} = \mathfrak{F}_1 \otimes \mathfrak{F}_2$ de frames.

Para demonstrar a preservação de completude, devemos antes apresentar alguns conceitos que são necessários para construir a prova.

Toda fórmula da linguagem bimodal LM_{12} pode ser vista como uma fórmula da linguagem LM_1 (LM_2) se tratarmos todas as subfórmulas cujo conectivo mais externo é $\Box_2$ ($\Box_1$) como átomos. Por exemplo, a fórmula:

$$\Box_1(p_0 \vee p_1) \rightarrow \Box_2(p_3 \wedge (\Box_1 p_0 \vee \Box_2 p_1))$$

pode ser vista uma fórmula de LM_1, se tomarmos $\Box_2(p_3 \wedge (\Box_1 p_0 \vee \Box_2 p_1))$ como um átomo:

$$\Box_1(p_0 \vee p_1) \rightarrow \boldsymbol{p}_{(\Box_2(p_3 \wedge (\Box_1 p_0 \vee \Box_2 p_1)))}$$

ou como uma fórmula de LM_2, se tomarmos $\Box_1(p_0 \vee p_1)$ e $\Box_1 p_0$ como átomos:

$$\boldsymbol{p}_{(\Box_1(p_0 \vee p_1))} \rightarrow \Box_2(p_3 \wedge (\boldsymbol{p}_{(\Box_1 p_0)} \vee \Box_2 p_1))$$

Usaremos a variável $\pi \in \{1,2\}$ no lugar dos índices 1 e 2 para indicar uma lógica monomodal e seus componentes (fórmula, frame, modelo, etc.) e usaremos a variável $\overline{\pi}$ para indicar a lógica monomodal oposta e seus componentes, isto é, $\pi = 1$ sse $\overline{\pi} = 2$ e vice versa. Note que π e $\overline{\pi}$ são símbolos da nossa metalinguagem e não fazem parte da linguagem da lógica modal, apenas indicam o índice de um objeto qualquer.

Definição 33 (Fragmentos Monomodais e Elementos)**.** Sendo $\boxdot_\pi = \{\Box_\pi \varphi \mid \varphi \in \mathsf{LM}\}$ o conjunto de todas as fórmulas cujo conectivo principal é a modalidade $\Box_\pi$, podemos definir os *fragmentos π-modais de* LM_{12} como sendo os menores conjuntos LM_π que satisfazem as seguintes regras:

$$\mathbb{P} \cup \boxdot_{\overline{\pi}} \subseteq \mathsf{LM}_\pi$$

[5] Alternativamente, uma lógica pode ser dita correspondente à alguma classe de frames sse ela for (forte ou fracamente) completa para essa classe.

Se $\varphi \in \mathsf{LM}_\pi$, então $\circ\varphi \in \mathsf{LM}_\pi$, sendo $\circ \in \{\Box_\pi, \neg\}$

Se $\varphi, \psi \in \mathsf{LM}_\pi$, então $\varphi \vee \psi \in \mathsf{LM}_\pi$

O conjunto $\Upsilon_\pi = \mathbb{P} \cup \boxdot_{\overline{\pi}}$ é chamado de conjunto de π-átomos de uma linguagem LM_π[6]. O conjunto $\Upsilon = \Upsilon_1 \cup \Upsilon_2$ é o conjunto de todos os elementos de LM_{12}, isto é, o conjunto de todas as fórmulas atômicas ou cujo conectivo mais externo é uma modalidade. ■

Definição 34 (π-Lógicas, π-Frames e π-Modelos)**.** Uma *π-lógica* é um conjunto $\mathcal{L}_\pi$ definido como o fecho do conjunto contendo todos as tautologias e o axioma K_π para Modus Ponens Necessitação para $\Box_\pi$ e Substituição de fórmulas por elementos de Υ_π[7,8].

Um *π-frame* é um par da forma $\langle \mathcal{W}, \mathcal{R}_\pi \rangle$ e um *π-modelo* é uma tripla da forma $\langle \mathcal{W}, \mathcal{R}_\pi, \mathcal{V}_\pi \rangle$ onde $\mathcal{V}_\pi : \Upsilon_\pi \to 2^{\mathcal{W}}$. ■

É importante ressaltar que átomos em π-lógicas são termos em $\mathbb{P}$ e fórmulas cujo conectivo principal é $\boxdot_{\overline{\pi}}$, ou seja, fórmulas cujo conectivo principal é $\boxdot_{\overline{\pi}}$ são *interpretadas* como átomos na lógica $\mathcal{L}_\pi$.

Definição 35 (π-Subfórmulas)**.** O conjunto de *π-subfórmulas* de uma fórmula $\varphi \in \mathsf{LM}_\pi$, denotado por $\mathrm{Sub}_\pi(\varphi)$, é definido indutivamente como o menor conjunto que satisfaz as seguintes restrições:

$$\mathrm{Sub}_\pi(p_j) = \{p_j\}, p_j \in \Upsilon_\pi$$
$$\mathrm{Sub}_\pi(\circ\varphi) = \{\circ\varphi\} \cup \mathrm{Sub}_\pi(\varphi), \text{ sendo } \circ \in \{\Box_\pi, \neg\}$$
$$\mathrm{Sub}_\pi(\varphi \vee \psi) = \{\varphi \vee \psi\} \cup \mathrm{Sub}_\pi(\varphi) \cup \mathrm{Sub}_\pi(\psi)$$ ■

No que segue, serão apresentadas diversas funções que operam sobre fórmulas e que são necessárias para a prova de transferência.

Definição 36 (Funções sobre Fórmulas)**.** O conjunto de *π-átomos* de uma fórmula $\varphi \in \mathsf{LM}_\pi$, denotado por $\mathrm{C}_\pi(\varphi)$, é o conjunto de todas as π-subfórmulas de φ que são interpretadas como atômicas em LM_π. É definido como:

$$\mathrm{C}_\pi(\varphi) = \mathrm{Sub}_\pi(\varphi) \cap \Upsilon_\pi = \{\psi \mid \psi \in \mathrm{Sub}_\pi(\varphi) \text{ e } \psi \in (\mathbb{P} \cup \boxdot_{\overline{\pi}})\}$$

O conjunto de *sub-elementos* de uma fórmula $\varphi \in \mathsf{LM}_{12}$, denotado por $\mathrm{SC}(\varphi)$, é o conjunto de todas as subfórmulas de φ que são interpretadas como átomos, isto é, é o *conjunto de todas as subfórmulas de φ que são interpretadas como atômicas em LM_π ou $\mathsf{LM}_{\overline{\pi}}$*. É definido como:

$$\mathrm{SC}(\varphi) = \mathrm{Sub}(\varphi) \cap \Upsilon = \{\psi \mid \psi \in \mathrm{Sub}(\varphi) \text{ e } \psi \in (\mathbb{P} \cup \boxdot_1 \cup \boxdot_2)\}$$

[6] Podemos entender Υ_π como as fórmulas que são interpretadas como atômicas em $\mathcal{L}_\pi$.
[7] Análogo à Definição 13.
[8] Essa definição de lógicas pode ser entendida como uma versão mais restrita da definição de lógicas apresentada na Definição 25.

O conjunto de π-*sub-elementos* de uma fórmula $\varphi \in \mathsf{LM}_\pi$, denotado por $\mathrm{S}_\pi(\varphi)$, é o conjunto de todos os sub-elementos de π-átomos de φ, isto é, *é o conjunto de todas as subfórmulas interpretadas como atômicas das* π*-subfórmulas interpretadas como atômicas em* LM_π *de* φ. É definido como:

$$\mathrm{S}_\pi(\varphi) = \mathrm{SC}(\mathrm{C}_\pi(\varphi)) = \mathrm{SC}(\mathrm{Sub}_\pi(\varphi) \cap \Upsilon_\pi) = \mathrm{Sub}(\mathrm{Sub}_\pi(\varphi) \cap \Upsilon_\pi) \cap \Upsilon = \\ \{\gamma \mid \gamma \in \mathrm{Sub}(\psi) \text{ e } \gamma \in (\mathbb{P} \cup \boxdot_1 \cup \boxdot_2) \mid \psi \in \mathrm{Sub}_\pi(\varphi) \text{ e } \psi \in (\mathbb{P} \cup \boxdot_{\overline{\pi}})\}$$

Note que está definição é equivalente à:

$$\mathrm{S}_\pi(\varphi) = \mathrm{SC}(\mathrm{SC}(\varphi)) \cap \Upsilon_\pi$$

Ou seja, $\mathrm{S}_\pi(\varphi)$ "abre" fórmulas compostas porém interpretadas como atômicas em uma linguagem, permitindo analisar suas subfórmulas. Por exemplo, sendo $\varphi = \Box_2 p_1 \wedge \Box_1 \Box_2 p_0$, então $\mathrm{S}_1(\varphi) = \{\Box_2 p_1, \Box_2 p_0, p_1, p_0\}$, mesmo que $\Box_2 p_0$ e $\Box_2 p_1$ sejam interpretadas como atômicas em LM_1.

O conjunto de *elementos booleanos* de uma fórmula $\varphi \in \mathsf{LM}_{12}$, denotado por $\mathrm{TC}(\varphi)$, é o conjunto de todos os átomos (termos em $\mathbb{P}$) de φ que estão dentro do escopo de algum conectivo booleano. Por exemplo, considerando $\varphi = \Box_1 p_0 \to \Box_2 p_3$, temos que $\mathrm{TC}(\varphi) = \{p_0, p_1\}$, mas considerando $\psi = \Box_1 \Box_2 \Box_1 p_0$, temos que $\mathrm{TC}(\psi) = \varnothing$.

As funções descritas acima podem todas ser aplicadas à conjuntos da seguinte forma:

$$\mathrm{f}(\Gamma) = \{\mathrm{f}(\gamma) \mid \gamma \in \Gamma\}$$

Por fim, o conjunto dos *componentes booleanos* de um conjunto de fórmulas $\Gamma \subseteq \mathsf{LM}_{12}$, denotado por $\mathrm{B}(\Gamma)$, é definido como o fecho de Γ para os operadores booleanos, isto é, sendo $\Gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_n\}$ temos que $\mathrm{B}(\Gamma) = \{\neg\gamma_0, \ldots, \neg\gamma_n, \gamma_0 \wedge \gamma_1, \ldots, \gamma_0 \vee \gamma_1, \ldots, \gamma_0 \to \gamma_1, \ldots\}$[9]. ■

Definição 37 (π-Profundidade de Fórmulas)**.** A π-profundidade $\mathrm{d}^0_\pi(\varphi)$ de uma fórmula $\varphi \in \mathsf{LM}_\pi$, representa a quantidade de modalidades $\Box_\pi$ aninhadas em φ e fora do escopo de qualquer $\Box_{\overline{\pi}}$. É definida como:

$$\mathrm{d}^0_\pi(p_j) = 0, \text{ se } p_j \in \Upsilon_\pi$$
$$\mathrm{d}^0_\pi(\neg\varphi) = \mathrm{d}^0_\pi(\varphi)$$
$$\mathrm{d}^0_\pi(\varphi \vee \psi) = \max(\mathrm{d}^0_\pi(\varphi), \mathrm{d}^0_\pi(\psi))$$
$$\mathrm{d}^0_\pi(\Box_\pi\varphi) = 1 + \mathrm{d}^0_\pi(\varphi)$$

Já a π-profundidade estendida $\mathrm{d}_\pi(\varphi)$ de uma fórmula $\varphi \in \mathsf{LM}_{12}$, representa a quantidade de modalidades $\Box_\pi$ aninhadas em φ, independente de instâncias de $\Box_{\overline{\pi}}$. Sua definição é análoga à

[9] Essa operação irá criar diversas fórmulas que são equivalentes (semanticamente e sintaticamente), por exemplo: $\gamma_0 \wedge \gamma_1$ e $\gamma_1 \wedge \gamma_0$.

anterior, alterando a primeira regra para $\mathrm{d}_\pi(p_j) = 0$, se $p_j \in \mathbb{P}$ e adicionando a regra: $\mathrm{d}_\pi(\Box_{\overline{\pi}}\varphi) = \mathrm{d}_\pi(\varphi)$. Para conjuntos de fórmulas Γ, $\mathrm{d}^0_\pi(\Gamma)$ é definido como:

$$\mathrm{d}^0_\pi(\Gamma) = \begin{cases} 0 & \text{Se } \Gamma = \varnothing \\ max\left(\left\{\mathrm{d}^0_\pi(\gamma) \mid \gamma \in \Gamma\right\}\right) & \text{Se existe máximo em } \left\{\mathrm{d}^0_\pi(\gamma) \mid \gamma \in \Gamma\right\} \\ \omega & \text{Caso contrário} \end{cases}$$

Onde ω denota a cardinalidade de $\mathbb{N}$. A definição é análoga para d_π. ■

Exemplo 6 (Aplicação das Funções Sobre Fórmulas)**.** Considerando a seguinte fórmula:

$$\varphi = \Box_1(p_0 \vee p_1) \rightarrow \Box_2(p_2 \wedge (\Box_1 p_0 \vee \Box_2 p_1))$$

e considerando:

$$q_0 \mapsto \boldsymbol{p}_{(\Box_2(p_2 \wedge (\Box_1 p_0 \vee \Box_2 p_1)))}$$
$$q_1 \mapsto \boldsymbol{p}_{(\Box_1(p_0 \vee p_1))}$$
$$q_2 \mapsto \boldsymbol{p}_{(\Box_1 p_0)}$$
$$q_3 \mapsto \boldsymbol{p}_{(\Box_2 p_1)}$$

obtemos os seguintes resultados ao aplicar as funções $\{\mathrm{Sub}_\pi, \mathrm{C}_\pi, \mathrm{SC}, \mathrm{S}_\pi, \mathrm{TC}\}$ em φ:

$$\begin{aligned} \mathrm{Sub}_1(\varphi) =& \{p_0, p_1, q_0, \Box_1(p_0 \vee p_1), \Box_1(p_0 \vee p_1) \rightarrow q_0\} \\ \mathrm{Sub}_2(\varphi) =& \{p_1, p_2, q_1, q_2, \Box_2 p_1, q_2 \vee \Box_2 p_1, p_2 \wedge (q_2 \vee \Box_2 p_1), \Box_2(p_2 \wedge (q_2 \vee \Box_2 p_1)), \\ & q_1 \rightarrow \Box_2(p_2 \wedge (q_2 \vee \Box_2 p_1))\} \\ \mathrm{C}_1(\varphi) =& \{p_0, p_1, q_0\} \\ \mathrm{C}_2(\varphi) =& \{p_1, p_2, q_1, q_2\} \\ \mathrm{SC}(\varphi) =& \{p_0, p_1, p_2, q_0, q_1, q_2, q_3\} \\ \mathrm{S}_1(\varphi) =& \{p_0, p_1, p_2, q_0, q_3\} \\ \mathrm{S}_2(\varphi) =& \{p_0, p_1, p_2, q_1, q_2\} \\ \mathrm{TC}(\varphi) =& \{p_0, p_1, p_2\} \end{aligned}$$ ■

Definição 38 (Caminhos e Distâncias)**.** Sendo $\mathcal{F}$ um frame, uma sequência de mundos $w_0, \ldots, w_n \in \mathcal{W}, n \geq 0$ é chamado de um *$\mathcal{R}$-caminho*, ou apenas *caminho*, de w_0 para w_n, com comprimento n, se $\forall i, 0 \leq i < n$, existe $\langle w_i, w_{i+1}\rangle$ tal que $w_i \mathcal{R} w_{i+1}$. Sendo $\mathcal{F}_\pi$ um π-frame, escrevemos $w_0 \underset{\pi}{\rightarrow} w_n$ para denotar que há um *$\mathcal{R}_\pi$-caminho* de w_0 para w_n.

A *distância* entre dois mundo w_i e w_j, representada por $\mathrm{dist}(w_i, w_j)$, é o menor $n \geq 0$ tal que há um caminho de comprimento n entre w_i e w_j, caso não exista então $\mathrm{dist}(w_i, w_j) = \omega$. ■

Sendo $\mathcal{F}_\pi$ um π-frame, escrevemos $\mathrm{dist}(w_i, w_j)$ para representar a π-distância entre w_i e w_j em casos onde não for necessário explicitar π. A importância semântica do conceito de distância é fácil de observar com o seguinte exemplo: sendo φ uma fórmula tal que $\mathrm{d}^0_\pi(\varphi) \neq 0$, então a valoração de φ em um mundo w de um π-modelo $\mathcal{M}$ depende dos mundos x tal que $\mathrm{dist}(w, x) \leq \mathrm{d}^0_\pi(\varphi)$.

Definição 39 (Fecho Modal)**.** Para todo conjunto $\Gamma \subseteq \mathsf{LM}_{12}$, chamamos $\mathrm{DC}_\pi(\Gamma)$ de $\Box_\pi$*-fecho conservador de grau* de Γ e definimos como o menor conjunto que satisfaz as seguintes regras:

1. $\Gamma \subseteq \mathrm{DC}_\pi(\Gamma)$;
2. Sendo $\gamma \in \mathrm{DC}_\pi(\Gamma)$ e $\psi \in \Gamma$, se $\mathrm{d}_\pi(\gamma) < \mathrm{d}_\pi(\psi)$ então $\Box_\pi\gamma \in \mathrm{DC}_\pi(\Gamma)$.

Alternativamente, podemos definir $\mathrm{DC}_\pi(\Gamma)$ como o conjunto de todas as fórmulas da forma $\Box^n_\pi\gamma$, onde $\gamma \in \Gamma$, $\Box^n\gamma = \underbrace{\Box\ldots\Box}_{n}\gamma$ e $\mathrm{d}_\pi(\Box^n_\pi\gamma) < \mathrm{d}_\pi(\Gamma)$. ■

Definição 40 (Espaço de Fórmulas)**.** Um conjunto de fórmulas Θ é chamado de *espaço de fórmulas* se Θ é fechado para a operação de subfórmulas e para a operação $\mathrm{B}(\Gamma)$ de componentes booleanos, apresentada na Definição 36. ■

Por fim, temos a definição de completude generalizada:

Definição 41 (Completude Generalizada)**.** Uma lógica $\mathcal{L}$ é dita *completa com relação a um conjunto de fórmulas* Γ se, e somente se, para todo $\Delta \subseteq \Gamma$, onde Δ é $\mathcal{L}$-consistente, Δ é satisfazível em um frame para $\mathcal{L}$. ■

Esta definição de completude é uma generalização da definição apresentada na Proposição 1. Considerando a classe $\mathfrak{F} = \{\mathcal{F} \mid \mathcal{F} \Vdash \mathcal{L}\}$[10] e considerando $\Gamma = \mathsf{LM}_{12}$, temos que a Definição 41 descreve exatamente a completude forte como apresentada na Proposição 1. Já, se mantivermos a mesma classe $\mathfrak{F}$ mas considerarmos $\Gamma \subseteq \mathsf{LM}_{12}$ e Γ finito, temos exatamente a completude fraca. Podemos então enunciar o teorema principal desta seção:

Teorema 4 (Transferência de Completude)**.** *Sendo $\mathcal{L}_1$ e $\mathcal{L}_2$ 1- e 2-lógicas definidas sobre LM_{12} e sendo Θ um espaço de fórmulas. Se $\mathcal{L}_1$ é completo com relação a $\mathrm{DC}_1(\Theta)$ e $\mathcal{L}_2$ completo com relação a $\mathrm{DC}_2(\Theta)$, então $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$ é completo com relação a Θ.* ■

No que segue, é apresentada uma prova informal deste teorema, retirada de (FINE; SCHURZ, 1996, p. 178). A prova formal deste teorema encontra-se no Apêndice A.

Ideia da Prova do Teorema 4. Sendo $\Gamma \subseteq \Theta$ um conjunto $\mathcal{L}_{12}$-consistente, devemos provar que Γ é verdadeiro em algum mundo de um 12-modelo baseado em um 12-frame para $\mathcal{L}_{12}$. Para isso, será apresentado um algoritmo que constrói este modelo, com base em certos 1- e 2-modelos baseados em 1- e 2-frames para $\mathcal{L}_1$ e $\mathcal{L}_2$ respectivamente, os quais tornam certos subconjuntos de $\mathrm{DC}_1(\Theta)$ e $\mathrm{DC}_2(\Theta)$ verdadeiros.

Como Γ é $\mathcal{L}_{12}$-consistente, Γ também é $\mathcal{L}_1$-consistente e $\mathcal{L}_2$-consistente. Ademais, como $\Gamma \subseteq \Theta \subseteq \mathrm{DC}_\pi(\Theta)$, existe um π-modelo, baseado em um π-frame para $\mathcal{L}_\pi$, onde Γ é verdadeiro em algum mundo (devido à premissa de completude de $\mathcal{L}_1$ e $\mathcal{L}_2$). Este modelo é chamado de *modelo inicial* e é denotado por $\mathcal{I}$, já o mundo onde Γ é verdadeiro é chamado de *mundo inicial*

[10] Abuso de notação para indicar que $\mathcal{L}$ é válida no frame $\mathcal{F}$.

e é denotado por $\boldsymbol{i}$. O mundo $\boldsymbol{i}$ tem um papel importante em $\boldsymbol{\mathcal{I}}$, logo, ele também recebe o nome de *mundo base* de $\boldsymbol{\mathcal{I}}$.

No restante da prova, lidaremos com π-*modelos etiquetados*, estes são triplas da forma $\langle \mathcal{M}, w_{\mathcal{M}}, \Sigma(\mathcal{M}) \rangle$, onde $\mathcal{M}$ é um modelo, $w_{\mathcal{M}}$ é um mundo de $\mathcal{M}$, chamado de mundo base e $\Sigma(\mathcal{M})$ é o conjunto de elementos de $\mathcal{M}$, que é fechado sob a função SC, onde o conjunto de elementos de $\boldsymbol{\mathcal{I}}$ é dado por $\mathrm{SC}(\Gamma)$. Para cada mundo w em $\mathcal{M}$ é atribuído um conjunto de elementos $\Sigma_{\mathcal{M}}(w)$ de acordo com a seguinte definição indutiva:

1. $\Sigma_{\mathcal{M}}(w_{\mathcal{M}}) = \Sigma(\mathcal{M})$;
2. Se $w_i \mathcal{R}_\pi w_j$ e $\Box_\pi \varphi \in \Sigma_{\mathcal{M}}(w_i)$ então $\mathrm{SC}(\varphi) \in \Sigma_{\mathcal{M}}(w_j)$.

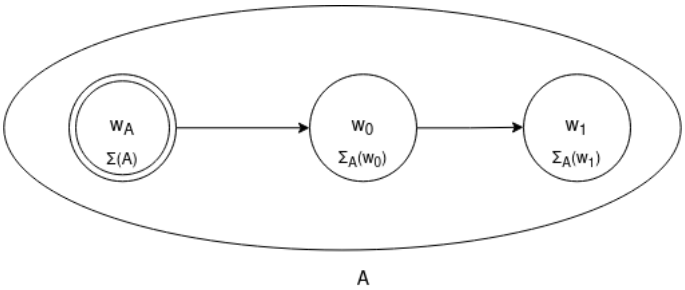

Figura 1 – Um exemplo de um modelo etiquetado chamado A, onde w_A é seu mundo base
Fonte: Elaborado pelo autor.

Considerando, para qualquer mundo w_j em $\boldsymbol{\mathcal{I}}$, os conjuntos $\mathrm{S}_\pi(\Sigma_{\boldsymbol{\mathcal{I}}}(w_j))$ de π-átomos de $\Sigma_{\boldsymbol{\mathcal{I}}}(w_j)$, esses conjuntos contém todos os elementos (fórmulas pertencentes à Υ) de fórmulas em $\Sigma_{\boldsymbol{\mathcal{I}}}(w_j)$. Estes conjuntos são chamados de *conjuntos de concordância*, pois com base neles construiremos novos $\overline{\pi}$-modelos etiquetados $\mathcal{N}$ cujo mundo base é w_j e que "concordam" com a valoração dos elementos dos conjuntos $\mathrm{S}_\pi(\Sigma_{\boldsymbol{\mathcal{I}}}(w_j))$.

Para isso, chamaremos o conjunto $\{\varsigma \mid \varsigma \in \mathrm{S}_\pi(\Sigma_{\boldsymbol{\mathcal{I}}}(w_j))\} \cup \{\neg\varsigma \mid \varsigma \notin \mathrm{S}_\pi(\Sigma_{\boldsymbol{\mathcal{I}}}(w_j))\}$ de elementos que são verdadeiros ou cuja negação é verdadeira em w_j, de *diagramas de concordância* e devemos garantir que estes conjuntos são $\mathcal{L}_{12}$-consistentes. Para garantir a consistência destes conjuntos, definimos os conjuntos

$$\mathrm{T}_\pi(\Delta) = \{\Box_\pi^n \delta \mid \delta \in \mathrm{B}(\mathrm{S}_\pi(\Delta)) \cap \mathcal{L}_{12} \text{ e } d_\pi(\Box_\pi^n \delta) \leq d_\pi(\Delta)\}$$

chamados de π-teorias e impomos a restrição que $\mathrm{T}_\pi(\mathrm{S}_\pi(\Sigma(\boldsymbol{\mathcal{I}})))$ deve ser verdadeiro em $\boldsymbol{i}$. A π-teoria de um dado conjunto Δ é o conjunto de todos os $\Box_\pi^n$-fechos de teoremas que são componentes booleanos de fórmulas em Δ, ou seja, é o conjunto de todas as fórmulas da forma $\Box_\pi^n \varphi$, tal que φ é um teorema de $\mathcal{L}_{12}$ e tem forma $\neg\psi$ ou $\psi_0 \vee \psi_1$, e n é menor ou igual ao comprimento da maior sequência de modalidades $\Box_\pi$ de alguma fórmula no conjunto Δ. Como Γ é $\mathcal{L}_{12}$-consistente e $\mathrm{T}_\pi(\mathrm{S}_\pi(\Sigma(\boldsymbol{\mathcal{I}}))) \subseteq \mathcal{L}_{12}$, $\Gamma \cup T_\pi(\mathrm{S}_\pi(\Sigma(\boldsymbol{\mathcal{I}})))$ é $\mathcal{L}_{12}$-consistente.

Para cada mundo w_j no conjunto de mundos de $\boldsymbol{\mathcal{I}}$, o diagrama de concordância de w_j é $\mathcal{L}_{12}$-consistente e está contido em Θ, logo, existe um $\overline{\pi}$-modelo etiquetado $\mathcal{N}$ baseado num frame

para $\mathcal{L}_{\overline{\pi}}$ que torna este diagrama verdadeiro em algum mundo. Chamaremos de w_x este mundo de $\mathcal{N}$, tomaremos ele como o mundo base para $\mathcal{N}$ e assumimos que este é o único mundo que $\mathcal{I}$ e $\mathcal{N}$ compartilham. O conjunto de elementos de $\mathcal{N}$ é o conjunto $\mathrm{S}_{\pi}(\Sigma_{\mathcal{I}}(w_x))$, que é o conjunto concordância de w_x. Por fim, $\mathcal{N}$ deve tornar a $\overline{\pi}$-teoria de $\Sigma(\mathcal{N}) = \mathrm{T}_{\overline{\pi}}(\mathrm{S}_{\overline{\pi}}(\Sigma(\mathcal{I})))$, verdadeira em $w_{\mathcal{N}}$.

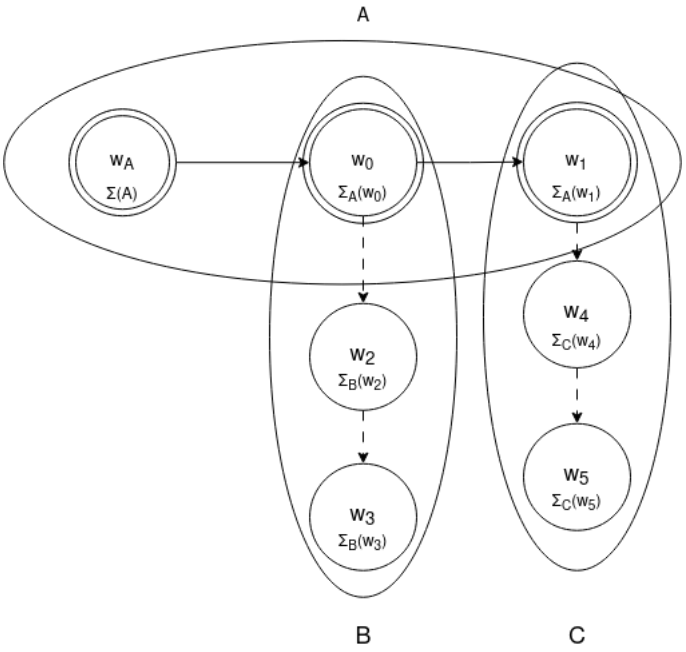

Figura 2 – Modelos ancorados em A, onde B e C são do tipo diferente que A, w_0 é o mundo base de B e w_1 é o mundo base de C
Fonte: Elaborado pelo autor.

Caso todas estas condições sejam satisfeitas, $\mathcal{N}$ é dito *ancorado em* $\mathcal{I}$ no mundo w_x. Ancoramos em $\mathcal{I}$ $\overline{\pi}$-modelos etiquetados disjuntos (ou seja, que não compartilham mundos entre si) em cada mundo do modelo. Então, cada mundo de $\mathcal{I}$ é o mundo base de algum $\overline{\pi}$-modelo etiquetado ancorado em $\mathcal{I}$. Adicionalmente, podemos ancorar π-modelos etiquetados nos $\overline{\pi}$-modelos, ancorados em $\mathcal{I}$, em mundos que não são base. Esta construção pode ser repetida infinitamente. É importante ressaltar que, exceto o modelo etiquetado ancorado em $\boldsymbol{i}$ em $\mathcal{I}$, nenhum modelo etiquetado será ancorado no mundo base de outro modelo etiquetado. Também é importante ressaltar que modelos etiquetados de um tipo só são ancorados em modelos etiquetados de outro tipo, ou seja, π-modelos etiquetados só são ancorados em $\overline{\pi}$-modelos etiquetados, e vice versa.

Essa relação de ancoramento constrói uma árvore de modelos, onde a raiz é $\mathcal{I}$ e novas folhas são inseridas ancorando modelos em algum modelo folha da árvore. Para então avaliar uma fórmula, devemos percorrer a árvore avaliando as subfórmulas da fórmula em modelos compatíveis com a fórmula[11]. Logo, temos uma forma de avaliar, passo a passo, fórmulas que

[11] Ou seja, caso a fórmula seja do tipo $\Box_1\varphi$ ela é avaliada em um 1-modelo e caso a fórmula seja do tipo $\Box_2\varphi$ ela é avaliada em um 2-modelo.

contenham modalidades distintas. Para então construir um 12-modelo que satisfaça o conjunto Γ, devemos considerar 1- e 2-modelos ancorados uns nos outros que satisfazem as fórmulas $\gamma \in \Gamma$. O conjunto de todo conjunto de modelos que satisfaz as fórmulas em Γ será chamado de conjunto de *brotos* de $\mathcal{I}$.

O lema de Zorn (ZORN, 1935) nos diz que o conjunto de brotos tem um elemento máximo. A união dos elementos (mundos, relações entre mundos e valorações) do elemento máximo do conjunto de brotos nos dará o 12-modelo desejado, do qual podemos obter um 12-frame para $\mathcal{L}_{12}$. Isso quer dizer que, na árvore de modelos gerada pela relação de ancoramento, existirá um caminho (sequência de modelos) que satisfaz todas as fórmulas do conjunto Γ, podemos então unir todos os modelos deste caminho em um único modelo, do qual podemos obter um frame para $\mathcal{L}_{12}$.

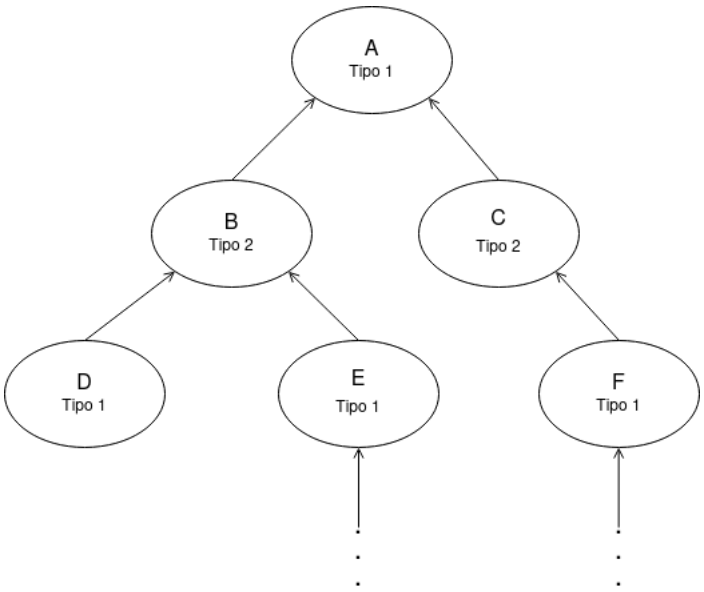

Figura 3 – Árvore de modelos com raiz em A, onde uma seta indica um ancoramento
Fonte: Elaborado pelo autor.

Isso conclui a prova informal do Teorema 4. ■

A prova formal do Teorema 4 encontra-se no apêndice A.

Teorema 5 (Transferência de Completude Forte)**.** *Sendo $\mathcal{L}_1$ e $\mathcal{L}_2$ lógicas modais fortemente completas com relação às classes de frames $\mathfrak{F}_1$ e $\mathfrak{F}_2$, então $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$ será fortemente completa com relação a classe de frames $\mathfrak{F}_{12} = \mathfrak{F}_1 \otimes \mathfrak{F}_2$.* ■

Prova do Teorema 5. Completude forte é exatamente completude generalizada com relação ao espaço de fórmulas LM_{12}. No enunciado do Teorema 4, tome $\Theta := \mathsf{LM}_{12}$, assim, $\mathrm{DC}_1(\mathsf{LM}_{12}) = \mathrm{DC}_2(\mathsf{LM}_{12}) = \mathsf{LM}_{12}$, logo, pelo Teorema 4, provamos o Teorema 5. ■

Teorema 6 (Transferência de Completude Fraca)**.** *Sendo $\mathcal{L}_1$ e $\mathcal{L}_2$ lógicas modais fracamente completas com relação as classes de frames $\mathfrak{F}_1$ e $\mathfrak{F}_2$, $\mathcal{L}_{12} = \mathcal{L}_1 \odot \mathcal{L}_2$ será fracamente completa com relação a classe de frames $\mathfrak{F}_{12} = \mathfrak{F}_1 \otimes \mathfrak{F}_2$.* ■

Prova do Teorema 6. Completude fraca é exatamente completude com relação à todo conjunto $\{\varphi\}$ onde $\varphi \in \mathsf{LM}_{12}$. Considerando um φ qualquer, no enunciado do Teorema 4, tome $\Theta = \mathrm{B}(\mathrm{Sub}(\varphi))$. Como o conjunto definido por $\mathrm{Sub}(\varphi)$ é finito por definição, $\Theta = \mathrm{B}(\mathrm{Sub}(\varphi))$ contém uma quantidade finita de formulas cujo valor verdade é diferente entre si[12]. Portanto, existe uma quantidade finita de fórmulas em $\mathrm{DC}_\pi(\Theta)$ que não são equivalente entre si em (um frame/modelo qualquer para) $\mathcal{L}_\pi$. Então, todo subconjunto $\Delta \subseteq \mathrm{DC}_\pi(\Theta)$ é equivalente em (um frame/modelo qualquer para) $\mathcal{L}_\pi$ a um conjunto de fórmulas finito Δ_f e, portanto, é equivalente em (um frame/modelo qualquer para) $\mathcal{L}_\pi$ a uma única fórmula (por exemplo $\bigwedge \Delta_f$), que será verdadeira no mundo i no modelo $\mathcal{I}$, pela Definição 46.

Portanto, $\mathcal{L}_\pi$ é completo com relação a $\mathrm{DC}_\pi(\Theta)$, logo, pelo Teorema 4, $\mathcal{L}_{12}$ é completa com relação a Θ e, portanto, é completo com relação a $\{\varphi\}$. ■

Este método de prova é suficientemente genérico para provar a transferência de outras propriedades, sendo necessário apenas pequenas variações em em componentes da prova, como demonstrado em Fine e Schurz (1996).

3.2 O SISTEMA BIMODAL $\mathbf{KT} \odot \mathbf{K4}$

O sistema bimodal $\mathbf{KT} \odot \mathbf{K4}$ é talvez um dos sistemas multimodais mais simples, ele consiste da fusão dos sistemas monomodais **KT** e **K4**. Este sistema será descrito formalmente nesta seção, pois foi usado como estudo de caso para a modelagem de fusão de lógicas modais em Coq, na biblioteca descrita no Capítulo 5.

3.2.1 Linguagem, Semântica e Axiomatização

A linguagem $\mathsf{LM}_{\mathcal{T}4}$ do sistema $\mathbf{KT} \odot \mathbf{K4}$ é baseada na assinatura $\mathsf{C} = \{\Box_T, \Diamond_T, \Box_4, \Diamond_4, \neg, \wedge, \vee, \rightarrow\}$ e é definida abaixo:

Definição 42 (Linguagem de $\mathbf{KT} \odot \mathbf{K4}$)**.** A linguagem do sistema é $\mathbf{KT} \odot \mathbf{K4}$ definida indutivamente como o menor conjunto que satisfaz as seguintes regras:

$\top, \bot \in \mathsf{LM}_{\mathcal{T}4}$

$\mathbb{P} \subseteq \mathsf{LM}_{\mathcal{T}4}$

Se $\varphi \in \mathsf{LM}_{\mathcal{T}4}$, então $\circ\varphi \in \mathsf{LM}_{\mathcal{T}4}, \circ \in \{\Box_T, \Box_4, \Diamond_T, \Diamond_4, \neg\}$

Se $\varphi, \psi \in \mathsf{LM}_{\mathcal{T}4}$, então $\varphi \circ \psi \in \mathsf{LM}_{\mathcal{T}4}, \circ \in \{\wedge, \vee, \rightarrow\}$ ■

Definição 43 ($\mathcal{T}4$-frames e $\mathcal{T}4$-modelos)**.** Um $\mathcal{T}4$-frame é uma tripla da forma $\mathcal{F}_{\mathcal{T}4} = \langle \mathcal{W}, \mathcal{R}_\mathcal{T}, \mathcal{R}_4 \rangle$, onde $\mathcal{W} \neq \varnothing$, $\mathcal{R}_\mathcal{T}, \mathcal{R}_4 \subseteq \mathcal{W} \times \mathcal{W}$ e $\mathcal{R}_\mathcal{T}$ é uma relação reflexiva e $\mathcal{R}_4$ é uma relação transitiva.

Um $\mathcal{T}4$-modelo é um par da forma $\mathcal{M}_{\mathcal{T}4} = \langle \mathcal{F}_{\mathcal{T}4}, \mathcal{V} \rangle$, onde $\mathcal{V} : \mathbb{P} \rightarrow 2^\mathcal{W}$ e $\mathcal{F}_{\mathcal{T}4}$ é um $\mathcal{T}4$-frame. ■

[12] Veja Definição 36.

Definição 44 (Valoração de Fórmulas)**.** A valoração de fórmulas em $\mathbf{KT} \odot \mathbf{K4}$ é definida de forma igual à valoração de fórmulas para lógicas monomodais, com os casos para $\Box$ e $\Diamond$ substituídos por:

$$\mathcal{M}_{\mathcal{T}4}, w_0 \Vdash \Box_T \varphi \text{ sse } \forall w_1 \in \mathcal{W}, (w_0 \mathcal{R}_T w_1 \rightarrow \mathcal{M}_{\mathcal{T}4}, w_1 \Vdash \varphi)$$
$$\mathcal{M}_{\mathcal{T}4}, w_0 \Vdash \Diamond_T \varphi \text{ sse } \exists w_1 \in \mathcal{W}, (w_0 \mathcal{R}_T w_1 \wedge \mathcal{M}_{\mathcal{T}4}, w_1 \Vdash \varphi)$$
$$\mathcal{M}_{\mathcal{T}4}, w_0 \Vdash \Box_4 \varphi \text{ sse } \forall w_1 \in \mathcal{W}, (w_0 \mathcal{R}_4 w_1 \rightarrow \mathcal{M}_{\mathcal{T}4}, w_1 \Vdash \varphi)$$
$$\mathcal{M}_{\mathcal{T}4}, w_0 \Vdash \Diamond_4 \varphi \text{ sse } \exists w_1 \in \mathcal{W}, (w_0 \mathcal{R}_4 w_1 \wedge \mathcal{M}_{\mathcal{T}4}, w_1 \Vdash \varphi)$$ ■

Outros conceitos semânticos são definidos de maneira análoga.

Definição 45 (Axiomatização de $\mathbf{KT} \odot \mathbf{K4}$)**.** O conjunto de axiomas $\Lambda_{\mathsf{LM}_{\mathcal{T}4}}$ para o sistema $\mathbf{KT} \odot \mathbf{K4}$ é uma especificação do conjunto Λ_{LM_n} multimodal, onde as n instâncias do axioma K_i são substituídas por:

$$\Box_T(p_0 \rightarrow p_1) \rightarrow (\Box_T p_0 \rightarrow \Box_T p_1) \qquad (K_T)$$
$$\Box_4(p_0 \rightarrow p_1) \rightarrow (\Box_4 p_0 \rightarrow \Box_4 p_1) \qquad (K_4)$$

e com a adição dos axiomas T:

$$\Box_T \varphi \rightarrow \varphi$$

e *4*:

$$\Box_4 \varphi \rightarrow \Box_4 \Box_4 \varphi$$

O conjunto de regras de derivação é definido como $\mathfrak{R} = \{Nec_T, Nec_4, MP\}$. Logo, o sistema $\mathbf{KT} \odot \mathbf{K4}$ pode ser axiomatizada pelo par $\langle \Lambda_{\mathsf{LM}_{\mathcal{T}4}}, \{Nec_T, Nec_4, MP\} \rangle$. ■

3.2.2 Corretude e Completude

Como demonstrado por Blackburn, Rijke e Venema (2001), os sistemas monomodais **KT** e **K4** são ambos corretos e fortemente completos, sendo **KT** correto e fortemente completo para a classe $\mathfrak{F}_{\mathcal{T}}$ de frames reflexivos e sendo **K4** correto e fortemente completo para a classe $\mathfrak{F}_4$ de frames transitivos. Portanto, pelos Teoremas 3 e 5, temos que o sistema $\mathbf{KT} \odot \mathbf{K4}$ é correto e fortemente completo para a classe $\mathfrak{F}_{\mathcal{T}4}$ de 2-frames da forma $\langle \mathcal{W}, \mathcal{R}_{\mathcal{T}}, \mathcal{R}_4 \rangle$, sendo $\mathcal{R}_{\mathcal{T}}$ uma relação reflexiva e $\mathcal{R}_4$ uma relação transitiva.

4 ASSISTENTES DE PROVAS

Segundo Geuvers (2009), assistentes de provas[1] são sistemas computacionais que permitem usuários definirem, provarem propriedades e raciocinarem sobre teorias e objetos matemáticos. Harrison, Urban e Wiedijk (2014) descrevem como um sistema que um humano usa interativamente para produzir provas formais. Já Silva (2019) descreve como programas que auxiliam no desenvolvimento de provas formais, mas não as fazem automaticamente e necessitam de um humano para guiar a prova.

Segundo Harrison, Urban e Wiedijk (2014) o conceito de assistentes de provas surge durante os anos de 1960 e 1970, a partir do reconhecimento que a capacidade de provadores automáticos de teoremas estavam estagnando, apesar de grande atividade e inovação na área. Provadores automáticos de teoremas, ao contrário de assistentes de provas, são softwares que automaticamente geram, se possível, provas para expressões matemáticas, geralmente por um método de busca em um domínio específico. De acordo com Harrison, Urban e Wiedijk (2014) e Geuvers (2009), durante este período surgiram os primeiros assistentes de provas, dentre eles estão Automath, LCF, Mizar e PVS.

De acordo com Silva (2019), provas assistidas por computador inicialmente não foram bem recebidas pela comunidade matemática, pois o objetivo de uma prova é convencer um leitor que algo é verdadeiro e explicar o porquê, algo que provas desenvolvidas em computador não são bem adaptadas, pois, em geral, suas linguagens internas não são de fácil entendimento. Porém, diversos resultados importantes foram provados em assistentes de provas, segundo Geuvers (2009), o teorema da Curva de Jordan foi provado em ambos Mizar e HOL Light, o Teorema dos Números Primos foi provado em Isabelle e o Teorema das Quatro Cores foi provado em Coq.

É de interesse analisar mais este último teorema, pois foi o primeiro grande resultado provado em um computador. Em específico, o problema foi detalhado e alguns casos foram provados em Appel e Haken (1976) e outros casos, além de uma prova em computador (em linguagem *assembly* para computadores IBM 370), foi apresentada em Appel, Haken e Koch (1977). O motivo da prova ter sido realizada em um computador é devido à quantidade de casos a serem analisados, cerca de 1 bilhão de acordo com Gonthier (2005).

A prova de Appel, Haken e Koch não foi imediatamente aceita pela comunidade matemática, apenas com a publicação de uma monografia definitiva da prova em Appel e Haken (1989) e com a prova de Robertson et al. (1997), também em computador mas muito mais simples e feita na linguagem C, o Teorema das Quatro Cores foi considerado provado (GONTHIER, 2005). Por fim, Gonthier (2005) formalizou a prova de Robertson et al. em Coq, dando, o que o autor descreve, como o passo final na prova. Ao longo dos anos, provas formalizadas em computador conseguiram reconhecimento, pois se demonstraram mais confiáveis que provas tradicionais.

Assistentes de provas também são amplamente usados para verificar componentes de software, segundo Avigad, Moura e Kong (2017) *verificação formal* consiste no uso de métodos

[1] Também chamados de provadores semi-automáticos de teoremas ou provadores interativos.

matemáticos e computacionais para expressar afirmações em termos matemáticos precisos. Essas afirmações podem se referir à corretude de certos sistemas de software ou hardware ou à garantia de que um certo estado de um programa nunca será atingido, estes que, ao serem expressos em termos matemáticos, podem ser demonstrados por meio de assistentes de provas.

Segundo Silva (2019), assistentes de provas tem propriedades interessantes, por exemplo, verificação rápida e mecânica de provas, comandos de busca de teoremas e lemas e automatização de provas. Uma propriedade que alguns assistentes de provas compartilham (por exemplo Coq (TEAM, 2022) e Lean (AVIGAD; MOURA; KONG, 2017; MOURA et al., 2015)) é satisfazer critério de de Bruijn. O critério afirma que o programa verificador de um assistente (seu *kernel* ou *núcleo*) deve ser um programa muito pequeno, que pode ser verificado manualmente, seja por meio de papel e caneta ou por um algoritmo simples que uma pessoa cética poderia desenvolver facilmente (BARENDREGT; GEUVERS, 2001). O verificador deve gerar algum tipo de "objeto de prova", por exemplo um termo λ, que pode ter sua veracidade comprovada da mesma forma que o verificador de tipos.

Neste capítulo, será apresentado o conceito de teoria de tipos e o assistente de provas Coq será descrito. Na Seção 4.1 será brevemente apresentada a Teoria de Tipos e na Seção 4.2 será descrito o assistente de provas Coq.

4.1 UMA BREVE APRESENTAÇÃO DE TEORIA DE TIPOS

Teoria dos tipos surge, de acordo com Coquand (2022), como uma tentativa de Russell para lidar com o Paradoxo de Russell: sendo $R = \{x \mid x \notin x\}$ então $R \in R \leftrightarrow R \notin R$, presente na Teoria de Conjuntos. O cerne do paradoxo se encontra na quantificação de x na definição de R, em específico, x é quantificado sobre todos os conjuntos, o que inclui o próprio conjunto R, possibilitando então que x seja instanciado como R.

Em uma carta para Frege escrita em 1902, Russell comunica a descoberta do paradoxo e indica que o problema está relacionado com a quantificação de objetos; Frege responde à Russell menos de uma semana depois, indicando que uma possível solução para o problema lidaria com "níveis" de objetos, assim impedindo que um objeto de um dado nível possa quantificar sobre objetos de níveis diferentes (HEIJENOORT, 2002). Porém, uma proposta de solução ao paradoxo surge apenas no apêndice B de Russell (1903)[2] onde Russell introduz provisoriamente a "Doutrina de Tipos", o que eventualmente se tornou a Teoria de Tipos moderna.

Após explorar diversas outras teorias para lidar com os paradoxos da Teoria de Conjuntos (HEIJENOORT, 2002), Russell voltou a trabalhar na teoria de tipos, resultando na Teoria de Tipos Ramificada (RUSSELL, 1908). Porém, problemas presentes nesta teoria[3] levaram Ramsey (1931) a desenvolver a Teoria de Tipos Simples, que é uma versão mais restrita da Teoria de

[2] É de interesse ressaltar a seguinte passagem do livro, presente no Apêndice A, página 762: "*A variable will not be able, except in special cases, to extend from one of these sets into another; and in $x \in u$, the x and the u must always belong to different types*[...]".

[3] Para mais detalhes sobre, veja (HEIJENOORT, 2002, p. 150-153).

Tipos Ramificada.

A Teoria de Tipos Simples foi reformulada por Church (CHURCH, 1940) para desenvolver a sua própria versão da teoria de tipos, usando a sintaxe do Cálculo-λ, o que hoje é chamado de Cálculo-λ Simplesmente Tipado (HUET, 1990). Na sua formulação, novos tipos são definidos indutivamente a partir de dois tipos básicos, (i para indivíduos e o para proposições) e de uma regra de definição de tipos (sendo A e B tipos, então toda função $A \to B$ de A para B é um tipo). Predicados tem tipo $i \to o$ e funções tem tipo $i \to i$. Por fim, expressões são definidas a partir de uma linguagem de Cálculo-λ estendida para lidar com tipos.

Segundo Howard (1980), Curry e Feys (1958) apontaram uma correspondência direta entre axiomas do fragmento positivo implicativo de lógica proposicional e combinadores do Cálculo-λ, já Tait (1967) observou uma correspondência direta entre a regra de eliminação do corte em Cálculo de Sequentes e redução de Termos λ. Howard (1980) então demonstrou, com base nestes resultados, que há uma correspondência direta entre a Lógica Intuicionista (descrita sintaticamente como um cálculo de sequentes) e o Cálculo-λ Simplesmente Tipado, dando surgimento ao que hoje é chamado de Correspondência de Curry-Howard[4].

Na década de 1970 foi desenvolvido um novo tipo de Cálculo-λ tipado, chamado de Sistema F ou Cálculo-λ de Segunda Ordem, independentemente por Girard (1971) e Reynolds (1974). Este sistema estende o Cálculo-λ Simplesmente Tipado adicionando uma operação de abstração de tipos, permitindo definir termos λ com variáveis que podem ser instanciadas como um tipo (GIRARD; TAYLOR; LAFONT, 1989). Isso se mostrou especialmente útil para a computação, em específico para a formalização do conceito de funções polimórficas.

Pouco tempo após o desenvolvimento do Sistema F, Girard apresentou o Sistema F_ω (GIRARD, 1972) que é uma generalização do Sistema F, adicionando os conceitos de construtores de tipos e *kinds*[5]. Construtores de tipos são tipos especiais de funções que recebem tipos e retornam tipos, já *kinds* podem ser entendidos como "tipos de tipos", essencialmente é uma cópia do Cálculo-λ Simplesmente Tipado a nível de tipos (PIERCE, 2002). As abstrações de tipos do Sistema F pode ser entendida como uma dependência de termos em tipos, já os construtores de tipos do Sistema F_ω pode ser entendidos como uma dependência entre tipos.

Outro importante desenvolvimento na teoria de tipos foi a Teoria de Tipos Intuicionista ou Teoria de Tipos de Martin-Löf (MARTIN-LÖF, 1975; MARTIN-LÖF; SAMBIN, 1984), proposta por Martin-Löf como um sistema lógico fundacional para a matemática construtivista, de forma análoga à Teoria de Conjuntos de Zermelo-Frankel para a matemática clássica. Uma das características deste sistema é uma distinção rigorosa entre proposições e julgamentos; proposições são objetos que podem ser combinados por meio de operações lógicas como $\neg, \wedge, \vee, \to, \exists, \forall$, já julgamentos são afirmações sobre proposições. Por exemplo, na frase "φ é verdadeiro", φ é uma proposição e "φ é verdadeiro" é um julgamento (MARTIN-LÖF; SAMBIN,

[4] Essa noção é expandida por Wadler (WADLER, 2015) onde o autor defende que a Correspondência é apenas um nome para um fenômeno maior chamado de *Propositions as Types* (Proposições como Tipos).

[5] Pode ser traduzido como "gênero".

1984).

A Teoria de Tipos de Martin-Löf está intimamente relacionada à Correspondência de Curry-Howard, pois toda proposição em Teoria de Tipos de Martin-Löf representa um tipo, o tipo de todas as provas da proposição. Por exemplo, a frase "971 é um número não primo" descreve a proposição que é provada fornecendo dois números naturais maiores que 1 e uma computação que mostra que seu produto é 971. Por outro lado, todo tipo também descreve uma proposição, em específico, a proposição que o tipo em questão não é vazio, que é provada fornecendo um objeto que habita o tipo (MARTIN-LÖF, 1975). Ao provar uma proposição em Teoria de Tipos de Martin-Löf, construímos um objeto que habita o tipo desta proposição, estes objetos são chamados de *objetos de prova* e são ditos serem *testemunhas* para a verdade da proposição (DYBJER; PALMGREN, 2023).

Duas noções importantes para a Teoria de Tipos são introduzidas na Teoria de Tipos de Martin-Löf, são elas o Universo de Tipos e Tipos Dependentes. Universo de tipos é uma forma de lidar com a necessidade de criar tipos arbitrários dentro da linguagem. Existem duas famílias de tipos: os tipos pequenos e os tipos grandes, estas respeitam a seguinte hierarquia bem fundada: $V_0 \in V_1 \in V_2 \in \cdots \in V_n \in \ldots$[6], sendo V_0 a família dos tipos pequenos e toda família $V_i, i \geq 1$ dos tipos grandes de ordem i, onde todo V_n tem tipo V_{n+1} (MARTIN-LÖF, 1975). É importante ressaltar que não é possível impor a restrição que um tipo seja um objeto dele mesmo, ou seja que V tenha tipo V, pois isso levaria a uma contradição semelhante ao Paradoxo de Russell (GIRARD, 1972). Uma variável de tipo em uma expressão é dita dependente se o tipo a qual ela será instanciada depende do valor de algum termo na expressão, nesse caso o tipo de toda a expressão é dito dependente (MARTIN-LÖF, 1975).

Segundo Team (2022), Coquand e Huet uniram o sistema F_ω e a Teoria de Tipos de Martin-Löf para criar o Cálculo de Construções (CoC), que é um sistema lógico baseado em teoria de tipos intuicionista com abstrações de tipos, construtores de tipos e tipos dependentes capaz de descrever indiretamente definições indutivas. Mais ainda, segundo Paulin-Mohring (2015), o Cálculo de Construções é um sistema de tipos puro, ou seja, é um tipo de Cálculo-λ com uma única linguagem que descreve ambos tipos e termos.

Na Figura 4, é apresentado um diagrama proposto originalmente por Barendregt (1991) para demonstrar relações entre diversos sistemas de tipos modernos. Cada flecha indica uma relação de inclusão ($\subseteq$) entre seus vértices, ou seja, o vértice de origem de uma flecha está incluso no vértice de destino da mesma. Uma das principais ideias no diagrama é representar as possíveis dependências mútuas entre tipos e termos, da seguinte forma:

- Tipos dependerem de termos ($\rightarrow$)
- Termos dependerem de tipos ($\uparrow$)
- Tipos dependerem de tipos ($\nearrow$)

[6] Aqui considere que a notação $a \in B$ indica que B é o tipo do objeto a.

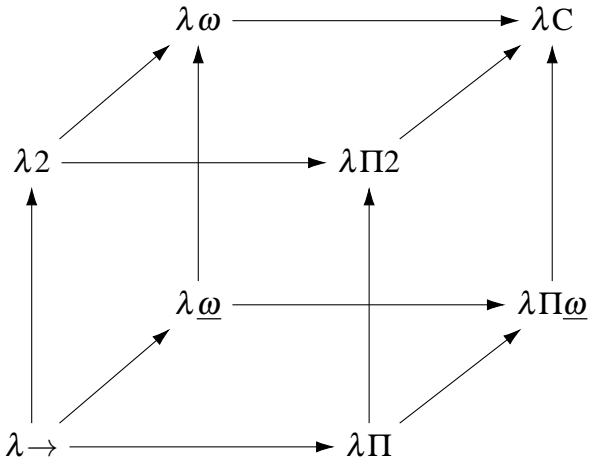

Figura 4 – Cubo λ[7]
Fonte: Adaptado de (BARENDREGT, 1991).

Como termos dependerem de outros termos é um componente básico do Cálculo-λ, isso não é representado. Os sistemas de tipos representados são os seguintes:

- $\lambda\rightarrow$: Cálculo-λ Simplesmente Tipado;
- $\lambda 2$: Sistema F;
- $\lambda\omega$: Sistema F_ω;
  - $\lambda\underline{\omega}$: Uma versão mais fraca do Sistema F_ω;
- $\lambda\Pi$: *Logical Framework*, um sistema de Cálculo-λ com tipos dependentes (HARPER; HONSELL; PLOTKIN, 1993);
- λC: Cálculo de Construções.

O Cálculo de Construções foi estendido por Coquand e Paulin (1988) com a inclusão de definições indutivas primitivas, o que gerou o Cálculo de Construções Indutivas (CCI) (TEAM, 2022). Este sistema de tipos é o formalismo por trás do Coq, todo objeto definido dentro do Coq é traduzido para um termo do CCI.

Como apresentado em Team (2022), no CCI, todo objeto tem um tipo, incluindo os próprios tipos. O tipo dos tipos se chama *sort*[8], e há uma hierarquia infinita e bem-fundada de *sorts*, onde os *sorts* base são *Set* e *Prop*. Usaremos a notação $A : B$ para denotar que o objeto A tem tipo B.

[7] Diagrama originalmente produzido por Artem Pelenitsyn, de título original "Lambda Cube", gratuitamente disponível em <https://www.overleaf.com/latex/examples/lambda-cube/drnrtqnnjfzf>, licenciado pela licença Creative Commons CC BY 4.0 <https://creativecommons.org/licenses/by/4.0/deed.pt_BR>.

[8] Pode ser traduzido adequadamente como "classe".

O *sort Set* é o tipo de todos os tipos pequenos, isto inclui tipos como o tipo dos números naturais e o tipo de termos booleanos, e também produtos, subconjuntos e funções sobre estes tipos. O *sort Set* é o menor dos tipos.

O *sort Prop* é o tipo de proposições lógicas, onde um objeto $\mathbb{P} : Prop$ descreve uma classe de termos que representam provas de $\mathbb{P}$, já um objeto $p : \mathbb{P}$ é dito uma *testemunha* para a provabilidade da proposição $\mathbb{P}$. Objetos em *Prop* que não tem testemunhas são contradições. O *sort SProp* é uma extensão do Coq originalmente proposta por Gilbert et al. (2019) que define uma classe de proposições estritas, estas são proposições cujas provas são irrelevantes (todas as provas são iguais). Ambos os *sorts Prop* e *SProp* são impredicativos, isto é, uma função que quantifica sobre todos os *Props*/*SProps* é também do tipo *Prop*/*SProp*. Formalmente, segundo Crosilla (2017), uma definição é impredicativa se define uma entidade por referência a alguma totalidade à qual a própria entidade pertence, e é predicativa caso contrário.

Como *Set* ser do tipo *Set* gera uma teoria inconsistente, a linguagem do Cálculo de Construções Indutivas tem infinitos *sorts Type(i)*, $i \geq 1$. Em particular, os *sorts Set*, *Prop* e *SProp* são do tipo *Type(1)* e $\forall i, Type(i) : Type(i+1)$. Os tipos são cumulativos, ou seja, para qualquer x, se $x : Type(i)$ então $x : Type(i+1)$, de forma análoga ao que ocorre na Teoria de Tipos de Martin-Löf.

4.2 O ASSISTENTE COQ

Coq é um assistente de provas para lógica de ordem superior desenvolvido desde 1984 por membros do instituto francês INRIA e seu *kernel* é baseado no CCI. De acordo com Silva (2019), a Correspondência de Curry-Howard é o que dá versatilidade suficiente para o Coq para expressar lógicas mais sofisticadas, o autor ainda afirma que parte considerável da matemática pode ser expressa em Coq.

Segundo Silva (2019), dentro da Ciência da Computação, Coq é usado principalmente como uma ferramenta para verificação de programas, como é o caso de Leroy (2021), que desenvolveu um compilador para a linguagem C totalmente verificado em Coq, chamado de CompCert. Coq também pode ser usado para formalizar sistemas lógicos, como é o caso de Silveira et al. (2022) e Wind (2001) que descrevem a lógica modal alética em Coq e Paiva e Caleiro (1998) que descreve a lógica temporal em Coq.

De acordo com Silva (2019), Coq é tanto uma linguagem de programação funcional quanto uma linguagem de provas. Mais especificamente, Coq é composto dos seguintes componentes:

- *Gallina*, que é uma linguagem de especificação e programação funcional, onde todo programa termina;
- *Vernacular*, que é uma linguagem de comandos, permite interagir com o *kernel* do Coq;
- Um conjunto de táticas para realizar provas, que são traduzidas para termos em *Gallina*;

- $\mathcal{L}$tac, que é uma linguagem para implementação de táticas.

Para utilizar o Coq de forma interativa, ferramentas como o CoqIDE[9], VSCoq[10] ou o ProofGeneral[11] podem ser utilizadas. No que segue, serão apresentados exemplos feitos no CoqIDE, porém o mesmo se aplica às outras ferramentas.

Para realizar uma prova em Coq usa-se os comandos `Theorem, Lemma, Example` ou `Corollary`[12] para declarar uma expressão que será provada. Provas são iniciadas com o comando `Proof` e finalizadas com o comando `Qed` ou o comando `Defined`, mais ainda, provas podem ser aceitas sem serem completas com o comando `Admitted` e interrompidas com o comando `Abort`.

Coq permite a definição indutiva de novos tipos, assim como a definição de funções recursivas e não recursivas. Tipos indutivos, definidos pelo comando `Inductive`, são definidos analogamente como na matemática "tradicional", onde é definido um conjunto de casos base para a indução e regras para a criação de novos objetos do tipo, onde ambos casos base e regras são chamados de construtores do tipo. Funções recursivas podem ser definidas pelo comando `Fixpoint`, estas operam sobre objetos de tipos indutivos podendo ter comportamentos diferentes dependendo de qual construtor de tipo foi usado para definir o objeto. Por fim, Coq também permite a definição de objetos arbitrários, com o comando `Definition`, que não passam da atribuição de um nome a algum termo (PIERCE et al., 2021).

No desenvolvimento de provas interativas com o Coq uma prova pode ser verificada "passo a passo" pelo (*kernel* do) Coq enquanto é desenvolvida (isto é, antes de ser finalizada). As ferramentas apresentadas anteriormente dispõem de duas janelas, uma janela de edição de texto, onde a prova de fato é escrita e uma janela de visualização, onde o estado da prova pode ser acompanhado, como pode ser visto com o caso do CoqIDE na Figura 5. O estado de uma prova é o conjunto de todas as hipóteses desta prova, assim como o conjunto de todos os termos que devem ser provados, chamados de *goals* ou objetivos (TEAM, 2022). O *goal* pode ser separado em diversos *subgoals* com a aplicação de algumas táticas, onde cada *subgoal* deve ser provado para terminar a prova. A janela de visualização é dividida em duas partes por uma linha de inferência. Acima da linha estão as hipóteses e abaixo da linha está o *goal*, como pode ser visto na Figura 5.

Durante o desenvolvimento interativo de provas, linhas são destacadas pela ferramenta que está sendo utilizada, a partir da resposta dada pelo verificador de tipos sobre o conteúdo daquela linha. Linhas destacadas em verde não geraram qualquer erro e foram aceitas pelo verificador de tipos, linhas destacadas em amarelo geraram algum tipo de aviso e linhas em vermelho foram rejeitadas pelo verificador de tipos, isto é, estão erradas de alguma forma.

Na Figura 6, as linhas 13 e 15 até 17 estão destacas em amarelo pois não foram verificadas pelo verificador de tipos do Coq, isso é devido aos comandos `Admitted` e `Axiom`, que fazem

[9] <https://coq.inria.fr/refman/practical-tools/coqide.html>.
[10] <https://github.com/coq-community/vscoq>.
[11] <https://proofgeneral.github.io/>.
[12] Internamente ao Coq todos são sinônimos, a distinção é apenas para os leitores.

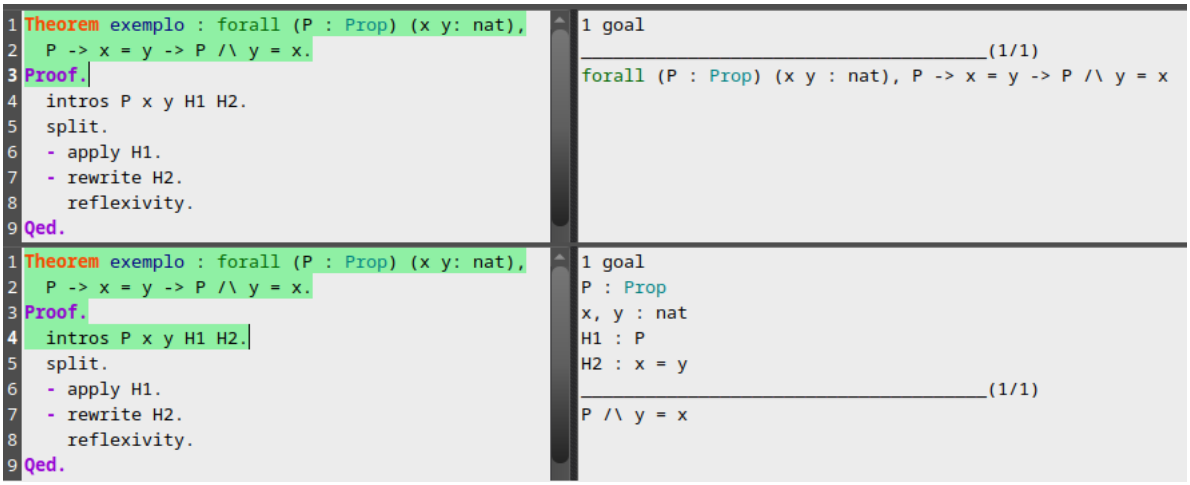

Figura 5 – Antes e depois de aplicar a tática `intros`.
Fonte: Elaborado pelo autor.

com que os termos não sejam verificados pelo Coq e sejam tratados como verdades. A linha 23 está destacada em vermelho pois ocorreu um erro nela, em específico, o Coq rejeitou a aplicação da tática `easy`. As restantes estão destacadas em verde pois foram aceitas pelo verificador de tipos.

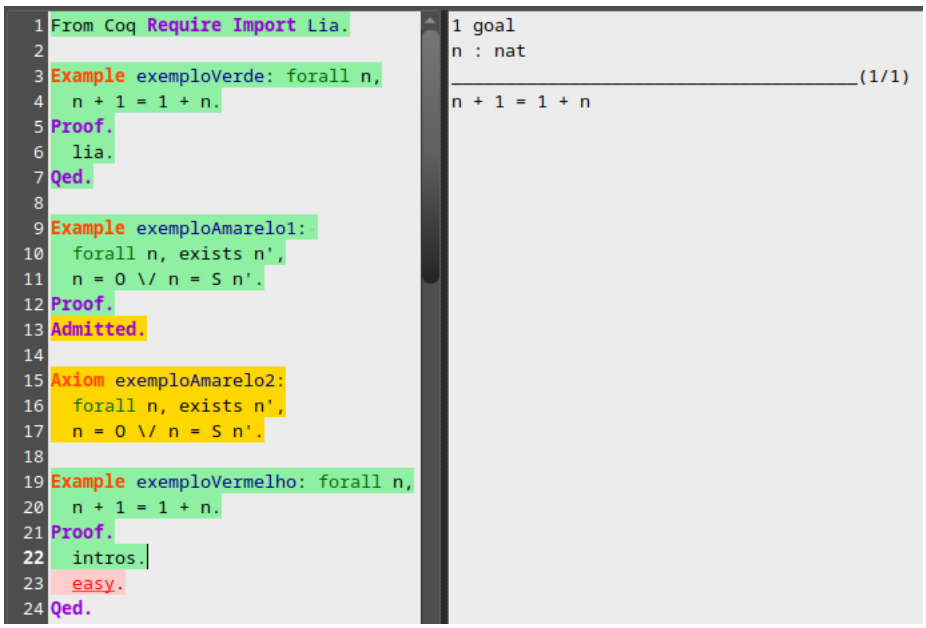

Figura 6 – Provas no CoqIDE.
Fonte: Elaborado pelo autor.

Dentro de provas no Coq, objetos são manipulados com táticas, por exemplo, a tática `intros` introduz termos quantificados universalmente no *goal* e termos à esquerda de implicações[13] para o conjunto de hipóteses, a tática `apply` modifica o *goal* ou as hipóteses com base

[13] Estes na realidade são um caso particular de quantificação universal onde não há dependência de tipos entre o termo quantificado e a proposição sobre a qual ele está quantificado (PIERCE et al., 2021).

no que está sendo aplicado e onde[14] e a tática `rewrite` reescreve equivalências no *goal* ou em hipóteses.

No que segue, conceitos de Coq serão apresentados por meio de exemplos práticos feitos pelo autor ou retirados de Pierce et al. (2021).

Exemplo 7 (Definições e Funções em Coq)**.** Abaixo é definido indutivamente o tipo "nat" que representa os número naturais. Um número natural ou é `O` (representa o número 0) ou é o sucessor de outro número natural, representado como a aplicação da função sucessor `S` a algum número. A função recursiva "plus" apresenta a definição recursiva da soma de dois números naturais, já a função "pred" descreve uma operação de predecessor de um número.

```
Inductive nat : Set :=
  | O
  | S: nat → nat.
Fixpoint plus (n : nat) (m : nat) : nat :=
  match n with
  | O ⇒ m
  | S n' ⇒ S (plus n' m)
  end.
Definition pred (n : nat) : nat :=
  match n with
  | O ⇒ O
  | S n' ⇒ n'
  end.
```

Definições indutivas também podem ser utilizadas para descrever relações, como é o caso da relação "ev", representando paridade de números, abaixo. Esta definição descreve uma relação que associa números naturais (`nat`) à provas (`Prop`).

```
Inductive ev : nat → Prop :=
  | ev_0 : ev 0
  | ev_SS (n : nat) (H : ev n) : ev (S (S n)).
```

O construtor `ev_SS` de `ev` requer *evidências* para ser aplicado, em específico, ele requer um número natural `n` e uma prova `H` de `ev n`, ou seja, uma prova que `n` é par. Ao aplicar `ev_SS` em algum termo, será necessário fornecer estas evidências para que verificador de tipos do Coq aceite a aplicação. ■

Para realizar provas em Coq, uma proposição é declarada, um nome é atribuído à ela e então um termo é construído a partir de axiomas, premissas e outros resultados já provados. Táticas são usadas para manipular estes e outros objetos relevantes para a prova até que esta seja

[14] Em específico, caso aplique-se um termo de tipo `A → B` numa hipótese de tipo `A`, seus tipos serão unificados e a hipótese resultante terá tipo `B`, ou seja, equivale a aplicar a regra de modus ponens; caso aplique-se um termo de tipo `A → B` num *goal* de tipo `B`, seus tipos serão unificados e o *goal* resultante terá tipo `A`, ou seja, é um tipo de *backwards reasoning* - para provar `B` basta provar `A`.

concluída. Quando concluída, o termo provado é incluído no universo de fatos do Coq e pode ser utilizada em provas futuras.

Exemplo 8 (Provas em Coq)**.** O teorema "exemplo1" mostra uma prova da comutatividade da disjunção lógica, já o lema "exemplo2" mostra uma prova indutiva que a soma de qualquer número natural com 0 é igual ao próprio número. Provas sobre relações indutivas podem ser feitas aplicando os construtores destas relações, no caso de `ev` do exemplo anterior, aplicando as regras `ev_0` e `ev_SS`, como apresentado no corolário "exemplo3" abaixo, que prova que 4 é um número par.

```
Theorem exemplo1: forall A B: Prop,
  A ∨ B → B ∨ A.
Proof.
  intros A B [HA | HB].
  - right. assumption.
  - left. assumption.
Qed.
Lemma exemplo2: forall n: nat,
  n + 0 = n.
Proof.
  intros n.
  induction n as [ | n' IHn].
  - reflexivity.
  - simpl. rewrite IHn. reflexivity.
Qed.
Corollary exemplo3: ev 4.
Proof. apply ev_SS. apply ev_SS. apply ev_0. Qed.
```

■

4.2.1 Automação em Coq

Há diversos comando de automação em Coq, cujo intuito é agilizar o desenvolvimento de provas. Estes comandos se dividem em duas categorias, os *tacticals* e as táticas de automação propriamente ditas. *Tacticals* são "táticas de ordem superior", táticas que recebem outras táticas como argumento (PIERCE et al., 2021). Alguns dos comandos mais comuns são apresentados abaixo.

Exemplo 9 (`try`)**.** O *tatical* `try` tenta executar uma tática $\mathbb{T}$, caso esta tenha sucesso, $\mathbb{T}$ é executada como normal, porém, caso $\mathbb{T}$ falhe, nada acontece, ao invés de ocorrer um erro.

```
Theorem exemplo5 : forall (P : Prop),
  P → P.
Proof.
```

```
  intros P HP.
  try reflexivity.
  apply HP.
Qed.
```

No teorema acima, apenas executar a tática `reflexivity` teria falhado, porém, como o *tatical* `try` foi utilizado, nada ocorre. ■

Exemplo 10 (Ponto e Vírgula)**.** O *tatical* ; (ponto e vírgula) tem duas formas, na sua forma mais simples, dadas duas táticas $\mathbb{T}_1$ e $\mathbb{T}_2$, $\mathbb{T}_1;\mathbb{T}_2$ irá executar a tática $\mathbb{T}_2$ em todos os resultados da aplicação da tática $\mathbb{T}_1$.

```
Fixpoint leb (n m : nat) : bool :=
  match n, m with
  | O, _       ⇒ true
  | _, O       ⇒ false
  | S n', S m' ⇒ leb n' m'
  end.
Lemma exemplo6: forall n,
  (leb 0 n) = true.
Proof.
  intros.
  destruct n; simpl; reflexivity.
Qed.
```

No exemplo acima, é definida uma função recursiva `leb` que verifica se um número é menor que outro, já no lema "exemplo6", é feita uma prova por análise de caso no termo `n` pela tática `destruct` e nos (dois) resultados desta tática é aplicada a tática `simpl`, em seguida, nos (dois) resultados da aplicação da tática `simpl`, é aplicada a tática `reflexivity`, que finaliza a prova.

Na sua forma genérica, o *tatical* ; pode ser aplicado a n+1 táticas $\mathbb{T}_0, \ldots \mathbb{T}_n$, usando a sintaxe $\mathbb{T}_0;[\mathbb{T}_1 \mid \mathbb{T}_2 \mid \ldots \mid \mathbb{T}_n]$, onde é inicialmente aplicada a tática $\mathbb{T}_0$ então, no seu primeiro resultado é aplicado $\mathbb{T}_1$, no seu segundo resultado é aplicado $\mathbb{T}_2$, ...e no seu n-ésimo resultado é aplicado $\mathbb{T}_n$. ■

Exemplo 11 (`repeat`)**.** O *tatical* `repeat` repete a aplicação de uma dada tática $\mathbb{T}$ até ela falhar. Caso $\mathbb{T}$ não seja aplicável (isto é, falha na primeira aplicação) não é aplicada nenhuma vez. Este *tatical* é ideal para ser combinado com os dois anteriores, como no seguinte exemplo:

```
Fixpoint In {A : Type} (x : A) (l : list A) : Prop :=
  match l with
  | [] ⇒ False
  | x' :: l' ⇒ x' = x ∨ In x l'
  end.
Theorem exemplo7:
```

```
  In 10 [1;2;3;4;5;6;7;8;9;10].
Proof.
  repeat (try (left; reflexivity); right).
Qed.
```

Neste exemplo, é definida uma função recursiva `In` que verifica se um argumento está numa lista, já no teorema "exemplo7" o *tatical* `repeat` é usado em combinação com os *taticals* ; e `try` para repetidamente verificar se 10 está na lista. ■

Por fim, temos três táticas que resolvem automaticamente uma grande variedade de provas, `auto`, `lia` e `intuition`.

Exemplo 12 (`auto`)**.** Segundo Pierce et al. (2021), `auto` implementa um algoritmo de busca de provas que tenta automaticamente resolver provas por meio de aplicações das táticas *intros* e *apply*[15]. Caso `auto` não consiga resolver uma prova, estado dela não será modificado. Há variantes de `auto`, como o `eauto` e o `rauto` que operam de maneira semelhante, mas utilizam táticas levemente diferentes[16]. É possível estender a base de fatos que `auto` e suas variantes podem usar em provas por meio do comando `Hint`. Também é possível indicar o uso de algum fato específico pelo `auto` por meio do comando `using`.

```
Example exemplo8: forall P Q R S T U : Prop,
  (P → Q) → (P → R) →
  (T → R) → (S → T → U) →
  ((P → Q) → (P → S)) →
  T → P → U.
Proof.
  auto.
Qed.

Lemma le_antisym : forall n m: nat, (n <= m ∧ m <= n) → n = m.
Proof. Admitted.
Example exemplo9: forall n m p : nat,
  (n <= p → (n <= m ∧ m <= n)) →
  n <= p → n = m.
Proof.
  auto using le_antisym.
Qed.
```

Neste exemplo, o lema "le_antisym" é assumido correto (será provado no próximo exemplo) e é usado com o `auto` para provar o exemplo "exemplo9".

[15] Na realidade, usa uma versão mais fraca do *apply* (TEAM, 2022).
[16] Em específico, utilizam as variantes *eapply* e *rapply* da tática *apply*.

A tática `auto` tem uma variante chamada `trivial` que não busca recursivamente por possíveis soluções e tenta aplicar apenas táticas de baixo custo (TEAM, 2022), é normalmente usada para provar termos que são instâncias de termos provados anteriormente. ■

Exemplo 13 (`lia`)**.** A tática `lia` implementa um algoritmo de decisão para um subconjunto da lógica de primeira ordem chamado de Aritmética de Presburger (PIERCE et al., 2021). A tática `lia` é capaz de decidir se uma fórmula envolvendo operação aritméticas e conectivos lógicos é verdadeira ou não. Caso seja verdadeira, uma aplicação de `lia` irá provar a fórmula, caso não seja ou caso o termo a ser provado não tenha apenas operações aritméticas, `lia` irá falhar.

```
From Coq Require Import Lia.

Lemma le_antisym1 : forall n m: nat, (n <= m ∧ m <= n) → n = m.
Proof. lia. Qed.
Example exemplo10: forall m n p,
  m + (n + p) = m + n + p.
Proof.
  intros. lia.
Qed.
```

Como `lia` não está na biblioteca padrão do Coq é necessário importar ela com o comando `From Coq Require Import Lia.` ■

Exemplo 14 (`inuition`)**.** A tática `intuition` implementa um algoritmo de decisão para lógica proposicional intuicionista capaz de provar qualquer instância de uma tautologia desta lógica (TEAM, 2022).

```
Lemma exemplo12: forall (P Q : Prop),
(Q → P) → (P → Q) → (P ↔ Q).
Proof. intuition. Qed.
```

Como é possível observar no exemplo, `intuition` é capaz de introduzir termos também, logo não é necessário usar a tática `intros` ■

5 IMPLEMENTAÇÃO

Os desenvolvimentos deste trabalho são baseados numa biblioteca em Coq versão 8.15.2 desenvolvida em Silveira (2020) e Silveira et al. (2022), que se encontra disponível em <https://github.com/funcao/LML/tree/master/coq>. A modelagem de lógica modal em Coq destes trabalhos é característica de um *deep embedding* este que, segundo Azurat e Prasetya (2002), é uma forma de modelar um sistema lógico $\mathcal{L}_0$ dentro de outro sistema lógico $\mathcal{L}_1$ representando completamente a sintaxe e semântica de $\mathcal{L}_0$.

Neste capítulo, será apresentada a biblioteca base sobre onde os desenvolvimentos deste trabalho foram feitos e os próprios desenvolvimentos serão descritos. Na Seção 5.1 será descrita a biblioteca de lógica modal que serviu de base para este trabalho, na Seção 5.2 será descrita a modelagem do sistema multimodal $\mathbf{KT} \odot \mathbf{K4}$ na biblioteca e na Seção 5.3 será descrita a modelagem genérica de lógicas multimodais na biblioteca.

5.1 A BIBLIOTECA EM COQ

A biblioteca é dividida em nove arquivos, dos quais oito são referentes a alguma parte da modelagem de lógica modal e um apresenta exemplo de uso da biblioteca. O arquivo `Modal_Library.v` apresenta as principais definições da biblioteca, como a definição de fórmulas da lógica modal, frames, modelos, valoração de fórmulas, consequência semântica e provas de diversas propriedades da consequência semântica. Fórmulas da lógica modal são representadas como tipo indutivo dentro do *sort Set*, frames e modelos são representados como estruturas do tipo *Record*, como descrito a seguir:

```
Inductive formula: Set :=
  | Lit    : nat → formula
  | Neg    : formula → formula
  | Box    : formula → formula
  | Dia    : formula → formula
  | And    : formula → formula → formula
  | Or     : formula → formula → formula
  | Implies: formula → formula → formula.
Record Frame: Type := {
  W: Set;
  R: W → W → Prop
}.
Record Model: Type := {
  F: Frame;
  v: nat → (W F) → Prop
}.
```

Para construir um frame ou modelo são usados os construtores `Build_Frame` e `Build_Model`, respectivamente. O arquivo `Modal_Notations.v` descreve uma notação para representar fórmulas da lógica modal e operações sobre fórmulas dentro do Coq de maneira mais conveniente. Por exemplo, uma fórmula como:

$$(p_0 \vee p_1) \rightarrow (p_0 \wedge \neg p_2) \rightarrow (\Box p_0 \rightarrow \Diamond p_1)$$

é representada dentro da biblioteca como (note os delimitadores):

```
[! (#0 \/ #1) -> (#0 /\ ~ #2) -> ([] #0 -> <> #1) !]
```

O arquivo `Logical_Equivalence.v` apresenta diversas provas de equivalências lógicas da lógica modal, por exemplo, provando a dualidade entre $\Box$ e $\Diamond$. O arquivo `Modal_Tactics.v` apresenta provas de diversos resultados úteis para desenvolver provas mais complexas. O arquivo `Frame_Validation.v` apresenta provas de corretude de frames, isto é, prova que se um frame respeita uma dada propriedade então o axioma correspondente desta propriedade será válido naquele frame e que se o axioma correspondente de uma propriedade for válido em um frame então este frame respeitará aquela propriedade.

O arquivo `Deductive_System.v` descreve o sistema axiomático de Hilbert para a lógica modal, onde axiomas são representados como um tipo indutivo dentro do *sort Set*, instâncias de axiomas são obtidas por meio de uma função de instanciação e deduções são representadas por uma relação definida indutivamente, como apresentado a seguir:

```
Inductive axiom : Set :=
  | ax1  : formula → formula → axiom
  | ax2  : formula → formula → formula → axiom
  (* algumas linhas foram omitidas *)
  | axK  : formula → formula → axiom
  | axPos : formula → formula → axiom
  (* algumas linhas foram omitidas *)
  | axGL : formula → axiom.
Definition instantiate (a: axiom): formula :=
  match a with
  | ax1  f0 f1   ⇒ [! f0 → (f1 → f0) !]
  | ax2  f0 f1 f2 ⇒ [! (f0 → (f1 → f2)) → ((f0 → f1) → (f0 → f2)) !]
  (* algumas linhas foram omitidas *)
  | axK  f0 f1   ⇒ [! []( f0 → f1) → ([] f0 → [] f1) !]
  | axPos f0 f1  ⇒ [! <> (f0 ∨ f1) → (<> f0 ∨ <> f1) !]
  (* algumas linhas foram omitidas *)
  | axGL f0      ⇒ [! [](  f0 → f0) → []f0 !]
  end.
Inductive deduction (A: axiom → Prop): theory → formula → Prop :=
  | Prem: forall (t: theory) (f: formula) (i: nat),
```

```
        (nth_error t i = Some f) → deduction A t f
| Ax: forall (t: theory) (a: axiom) (f: formula),
      A a → instantiate a = f → deduction A t f
| Mp: forall (t: theory) (f g: formula) (d1: deduction A t [! f → g !])
            (d2: deduction A t f), deduction A t g
| Nec: forall (t: theory) (f: formula) (d1: deduction A t f),
      deduction A t [! [] f !].
```

Os arquivos `Soundness.v` e `Completeness.v` apresentam provas de corretude e completude para a lógica modal, respectivamente. Para a prova de corretude, é provado que todos os axiomas do sistema de Hilbert são tautologias da lógica modal e que as regras de *Modus Ponens* e Necessitação preservam validade de fórmulas. Já para a prova de completude, é definido como construir conjuntos de Lindenbaum para então construir conjuntos maximais consistentes, porém, até o momento da escrita deste trabalho, a prova de completude não foi terminada.

5.2 O SISTEMA BIMODAL **KT** $\odot$ **K4** EM COQ

Para este trabalho, foi modelado em Coq versão 8.15.2 o sistema bimodal **KT** $\odot$ **K4**, descrito formalmente na Seção 3.2, como prova de conceito da possibilidade de modelar lógicas multimodais na biblioteca descrita anteriormente neste capítulo. O sistema foi implementado em três arquivos distintos que modelam, respectivamente, os componentes básicos do sistema (linguagem, frames e modelos, sistema axiomático), as propriedades semânticas do sistema (da valoração em mundos/modelos e da consequência semântica) e as propriedades sintáticas do sistema (da relação de consequência sintática).

O código desenvolvido seguiu os padrões de nomenclatura e estilo da biblioteca base, com o intuito de tornar este o mais próximo possível do que já foi desenvolvido. O código desenvolvido encontra-se disponível em: <https://github.com/funcao/LML/tree/Fusao>.

Para esta implementação, foram feitos vários pares de definições/funções/teoremas para representar certas propriedades sobre o sistema **KT** $\odot$ **K4** com respeito aos sistemas **KT** e **K4**, pares estes que são essencialmente iguais, portanto, em casos onde não for estritamente necessário apresentar os dois elementos de um par, será apresentado apenas um para evitar repetições desnecessárias. Para representar fórmulas da linguagem do sistema **KT** $\odot$ **K4**, foi necessário definir um tipo indutivo novo, análogo ao tipo das fórmulas para a linguagem modal básica, como apresentado a seguir:

```
Inductive KT4formula : Set :=
  | T4Lit   : nat       → KT4formula
  | T4Neg   : KT4formula → KT4formula
  | TBox    : KT4formula → KT4formula
  | TDia    : KT4formula → KT4formula
  | K4Box   : KT4formula → KT4formula
  | K4Dia   : KT4formula → KT4formula
```

```
| T4And    : KT4formula → KT4formula → KT4formula
| T4Or     : KT4formula → KT4formula → KT4formula
| T4Implies: KT4formula → KT4formula → KT4formula.
```

Não foi possível utilizar o tipo já definido de fórmulas, pois as fórmulas para **KT** $\odot$ **K4** contém duas modalidades distintas, já as fórmulas da lógica modal descritas na biblioteca básica contém apenas uma modalidade.

É definida uma notação para fórmulas de forma análoga ao que foi feito na biblioteca base, onde uma fórmula:

$$(p_0 \vee p_1) \rightarrow (p_0 \wedge \neg p_2) \rightarrow (\Box_T p_0 \rightarrow \Diamond_4 p_1)$$

é representada dentro da biblioteca como (note os delimitadores):

```
<! (#0 \/ #1) -> (#0 /\ ~ #2) -> ([T] #0 -> <4> #1) !>
```

Para definir frames e modelos de **KT** $\odot$ **K4**, foi necessário redefinir os conceitos de reflexividade e transitividade de relações pois, na biblioteca base, essas relações são definidas para frames com apenas uma relação de acessibilidade da seguinte forma:

```
Definition reflexivity_frame (F: Frame): Prop :=
  forall w, R F w w.
Definition transitivity_frame (F: Frame): Prop :=
  forall w0 w1 w2: W F, (R F w0 w1 ∧ R F w1 w2) → R F w0 w2.
```

O que não é compatível com o conceito de frames para **KT** $\odot$ **K4**, logo as seguintes definições foram feitas:

```
Definition reflexive_rel (X: Set) (R: X → X → Prop): Prop :=
  forall w, R w w.
Definition transitive_rel (X: Set) (R: X → X → Prop): Prop :=
  forall w0 w1 w2, (R w0 w1 ∧ R w1 w2) → R w0 w2.
```

E provadas corretas com as definições de anteriores, exclusivas para frames:

```
Theorem refl_equivalence: forall F W R,
  F = Build_Frame W R →
  reflexivity_frame F ↔ reflexive_rel W R.
Theorem trans_equivalence: forall F W R,
  F = Build_Frame W R →
  transitivity_frame F ↔ transitive_rel W R.
```

No trecho acima, a expressão `Build_Frame W R` denota o construtor do `Record Frame` apresentado anteriormente na Seção 5.1. Com estas definições é possível definir frames e modelos para **KT** $\odot$ **K4**:

```
Record KT4Frame: Type := {
  WT4: Set;
```

```
  RT: WT4 → WT4 → Prop;
  RT_refl: reflexive_rel WT4 RT;
  R4: WT4 → WT4 → Prop;
  R4_trans: transitive_rel WT4 R4
}.
Record KT4Model: Type := {
  FT4: KT4Frame;
  VT4: nat → (WT4 FT4) → Prop
}.
```

Também são dadas definições explícitas de frames e modelos para os sistemas **KT** e **K4**. É importante ressaltar que estes conceitos já existiam antes, mas de forma implícita:

```
Definition KTFrame (F: Frame) := reflexivity_frame F.
Definition KTModel (M: Model) := exists F v, M = [F -- v] ∧ KTFrame F.

Definition K4Frame (F: Frame) := transitivity_frame F.
Definition K4Model (M: Model) := exists F v, M = [F -- v] ∧ K4Frame F.
```

No trecho acima, a expressão `[F -- v]` é uma notação para o construtor `Build_Model` do `Record Model` apresentado anteriormente na Seção 5.1. É necessário provar que frames (ou modelos) para **KT** ⊙ **K4** são extensões dos mesmos conceitos para **KT** e **K4**, para isso é necessário demonstrar como obter um frame (modelo) de **KT** (**K4**) a partir de um frame (modelo) de **KT** ⊙ **K4**. Isso é feito definindo funções que constroem frames (modelos) de **KT** (**K4**) com os componentes de um frame (modelo) de **KT** ⊙ **K4**:

```
Definition KT4_frame_into_KT (F: KT4Frame): Frame :=
  match F with
  | Build_KT4Frame W RT RT_refl _ _ ⇒ Build_Frame W RT
  end.
Definition KT4_model_into_KT (M: KT4Model): Model :=
  match M with
  | Build_KT4Model (Build_KT4Frame W RT _ R4 _ as F) V ⇒
    Build_Model (KT4_frame_into_KT F) V
  end.
```

E provando que estas funções são corretas, isto é, que de fato transformam frames de um tipo em frames do outro tipo:

```
Theorem KT4_frame_into_KT_sound: forall F,
  KTFrame (KT4_frame_into_KT F).
Theorem KT4_model_into_KT_sound: forall M,
  KTModel (KT4_model_into_KT M).
```

A valoração de fórmulas de **KT** ⊙ **K4** é definida de forma análoga ao que foi definida para a biblioteca base:

```
Fixpoint KT4eval (M: KT4Model) (w: WT4 (FT4 M)) (f: KT4formula): Prop :=
  match f with
    | T4Lit     x       ⇒ VT4 M x w
    | TBox      f1      ⇒ forall w', RT (FT4 M) w w' → KT4eval M w' f1
    | TDia      f1      ⇒ exists w', RT (FT4 M) w w' ∧ KT4eval M w' f1
    | K4Box     f1      ⇒ forall w', R4 (FT4 M) w w' → KT4eval M w' f1
    | K4Dia     f1      ⇒ exists w', R4 (FT4 M) w w' ∧ KT4eval M w' f1
    | T4Neg     f1      ⇒ ¬KT4eval M w f1
    | T4And     f1 f2   ⇒ KT4eval M w f1 ∧ KT4eval M w f2
    | T4Or      f1 f2   ⇒ KT4eval M w f1 ∨ KT4eval M w f2
    | T4Implies f1 f2 ⇒ KT4eval M w f1 → KT4eval M w f2
  end.
```

São definidas duas funções que transformam fórmulas da biblioteca básica para fórmulas de **KT** ⊙ **K4** são elas `KTformula_to_KT4formula` e `K4formula_to_KT4formula`, isso é importante para provar que certas propriedades que **KT** e **K4** respeitam também são respeitadas por **KT** ⊙ **K4**. Também é provado que estas funções são injetivas:

```
Theorem KTformula_to_KT4formula_injective: forall f1 f2,
  KTformula_to_KT4formula f1 = KTformula_to_KT4formula f2 →
    f1 = f2.
```

Com a definição de valoração de fórmulas, é definido o conceito de validade em modelos, consequência semântica em um modelo e consequência semântica em todos os modelos, são provadas propriedades da valoração de fórmulas, das operações de validade e consequência em modelos e é provado que valoração de fórmulas (e validade e consequência) semântica em **KT** ⊙ **K4** é uma generalização dos casos para **KT** e **K4**.

```
Theorem KT_eval_generalization: forall W RT RT_refl R4 R4_trans V f,
  let FT4 := Build_KT4Frame W RT RT_refl R4 R4_trans in
  let MT4 := Build_KT4Model FT4 V in
  let M := KT4_model_into_KT MT4 in
  forall w, fun_validation M w f ↔
  KT4eval MT4 w (KTformula_to_KT4formula f).
```

O sistema sintático de **KT** ⊙ **K4** é definido com base num conjunto de axiomas, definido como um tipo indutivo, uma função que associa objetos deste tipo a fórmulas, de forma análoga ao que foi feito na biblioteca base e as regras de derivação do sistema de Hilbert são definidas como um tipo indutivo, tal qual na biblioteca base.

```
Inductive KT4axiom : Set :=
  | KT4ax1   : KT4formula → KT4formula → KT4axiom
  (* algumas linhas foram omitidas *)
  | KT4axK_T : KT4formula → KT4formula → KT4axiom
  | KT4axK_4 : KT4formula → KT4formula → KT4axiom
```

```
  | KT4axPos_T: KT4formula → KT4formula → KT4axiom
  | KT4axPos_4: KT4formula → KT4formula → KT4axiom
  | KT4axT   : KT4formula → KT4axiom
  | KT4axK4  : KT4formula → KT4axiom.

Definition KT4instantiate (a: KT4axiom): KT4formula :=
  match a with
  | KT4ax1    f0 f1   ⇒ <! f0 → (f1 → f0) !>
  (* algumas linhas foram omitidas *)
  | KT4axK_T  f0 f1   ⇒ <! [T](f0 → f1) → ([T] f0 → [T] f1)  !>
  | KT4axK_4  f0 f1   ⇒ <! [4](f0 → f1) → ([4] f0 → [4] f1)  !>
  | KT4axPos_T f0 f1  ⇒ <! <T> (f0 ∨ f1)  → (<T> f0 ∨ <T> f1) !>
  | KT4axPos_4 f0 f1  ⇒ <! <4> (f0 ∨ f1)  → (<4> f0 ∨ <4> f1) !>
  | KT4axT    f0      ⇒ <! [T]  f0 → f0 !>
  | KT4axK4   f0      ⇒ <! [4]  f0 → [4][4]  f0 !>
  end.
```

São provadas algumas propriedades adicionais da transformação de teorias dos sistemas **KT** e **K4** para o sistema **KT** $\odot$ **K4** e são provadas propriedades da derivação no sistema **KT** $\odot$ **K4**. Por fim, é provado que toda dedução nos sistemas **KT** / **K4** é possível de ser feita no sistema **KT** $\odot$ **K4**, por meio de predicados definidas indutivamente, que associam axiomas da biblioteca base para axiomas do sistema **KT** $\odot$ **K4**.

```
Theorem KTdeduction_generalization: forall G f,
  let KT_to_KT4 := KTformula_to_KT4formula in
  deduction T G f → KT4deduction KT4Ax (KT_theory_to_KT4theory G) (KT_to_KT4 f).
```

Por fim, é definido um predicado que descreve a corretude de um dado sistema axiomático com respeito a alguma classe de frames (representado como uma função de tipo `Frame → Prop`), para então provar que, assumindo a corretude dos sistemas **KT** e **K4**, o sistema **KT** $\odot$ **K4** é correto, ou seja, provar a transferência da corretude para este caso particular de fusão.

```
Definition relative_soundness (A: axiom → Prop) (P: Frame → Prop) :=
  forall G f, (A;  G |-- f)  → forall F V, P F  → entails (Build_Model F V) G f.
```

Este predicado é provado correto com relação à definição de corretude para o sistema **K** da biblioteca base, onde o predicado indutivo `anyFrame` descreve a classe de todos os frames[1]:

```
Inductive anyFrame (F: Frame): Prop := anyFrameMk: anyFrame F.

Lemma relative_soundness_correct:
  relative_soundness K anyFrame ↔
  (forall (G: theory) (f:  formula), (K;  G |-- f)  → (G ||=  f)).
```

[1] Este predicado poderia ser substituído por uma função que recebe um frame e retorna `True` independente do frame.

Com base nessa definição, foi provada a corretude dos sistemas **KT** e **K4**:

```
Theorem KT_soundness:
  relative_soundness T reflexivity_frame.

Theorem K4_soundness:
  relative_soundness K4 transitivity_frame.
```

Analogamente ao que foi feito anteriormente para o sistema **K**, foi definido um predicado que representa a corretude no sistema **KT** $\odot$ **K4** e uma função que descreve a classe de todo 2-frame reflexivo e transitivo:

```
Inductive anyKT4Frame (F: KT4Frame): Prop := anyKT4FrameMk: anyKT4Frame F.

Definition relative_KT4soundness (A: KT4axiom → Prop) (R: KT4Frame → Prop) :=
  forall G f, (A; G |--t4 f) → forall F V, R F → KT4entails (Build_KT4Model F V) G f.
```

Com essas definições, foi provada a corretude do sistema **KT** $\odot$ **K4**:

```
Theorem KT4_soundness:
  relative_KT4soundness KT4Ax anyKT4Frame.
```

Porém não foi possível completar essa prova com base nas premissas de que **KT** e **K4** são corretos. Essas premissas não levaram a nenhuma contradição, porém elas também não contribuíram na prova pois, devido a maneira que foram definidos os conceitos de consequência sintática e semântica e como as interpretações destes conceitos na biblioteca base e na prova de conceito foram relacionados, as premissas de que **KT** e **K4** são corretos geram outras premissas inúteis. A prova da corretude de **KT** $\odot$ **K4** foi feita provando (com grande auxílio das ferramentas de automação do Coq) cada caso da prova "manualmente", independente da hipótese de corretude de **KT** e **K4**.

5.3 LÓGICAS MULTIMODAIS

A segunda parte da implementação deste trabalho consiste na modelagem em Coq versão 8.15.2 de lógicas multimodais, descritas formalmente na Seção 2.6, também usando como base a biblioteca de lógica modal descrita anteriormente neste capítulo. Este sistema foi modelado em cinco arquivos distintos, dos quais quatro representam componentes de fato do sistema multimodal e um descreve exemplos de aplicação. Assim como na modelagem do sistema **KT** $\odot$ **K4**, o código desenvolvido seguiu os padrões de nomenclatura e estilo da biblioteca base e encontra-se disponível em: <https://github.com/funcao/LML/tree/Fusao>.

Inicialmente, fórmulas da linguagem multimodal foram modeladas como um tipo indutivo de maneira semelhante ao que foi feito na biblioteca base e na modelagem do sistema **KT** $\odot$ **K4**, porém a principal diferença entre a modelagem aqui descrita e as outras modelagens é que as fórmulas da linguagem multimodal que contém modalidades recebem um argumento adicional, que indica o índice da modalidade:

```
Inductive MMformula : Set :=
  | MMLit   : nat          → MMformula
  | MMNeg   : MMformula    → MMformula
  | MMBox   : nat          → MMformula → MMformula
  | MMDia   : nat          → MMformula → MMformula
  | MMAnd   : MMformula    → MMformula → MMformula
  | MMOr    : MMformula    → MMformula → MMformula
  | MMImplies: MMformula   → MMformula → MMformula.
```

A notação para fórmulas definida na prova de conceito é reaproveitada nesta seção, onde uma fórmula como:

$$(p_0 \vee p_1) \rightarrow (p_0 \wedge \neg p_2) \rightarrow (\Box_0 p_0 \rightarrow \Diamond_3 p_1)$$

É representada dentro da biblioteca como[2]:

```
<! (#0 \/ #1) -> (#0 /\ ~ #2) -> ([0] #0 -> <<3>> #1) !>
```

Note que esta definição indutiva não representa de fato a união das assinaturas de duas linguagens distintas, ou seja, não representa precisamente a definição de linguagens multimodais como foi descrito na Definição 30. Mais será comentado sobre isso na Seção 5.4.

Há dois motivos por trás da representação de modalidades com um índice: permitir que um único tipo indutivo seja capaz de representar fórmulas de qualquer sistema multimodal, independente de quantas modalidades ele tenha e garantir que uma fórmula (leia-se, um objeto que habita o tipo `MMformula`) não precise de alguma prova para ser construída, ou seja, não é necessário fornecer uma prova que o índice da modalidade de uma fórmula é menor que o número de modalidades num dado sistema modal para então poder construir uma fórmula. Porém, essa escolha carrega um problema: como garantir que, em um sistema multimodal com n modalidades, não seja possível operar sobre fórmulas que contenham modalidades cujo índice é maior que n?

Para resolver este problema foi utilizado o mecanismo de seções do Coq com o intuito de definir uma variável que representa a quantidade de modalidades em um dado sistema multimodal, assim como uma restrição de que há mais de uma modalidade no sistema multimodal.

```
Section MultiModal.
  Variable Modalities: nat.
  Hypothesis minimum_modalities: Modalities > 1.
  (* ... *)
End MultiModal.
```

A definição dessa variável não passa de uma conveniência pois apenas adiciona, no contexto de toda definição/tipo indutivo/prova dentro da seção, um termo do tipo `nat` chamado `Modalities`.

[2] Não foi possível representar a modalidade $\Diamond_x$ apenas por <x> pois a notação "? < x", onde x é um natural, é reservada para a operação de menor, que está definida na biblioteca padrão e portanto não pode ser sobrecarregada sem antes realizar outras alterações.

Ou seja, adiciona um argumento/premissa já instanciado, este que se torna um termo quantificado universalmente quando o objeto definido dentro da seção é invocado fora dela.

```
Require Import Nat.
Section MultiModal.
  Variable Modalities: nat.
  Definition foo (n: nat): nat :=
    match leb n Modalities with
    | true  ⇒ 0
    | false ⇒ 1
    end.
  Check foo. (*foo: nat → nat*)
  (*...*)
End MultiModal.
Check foo. (*foo: nat → nat → nat*)
```

A partir destas definições, foi implementada uma função recursiva que verifica se uma fórmula pode ser deduzida num sistema multimodal, onde uma fórmula φ pode ser deduzida num sistema multimodal com x modalidades se, e somente se, para toda subfórmula de φ da forma $\Box_i\psi$ ou $\Diamond_i\psi$, $i < x$. Uma função semelhante é definida para verificar dedutibilidade de teorias (listas de fórmulas).

```
Fixpoint deducible_formula (f: MMformula): Prop :=
  match f with
  | MMLit _                  ⇒ True
  | MMNeg f1                 ⇒ deducible_formula f1
  | MMBox x f1  | MMDia x f1 ⇒ x < Modalities ∧ deducible_formula f1
  | MMAnd f1 f2 | MMOr f1 f2 | MMImplies f1 f2 ⇒
    deducible_formula f1 ∧ deducible_formula f2
  end.
Fixpoint deducible_theory (T: MMtheory): Prop :=
  match T with
  | [ ]    ⇒ True
  | h :: t ⇒ (deducible_formula h) ∧ (deducible_theory t)
  end.
```

Frames e modelos para o sistema multimodal são definidos de forma semelhante ao que é feito na prova de conceito, onde um frame multimodal contém um conjunto de mundos, uma lista de relações de acessibilidade definidas sobre o conjunto de mundos e uma prova de que há tantas relações quanto há modalidades no sistema multimodal, já um modelo multimodal contém um frame multimodal e uma função de valoração definida sobre o conjunto de mundos do frame.

```
Record nFrame: Type :={
  nW      : Set;
```

```
  nR      : list (nW → nW → Prop);
  rel_cond: (length nR) = Modalities
}.
Record nModel: Type := {
  nF: nFrame;
  nV: nat → (nW nF) → Prop
}.
```

Para acessar uma relação arbitrária do conjunto (lista) de relações de um n-frame, foi definida uma função, chamada `get_rel`, que apenas encapsula a função `nth` da biblioteca padrão e foi definida uma relação (vazia) especial, chamada `dummyRel`, que é usada como o retorno padrão da função `nth`. É de interesse ressaltar que esta relação especial só será retornada pela função `get_rel` caso os outros argumentos passados a ela sejam inválidos, por exemplo caso seja passado um índice maior que o tamanho da lista ou seja passada uma lista vazia. Também foram provados alguns teoremas sobre a corretude da função `get_rel`.

```
Definition dummyRel (X: Set) := (fun x:  X ⇒ (fun y: X ⇒ False)).
Definition get_rel (X: Set) (lR: list (X → X → Prop)) (index: nat) :=
  nth index lR (dummyRel X).
Lemma get_rel_sound: forall X lR n,
  length lR < n → (get_rel X lR n) = (dummyRel X).
Lemma get_rel_sound2: forall X lR n,
  ¬ In (dummyRel X) lR → n < length lR → (get_rel X lR n) <> (dummyRel X).
```

Não foi possível representar a união de frames como foi descrito na Definição 30 (mais será comentado sobre isso na Seção 5.4), foram apenas definidas funções que separam um n-frame em um ou mais frames, algo semelhante foi definido para n-modelos:

```
Definition split_frame (F: nFrame): list Frame:=
  match F with
    | Build_nFrame nW nR _ ⇒ map (Build_Frame nW) nR
  end.
Definition nFrame_to_Frame (F: nFrame) (index: nat): Frame :=
  match F with
    | Build_nFrame nW nR _ ⇒ Build_Frame nW (get_rel nW nR index)
  end.
Definition nModel_to_Model (M: nModel) (index: nat): Model :=
  match M with
    | Build_nModel (Build_nFrame nW nR _) V ⇒
      Build_Model (Build_Frame nW (get_rel nW nR index)) V
  end.
```

Com isso então foi definida a valoração de fórmulas no sistema multimodal, onde só é possível valorar fórmulas caso elas sejam dedutíveis no sistema multimodal em questão e uma

fórmula que contém uma modalidade de índice x será valorada com base na x-ésima relação do frame do modelo em questão:

```
Fixpoint valuation (M: nModel) (w: nW (nF M)) (f: MMformula): Prop:=
  match f with
  | MMLit i     ⇒ (nV M) i w
  | MMNeg f1    ⇒ deducible_formula f1 →
    (¬ valuation M w f1)
  | MMBox i f1  ⇒ deducible_formula (MMBox i f1) →
    (forall w', (get_rel (nW (nF M)) (nR (nF M)) i) w w' → valuation M w' f1)
  | MMDia i f1  ⇒ deducible_formula (MMDia i f1) →
    (exists w', (get_rel (nW (nF M)) (nR (nF M)) i) w w' ∧ valuation M w' f1)
  | MMAnd f1 f2 ⇒ deducible_formula (MMAnd f1 f2) →
    (valuation M w f1 ∧ valuation M w f2)
  | MMOr f1 f2 ⇒ deducible_formula (MMOr f1 f2) →
    (valuation M w f1 ∨ valuation M w f2)
  | MMImplies f1 f2 ⇒ deducible_formula (MMImplies f1 f2) →
    (valuation M w f1 → valuation M w f2)
  end.
```

Com base nesta definição de valoração, foram definidas relações de consequência semântica e foi provado que a valoração (ou consequência semântica) em n-modelos é uma generalização da valoração (consequência semântica) em modelos, tal qual foi feito para a prova de conceito.

```
Theorem MMvaluation_generalization: forall modalities W lR RC V f n,
  let NF := Build_nFrame modalities W lR RC in
  let NM := Build_nModel modalities NF V in
  let M := (nModel_to_Model modalities NM n) in
  deducible_formula modalities (formula_to_MMformula f n) →
    forall w, fun_validation M w f ↔
    valuation modalities NM w (formula_to_MMformula f n).
```

Foi definido um conjunto de axiomas e um sistema de Hilbert para a lógica multimodal, de maneira semelhante ao que foi feito na biblioteca base e na prova de conceito, com tipos indutivos representando axiomas e derivações no sistema de Hilbert e uma função que transforma objetos do tipo de axiomas em fórmulas multimodais.

```
Inductive MMaxiom : Set :=
  | MMax1 : MMformula → MMformula → MMaxiom
  | MMax2 : MMformula → MMformula → MMformula → MMaxiom
  (*algumas linhas foram omitidas*)
  | MMax10 : MMformula → MMformula → MMaxiom
  | MMaxK : nat      → MMformula → MMformula → MMaxiom
  (*algumas linhas foram omitidas*)
```

```
  | MMaxGL : nat        → MMformula → MMaxiom.
Definition MMinstance (a: MMaxiom): MMformula :=
  match a with
  | MMax1   f1 f2     ⇒ <! f1 → (f2 → f1) !>
  | MMax2   f1 f2 f3  ⇒ <! (f1 → (f2 → f3)) → ((f1 → f2) → (f1 → f3)) !>
  (*algumas linhas foram omitidas*)
  | MMax10  f1 f2     ⇒ <! ¬¬f1 → f1 !>
  | MMaxK   i f1 f2   ⇒ <! [i](f1 → f2) → ([i] f1 → [i] f2)  !>
  (*algumas linhas foram omitidas*)
  | MMaxGL  i f       ⇒ <! [i]([ i]f  → f) → [i]f !>
  end.
Reserved Notation "A  ;  G  |--M  p" (at level 110, no associativity).
Inductive MMdeduction (A: MMaxiom → Prop): MMtheory → MMformula → Prop :=
  | MMPrem: forall (G:  MMtheory) (f:  MMformula) (i: nat),
                 (deducible_theory G) → (nth_error G i = Some f) → A; G |--M f
  | MMAx: forall (G:  MMtheory) (a:  MMaxiom) (f:  MMformula),
               A a → (deducible_formula f) → MMinstance a = f →
               (deducible_theory G) → A; G |--M f
  | MMMp: forall (G:  MMtheory) (f1 f2:  MMformula),
               (deducible_formula <!f1 → f2!>) → (deducible_theory G) →
               (A;  G |--M (<!f1 → f2!>)) → (A; G |--M f1)  → (A; G |--M f2)
  | MMNec: forall (G:  MMtheory) (f:  MMformula) (i: nat),
                 i < Modalities → (deducible_formula f) → (deducible_theory G) →
                 (A;  G |--M f)  → (A; G |--M <![i]f!>)
where "A  ;  G  |--M  p" := (MMdeduction A G p).
```

A partir destas definições foram descritos sistemas axiomáticos para a lógica multimodal, em específico, foram modeladas versões multimodais de cada sistema axiomático na biblioteca base:

```
Inductive Kn (index: list nat): MMaxiom → Prop :=
  | Kn_ax1: forall f1 f2, (deducible_formula (MMinstance (MMax1 f1 f2))) →
      Kn index (MMax1   f1 f2)
  | Kn_ax2: forall f1 f2 f3, (deducible_formula (MMinstance (MMax2 f1 f2 f3))) →
    Kn index (MMax2   f1 f2 f3)
    (*algumas linhas foram omitidas*)
  | Kn_axK: forall i f1 f2,  In i index →
    (deducible_formula (MMinstance (MMaxK i f1 f2))) → Kn index (MMaxK i f1 f2)
  | Kn_axPos: forall i f1 f2, In i index →
    (deducible_formula (MMinstance (MMaxPos i f1 f2))) → Kn index (MMaxPos i f1 f2).
```

Estes sistemas são parametrizados por uma lista de números naturais, que representam as modalidades daquele sistema. Então, por exemplo, caso um sistema seja parametrizado pela

lista [1,2,3], então as modalidades deste sistema serão $\Box_1/\Diamond_1, \Box_2/\Diamond_2$ e $\Box_3/\Diamond_3$. Para definir um sistema que contém todas as modalidades até a variável `Modalities`, basta passar o parâmetro `seq 0 Modalities`, a função `seq`, que é definida na biblioteca padrão, computa a lista de todos os naturais menores que o segundo argumento, iniciando no primeiro. Ou seja, `seq 0 Modalities` irá retornar a lista [0, ..., `Modalities`-1].

Foram modelados sistemas axiomáticos que representam a fusão (tal qual apresentada na Definição 30) de outros dois sistemas axiomáticos, sejam eles dois sistemas monomodais (sistema `join`), um sistema monomodal e outro sistema multimodal (sistema `join_one`) ou dois sistemas multimodais (sistema `join_two`).

```
Inductive join (S1 S2: axiom → Prop) (index1 index2: nat):
  list nat → MMaxiom → Prop :=
    | derivable_S1: forall a b, S1 a → axiom_to_MMaxiom index1 a b →
      deducible_formula (MMinstance b) →
        join S1 S2 index1 index2 (index1 :: index2 :: nil) b
    | derivable_S2: forall a b, S2 a → axiom_to_MMaxiom index2 a b →
      deducible_formula (MMinstance b) →
        join S1 S2 index1 index2 (index1 :: index2 :: nil) b.
Inductive join_one (S1: axiom → Prop) (S2: list nat → MMaxiom → Prop)
  (index1: nat) (index2: list nat): list nat → MMaxiom → Prop :=
    | derivable_S1_one: forall a b, S1 a → axiom_to_MMaxiom index1 a b →
      deducible_formula (MMinstance b) →
        join_one S1 S2 index1 index2 (index1 :: index2) b
    | derivable_S2_one: forall a, S2 index2 a →
      join_one S1 S2 index1 index2 (index1 :: index2) a.
Inductive join_two (S1 S2: list nat → MMaxiom → Prop) (index1 index2: list nat):
  list nat → MMaxiom → Prop :=
    | derivable_S1_two: forall a, S1 index1 a →
      join_two S1 S2 index1 index2 (index1 ++ index2) a
    | derivable_S2_two: forall a, S2 index2 a →
      join_two S1 S2 index1 index2 (index1 ++ index2) a.
```

Estes sistemas utilizam a mesma noção de parametrização por listas de índices de modalidades que os sistemas multimodais apresentados anteriormente, sendo que as listas de índices destes sistemas resultantes de fusão é gerada a partir das listas de índices dos sistemas unidos. No caso da junção envolvendo sistemas monomodais, um número natural é atribuído a cada sistema, para servir de índice das suas modalidades quando este for elevado para o sistema multimodal.

Para definir a fusão de sistemas axiomáticos foi necessário antes definir uma forma de transformar axiomas do sistema monomodal para o sistema multimodal, o que foi feito por meio de um predicado definido indutivamente.

```
Inductive axiom_to_MMaxiom (index: nat): axiom → MMaxiom → Prop :=
  | transform_ax1: forall f1 f2, axiom_to_MMaxiom index (ax1 f1 f2)
```

```
    (MMax1 (formula_to_MMformula f1 index) (formula_to_MMformula f2 index))
| transform_ax2: forall f1 f2 f3, axiom_to_MMaxiom index (ax2 f1 f2 f3)
    (MMax2 (formula_to_MMformula f1 index) (formula_to_MMformula f2 index)
      (formula_to_MMformula f3 index))
(*algumas linhas foram omitidas*)
| transform_axK: forall f1 f2, axiom_to_MMaxiom index (axK f1 f2)
    (MMaxK index (formula_to_MMformula f1 index) (formula_to_MMformula f2 index))
(*algumas linhas foram omitidas*)
| transform_axGL: forall f, axiom_to_MMaxiom index (axGL f)
    (MMaxGL  index (formula_to_MMformula f index)).
```

Por fim, foi provado que toda fórmula derivável no sistema **K** também é derivável no sistema $\mathbf{K}_n$ e que a fusão de sistemas dedutivos preserva derivações, ou seja, toda fórmula derivável num sistema S_1 também será derivável no sistema $S_1 \oplus S_2$ para qualquer S_2, sejam estes sistemas mono ou multimodais.

```
Theorem MMdeduction_generalization_Kn: forall modalities indexes n G f,
  let K_to_Kn := formula_to_MMformula in
  n < modalities → In n indexes → deduction K G f →
  MMdeduction modalities (Kn modalities indexes)
    (theory_to_MMtheory G n) (K_to_Kn f n).
Theorem join_preserves_deduction: forall modalities S1 n1 G f,
  let K_to_Kn := formula_to_MMformula in
  n1 < modalities → deduction S1 G f →
  forall S2 n2 b, (forall a, axiom_to_MMaxiom n1 a b) →
    join modalities S1 S2 n1 n2 (n1 ::  n2 ::  nil) b →
    MMdeduction modalities (join modalities S1 S2 n1 n2 (n1 :: n2 :: nil))
      (theory_to_MMtheory G n1) (K_to_Kn f n1).
Theorem join_two_preserves_deduction: forall modalities S1 S2 l1 l2 G f a,
  MMdeduction modalities (S1 l1) G f → join_two S1 S2 l1 l2 (l1 ++ l2) a →
  MMdeduction modalities (join_two S1 S2 l1 l2 (l1 ++ l2)) G f.
```

Isso conclui a modelagem de lógicas multimodais no Coq com base na biblioteca de lógica modal apresentada no início deste capítulo. Devido às complexidades impostas pela modelagem no Coq e restrições de tempo, não foi possível modelar provas de transferência de propriedades descritas na Seção 3.1, porém, os resultados obtidos pela prova de conceito e de tentativas de começar a modelagem desta prova indicam que é possível provar, pelo menos, a transferência de corretude.

5.4 DIFICULDADES NA IMPLEMENTAÇÃO

Um dos pontos que causou grande dificuldade na implementação foi a definição das linguagens, tanto para o sistema $\mathbf{KT} \odot \mathbf{K4}$ quanto para o sistema multimodal. Em específico,

o problema era como seria possível implementar no Coq as linguagens de tal forma que elas representassem corretamente suas respectivas apresentações, e também que fosse possível definir traduções entre as linguagens da biblioteca base e do sistema multimodal. A solução selecionada resolve esses problemas, porém, uma limitação desta solução é que ele não representa a união de assinaturas de duas linguagens distintas[3]. A necessidade de realizar traduções também não é ideal, pois limita a expressividade dos sistemas desenvolvidos.

Outro ponto de grande dificuldade foi na definição e tradução de frames para o sistema multimodal. Inicialmente, pretendia-se definir um n-frame a partir da união de frames da biblioteca base, tal e qual é feita na definição de fusão de lógicas, porém isso não é possível devido ao fato que o conjunto de mundos dos frames da biblioteca base são objetos do tipo `Set`. Isto gera dois problemas: igualdade entre objetos em `Set` não é a mesma que igualdade entre conjuntos e o Coq não consegue verificar se os conjuntos de mundos de dois frames distintos são iguais sem saber como estes frames são construídos.

Este segundo problema se mostrou o maior, pois torna muito difícil (ou até impossível) obter a lista de relações de acessibilidade de frames a partir de uma lista de frames, algo necessário para representar a união de vários frames para a obtenção de um n-frame. Considerando uma função que recebe uma lista de frames como argumento e que deve retornar a lista das relações de acessibilidade de todos os frames na lista de frames, qual será o tipo do seu retorno? Certamente seria da forma `list(X -> X -> Prop)` onde `X: Set` e `X` é o conjunto de mundo dos frames da lista, porém o Coq não consegue verificar se `X` é conjunto de mundos de todos os frames da lista, pois não sabe como eles são construídos, logo ele não consegue verificar se o tipo da função é válido e, portanto, não permite a definição desta função. Este mesmo motivo torna muito difícil a conversão de n-modelos multimodais para modelos monomodais, mas, ao invés do problema estar na lista de relações de frames, está na função de valoração dos n-modelos.

Para resolver este problema, uma possível solução seria alterar a definição de frames da biblioteca base para que todo frame seja parametrizado por algum tipo `T` e então o conjunto de mundos do frame seria do tipo `T -> Prop`, ou seja, seria a função característica de um conjunto de objetos que habitam o tipo `T`[4]. Essa não é a única maneira de modelar conjuntos, no Coq, porém talvez seja a mais adequada para esta situação.

Outro problema está relacionado com a lista de relações de um n-frame. Como é necessário acessar elementos arbitrários desta lista, foi utilizada a função `nth` da biblioteca padrão do Coq, que retorna o n-ésimo elemento de uma lista, porém esta função requer um objeto do mesmo tipo que os objetos na lista para ser retornado caso tente-se acessar um elemento maior que o tamanho da lista. Para tanto, foi definida uma relação (vazia) "especial" chamada `dummyRel`, apresentada anteriormente, porém, o problema com essa forma de lidar com isso é que esta é uma relação válida, logo ela poderia estar contida na lista de relações de algum

[3] A impossibilidade de modelar a união de assinaturas de linguagens com tipos indutivos no Coq não é uma hipótese a ser descartada, talvez outra abordagem seja necessária para tratar este problema.

[4] O autor vem por meio desta nota expressar seu grande desgosto de trabalhar com teoria de conjuntos no Coq.

n-frame, portanto, caso essa relação fosse retornada ao aplicar a função numa lista de relações que contém uma relação vazia, seria impossível determinar se a função teve esse retorno pois não encontrou o elemento na lista de relações ou porque essa era a relação selecionada.

Por fim, provar a transferência de corretude, tanto na prova de conceito quanto na modelagem final dos sistemas multimodais, se mostrou um problema maior do que o esperado. Apesar de ter sido possível provar a corretude do sistema $\mathbf{KT} \odot \mathbf{K4}$ na prova de conceito, por ser uma prova direta que não necessitou das premissas de corretude dos sistemas base, tal prova não foi uma prova de transferência. Este resultado inicial indica que, caso seja possível provar esse fato no Coq, talvez essa prova não seja pelo método de transferência, ou pelo menos que essa prova não seria tão simples quanto ela é no "papel e caneta". Mais ainda, para a implementação dos sistemas multimodais foi possível formular um predicado de corretude porém, devido a restrições de tempo, a complexidade dessa prova e as dificuldades envolvendo frames descritas anteriormente, a prova de corretude de fato não pode ser concluída.

6 CONCLUSÕES

Combinações de lógicas é um tópico relativamente novo e complexo dentro da lógica, tanto do ponto de vista filosófico quanto matemático, porém muito já foi estudado e desenvolvido sobre, algo que foi evidenciado no Capítulo 3 e no Apêndice A deste trabalho, tal apresentação entende-se por satisfazer o primeiro objetivo deste trabalho, especificamente "Estudar os principais conceitos de combinações de lógicas, em especial, a fusão". Assistentes de provas são ferramentas também novas, que permitem expressar grande parte da matemática em suas linguagens e que possibilitam seus usuários provarem propriedades sobre objetos expressos dentro de si. Ademais, essas provas independem da verificação de um humano para garantir a confiabilidade, logo, assistentes de provas são excelentes ferramentas para provar propriedades de sistemas complexos, como sistemas de software.

Portanto, temos que assistentes de provas podem ser usados para modelar e verificar a corretude de sistemas lógicos resultantes da combinação de outros sistemas lógicos mais simples, algo que foi evidenciado com os desenvolvimentos deste trabalho. Foi possível modelar e verificar a corretude do sistema $\mathbf{KT} \odot \mathbf{K4}$, um sistema resultante da fusão de dois sistemas lógicos mais simples. Entende-se que esta implementação satisfaz o segundo objetivo deste trabalho, especificamente "Realizar um estudo de caso de fusões de lógicas modais no Coq". Esta implementação não foi trivial e a prova de corretude não se deu por meio do método de transferência, porém ela indica que, pelo menos para casos restritos, é possível provar em Coq a corretude de sistemas lógicos resultantes da fusão.

A principal contribuição deste trabalho foi a modelagem de sistemas sintáticos de lógicas resultantes da fusão de forma paramétrica, sendo também demonstrado que a fusão de sistemas sintáticos preserva derivações. Com isso, é possível representar, sintaticamente, sistemas lógicos resultantes da fusão de dois outros sistemas lógicos arbitrários. Entende-se que este resultado satisfaz o terceiro objetivo deste trabalho, especificamente, "Modelar, de forma paramétrica, sistemas de lógicas multimodais resultantes de fusão de lógicas modais em Coq".

Apesar disto, a modelagem é complexa e não foi possível representar explicitamente a combinação dos sistemas semânticos de lógicas monomodais, apenas foi possível representar a combinação de seus sistemas sintáticos. Além do trabalho de representar os componentes do sistema lógico, é também interessante provar que estes estão corretos com relação aos sistemas básicos que foram combinados, algo que, devido à impossibilidade de modelar os sistemas semânticos, não foi possível de ser feito.

Propõe-se como trabalhos futuros alterações tanto na biblioteca modal base quanto na extensão desenvolvida nesse trabalho, modificando a definição de frame da biblioteca base para que este seja parametrizado por algum tipo (como descrito na Seção 5.4), refletir essa alteração nos n-frames da extensão para poder descrever a fusão de frames, lidar com a seleção de uma relação da lista de relações de uma n-frame de forma que não seja necessário utilizar a função `nth` ou variantes, modelar a união de assinaturas de linguagens e, por fim, provar a transferência

de corretude e completude pela fusão.

REFERÊNCIAS

APPEL, Kenneth; HAKEN, Wolfgang. Every planar map is four colorable. **Bulletin of the American mathematical Society**, American Mathematical Society, v. 82, n. 5, p. 711–712, 1976. Citado na página 46.

APPEL, Kenneth; HAKEN, Wolfgang; KOCH, John. Every planar map is four colorable. part ii: Reducibility. **Illinois Journal of Mathematics**, Duke University Press, v. 21, n. 3, p. 491–567, 1977. Citado na página 46.

APPEL, Kenneth I; HAKEN, Wolfgang. **Every planar map is four colorable**. Washington: American Mathematical Society, 1989. v. 98. Citado na página 46.

AVIGAD, Jeremy; MOURA, Leonardo de; KONG, Soonho. Theorem proving in lean. **Online at <https://leanprover.github.io/theorem_proving_in_lean/theorem_proving_in_lean.pdf>**, 2017. Citado 2 vezes nas páginas 46 e 47.

AZURAT, Ade; PRASETYA, Wishnu. **A Survey on Embedding Programming Logics in a Theorem Prover**. Utrecht, 2002. Citado 2 vezes nas páginas 13 e 59.

BARENDREGT, Henk. Introduction to generalized type systems. **Journal of Functional Programming**, Cambridge University Press, v. 1, n. 2, p. 125–154, 1991. Citado 2 vezes nas páginas 49 e 50.

BARENDREGT, Henk; GEUVERS, Herman. Proof-assistants using dependent type systems. In: **Handbook of Automated Reasoning**. NLD: Elsevier Science Publishers B. V., 2001. v. 2, p. 1149–1238. ISBN 0444508120. Citado na página 47.

BENZMÜLLER, Christoph. Combining logics in simple type theory. In: SPRINGER. **International Workshop on Computational Logic in Multi-Agent Systems**. Berlin, 2010. p. 33–48. Citado na página 13.

BLACKBURN, Patrick; RIJKE, Maarten De; VENEMA, Yde. **Modal logic**. Cambridge: Cambridge University Press, 2001. v. 53. Citado 10 vezes nas páginas 15, 17, 23, 24, 25, 27, 29, 30, 31 e 45.

BRUIJN, Nicolaas Govert De. A survey of the project automath. In: **Studies in Logic and the Foundations of Mathematics**. Amsterdam: Elsevier, 1980. v. 133, p. 141–161. Citado na página 12.

CARNAP, Rudolf. **Introduction to Semantics**. Cambridge: Harvard University Press, 1942. Citado 2 vezes nas páginas 11 e 15.

CARNIELLI, Walter et al. **Analysis and Synthesis of Logics: how to cut and paste reasoning systems**. Netherlands: Springer Science & Business Media, 2008. v. 35. Citado 2 vezes nas páginas 16 e 33.

CARNIELLI, Walter; CONIGLIO, Marcelo Esteban. Combining Logics. In: ZALTA, Edward N. (Ed.). **The Stanford Encyclopedia of Philosophy**. Fall 2020. Stanford: Metaphysics Research Lab, Stanford University, 2020. Citado 2 vezes nas páginas 11 e 33.

CHELLAS, Brian F. **Modal Logic: An Introduction**. Cambridge: Cambridge University Press, 1980. Citado 8 vezes nas páginas 16, 17, 18, 19, 21, 23, 24 e 25.

CHURCH, Alonzo. A formulation of the simple theory of types. **The journal of symbolic logic**, Cambridge University Press, v. 5, n. 2, p. 56–68, 1940. Citado 2 vezes nas páginas 11 e 48.

COQUAND, Thierry. Type Theory. In: ZALTA, Edward N.; NODELMAN, Uri (Ed.). **The Stanford Encyclopedia of Philosophy**. Fall 2022. Stanford: Metaphysics Research Lab, Stanford University, 2022. Citado na página 47.

COQUAND, Thierry; HUET, Gerard. Constructions: A higher order proof system for mechanizing mathematics. In: SPRINGER. **European Conference on Computer Algebra**. Rocquencourt, 1985. p. 151–184. Citado na página 49.

COQUAND, Thierry; PAULIN, Christine. Inductively defined types. In: SPRINGER. **International Conference on Computer Logic**. Tallinn, 1988. p. 50–66. Citado na página 50.

CROSILLA, Laura. Predicativity and feferman. **Feferman on foundations: Logic, mathematics, philosophy**, Springer, p. 423–447, 2017. Citado na página 51.

CURRY, Haskell Brooks; FEYS, Robert. **Combinatory logic**. Amsterdam: North-Holland Amsterdam, 1958. v. 1. Citado 2 vezes nas páginas 12 e 48.

DYBJER, Peter; PALMGREN, Erik. Intuitionistic Type Theory. In: ZALTA, Edward N.; NODELMAN, Uri (Ed.). **The Stanford Encyclopedia of Philosophy**. Spring 2023. Stanford: Metaphysics Research Lab, Stanford University, 2023. Citado na página 49.

EMERSON, E. Allen. Temporal and modal logic. In: VAN LEEUWEN, JAN (Ed.). **Formal Models and Semantics**. Amsterdam: Elsevier, 1990, (Handbook of Theoretical Computer Science). p. 995–1072. ISBN 978-0-444-88074-1. Disponível em: <https://www.sciencedirect.com/science/article/pii/B9780444880741500214>. Citado 2 vezes nas páginas 27 e 28.

FAGIN, Ronald et al. **Reasoning about knowledge**. Cambridge: MIT press, 1995. Citado 3 vezes nas páginas 11, 28 e 29.

FINE, Kit; SCHURZ, Gerhard. Transfer theorems for multimodal logics. **Logic and reality: essays on the legacy of Arthur Prior**, Oxford University Press Oxford, p. 169–213, 1996. Citado 5 vezes nas páginas 29, 34, 35, 40 e 44.

FITTING, Melvin. Logics with several modal operators. **Theoria**, Blackwell Publishing Ltd Oxford, UK, v. 35, n. 3, p. 259–266, 1969. Citado 2 vezes nas páginas 11 e 33.

FUENMAYOR, David; BENZMÜLLER, Christoph. Mechanised assessment of complex natural-language arguments using expressive logic combinations. In: SPRINGER. **International Symposium on Frontiers of Combining Systems**. London, 2019. p. 112–128. Citado na página 13.

GABBAY, D.M. **Many-dimensional Modal Logics: Theory and Applications**. Amsterdam: North Holland Publishing Company, 2003. (Studies in logic and the foundations of mathematics). ISSN 0049-237X. ISBN 9780444508263. Citado 14 vezes nas páginas 11, 15, 16, 17, 19, 20, 21, 23, 25, 27, 28, 29, 33 e 34.

GEUVERS, H. Proof assistants: History, ideas and future. **Sadhana**, Springer Science and Business Media LLC, v. 34, n. 1, p. 3–25, feb 2009. Disponível em: <https://doi.org/10.1007/s12046-009-0001-5>. Citado 2 vezes nas páginas 12 e 46.

GILBERT, Gaëtan et al. Definitional proof-irrelevance without k. **Proceedings of the ACM on Programming Languages**, Association for Computing Machinery, New York, NY, USA, v. 3, n. POPL, jan 2019. Disponível em: <https://doi.org/10.1145/3290316>. Citado na página 51.

GIRARD, Jean-Yves. Une extension de l'interpretation de gödel a l'analyse, et son application a l'elimination des coupures dans l'analyse et la theorie des types. In: **Studies in Logic and the Foundations of Mathematics**. Oslo: Elsevier, 1971. v. 63, p. 63–92. Citado na página 48.

GIRARD, Jean-Yves. **Interprétation fonctionnelle et élimination des coupures de l'arithmétique d'ordre supérieur**. Tese (Doutorado) — Éditeur inconnu, 1972. Citado 2 vezes nas páginas 48 e 49.

GIRARD, Jean-Yves; TAYLOR, Paul; LAFONT, Yves. **Proofs and types**. Cambridge: Cambridge University Press, 1989. v. 7. Citado na página 48.

GÖDEL, Kurt. An interpretation of the intuitionistic propositional calculus. **Collected Works**, Oxford University Press New York, v. 1, p. 301–303, 1986. Citado 2 vezes nas páginas 11 e 15.

GOLDBLATT, Robert. **Mathematics of Modality**. Stanford: Center for the Study of Language and Information Publications, 1993. Citado na página 15.

GONTHIER, Georges. **A computer-checked proof of the four colour theorem**. Cambridge, 2005. Citado 2 vezes nas páginas 12 e 46.

HARPER, Robert; HONSELL, Furio; PLOTKIN, Gordon. A framework for defining logics. **Journal of the ACM (JACM)**, ACM New York, NY, USA, v. 40, n. 1, p. 143–184, 1993. Citado na página 50.

HARRISON, John; URBAN, Josef; WIEDIJK, Freek. History of interactive theorem proving. In: **Computational Logic**. Amsterdam: [s.n.], 2014. v. 9, p. 135–214. Citado 2 vezes nas páginas 12 e 46.

HEIJENOORT, Jean Van. **From Frege to Gödel: a source book in mathematical logic, 1879-1931**. Cambridge: Harvard University Press, 2002. Citado na página 47.

HINTIKKA, Kaarlo Jaakko Juhani. **Knowledge and Belief: An Introduction to the Logic of the Two Notions**. Ithaca: Cornell University Press, 1962. Citado na página 11.

HOWARD, William Alvin. The formulae-as-types notion of construction. In: CURRY, Haskell et al. (Ed.). **To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism**. Chicago: Academic Press, 1980. Citado 2 vezes nas páginas 12 e 48.

HUET, Gérard. **Logical foundations of functional programming**. Austin: Addison-Wesley Longman Publishing Co., Inc., 1990. Citado na página 48.

KLEIN, Gerwin et al. Sel4: Formal verification of an operating-system kernel. **Commun. ACM**, Association for Computing Machinery, New York, NY, USA, v. 53, n. 6, p. 107–115, jun 2010. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/1743546.1743574>. Citado na página 12.

KRACHT, Marcus; WOLTER, Frank. Properties of independently axiomatizable bimodal logics. **The Journal of Symbolic Logic**, Cambridge University Press, v. 56, n. 4, p. 1469–1485, 1991. Citado na página 34.

KRIPKE, Saul A. A completeness theorem in modal logic. **The journal of symbolic logic**, Cambridge University Press, v. 24, n. 1, p. 1–14, 1959. Citado 2 vezes nas páginas 11 e 15.

KRIPKE, Saul A. Semantical analysis of modal logic i normal modal propositional calculi. **Mathematical Logic Quarterly**, WILEY-VCH Verlag Berlin GmbH Berlin, v. 9, n. 5-6, p. 67–96, 1963. Citado 2 vezes nas páginas 11 e 15.

LAMPORT, Leslie. Specifying concurrent program modules. **ACM Transactions on Programming Languages and Systems (TOPLAS)**, ACM New York, NY, USA, v. 5, n. 2, p. 190–222, 1983. Citado na página 28.

LAMPORT, Leslie. The temporal logic of actions. **ACM Transactions on Programming Languages and Systems (TOPLAS)**, ACM New York, NY, USA, v. 16, n. 3, p. 872–923, 1994. Citado na página 28.

LAMPORT, Leslie. **Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers**. USA: Addison-Wesley Longman Publishing Co., Inc., 2002. ISBN 032114306X. Citado 2 vezes nas páginas 11 e 28.

LEROY, Xavier. **The CompCert C verified compiler: Documentation and user's manual**. Tese (Doutorado) — Inria, 2021. Citado 2 vezes nas páginas 12 e 51.

LESCANNE, Pierre. Mechanizing epistemic logic with coq. Citeseer, 2004. Citado na página 28.

LESCANNE, Pierre; PUISSÉGUR, Jérôme. Dynamic logic of common knowledge in a proof assistant. **arXiv preprint arXiv:0712.3146**, 2007. Citado na página 13.

LEWIS, Clarence Irving. **A survey of symbolic logic**. Berkeley: University of California press, 1918. Citado na página 15.

LEWIS, Clarence Irving; LANGFORD, Cooper Harold; LAMPRECHT, P. **Symbolic logic**. New York: Dover Publications New York, 1959. v. 170. Citado na página 15.

MARTIN-LÖF, Per. An intuitionistic theory of types: Predicative part. In: **Studies in Logic and the Foundations of Mathematics**. Stockholm: Elsevier, 1975. v. 80, p. 73–118. Citado 3 vezes nas páginas 12, 48 e 49.

MARTIN-LÖF, Per; SAMBIN, Giovanni. **Intuitionistic type theory**. Naples: Bibliopolis Naples, 1984. v. 9. Notas de Aula. Citado 3 vezes nas páginas 12, 48 e 49.

MOURA, Leonardo de et al. The lean theorem prover (system description). In: FELTY, Amy P.; MIDDELDORP, Aart (Ed.). **Automated Deduction - CADE-25**. Cham: Springer International Publishing, 2015. p. 378–388. ISBN 978-3-319-21401-6. Citado na página 47.

ORLOV, Ivan E. The calculus of compatibility of propositions. **Mathematics of the USSR, Sbornik**, v. 35, p. 263–286, 1928. Citado na página 15.

PAIVA, Nuno; CALEIRO, Carlos. **Temporal logic in Coq**. Dissertação (Projeto de Diplomação) — Licenciatura em Matemática Aplicada e Computação – Instituto Superior Técnico, 1998. Citado na página 51.

PAULIN-MOHRING, Christine. Introduction to the coq proof-assistant for practical software verification. In: ____. **Tools for Practical Software Verification: LASER, International Summer School 2011, Elba Island, Italy, Revised Tutorial Lectures**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 45–95. ISBN 978-3-642-35746-6. Citado na página 12.

PAULIN-MOHRING, Christine. Introduction to the Calculus of Inductive Constructions. In: PALEO, Bruno Woltzenlogel; DELAHAYE, David (Ed.). **All about Proofs, Proofs for All**. London: College Publications, 2015, (Studies in Logic (Mathematical logic and foundations), v. 55). Citado na página 49.

PIERCE, Benjamin C. **Types and programming languages**. Cambridge: MIT press, 2002. Citado na página 48.

PIERCE, Benjamin C. et al. **Logical Foundations**. Pennsylvania: Electronic textbook, 2021. v. 1. (Software Foundations, v. 1). Version 6.1, <http://softwarefoundations.cis.upenn.edu>. Citado 6 vezes nas páginas 52, 53, 54, 55, 57 e 58.

PNUELI, Amir. The temporal logic of programs. In: IEEE. **18th Annual Symposium on Foundations of Computer Science (sfcs 1977)**. USA, 1977. p. 46–57. Citado 2 vezes nas páginas 11 e 28.

PRIEST, Graham. **An Introduction to Non-Classical Logic: From If to Is**. 2. ed. Cambridge: Cambridge University Press, 2008. (Cambridge Introductions to Philosophy). Citado na página 19.

PRIOR, Arthur N. **Time and modality: Being the John Locke Lectures for 1955-6 Delivered in the University of Oxford**. Oxford: Claredon Press, 1957. Citado na página 11.

PRIOR, Arthur N. **Past, Present and Future**. Oxford: Clarendon Press, 1967. Citado na página 11.

RABE, Florian. How to identify, translate and combine logics? **Journal of Logic and Computation**, Oxford University Press, v. 27, n. 6, p. 1753–1798, 2017. Citado na página 13.

RAMSEY, Frank Plumpton. General propositions and causality. Kegan Paul, Trench & Trubner, 1931. Citado na página 47.

REYNOLDS, John C. Towards a theory of type structure. In: SPRINGER. **Programming Symposium: Proceedings, Colloque sur la Programmation Paris, April 9–11, 1974**. Paris, 1974. p. 408–425. Citado na página 48.

ROBERTSON, Neil et al. The four-colour theorem. **journal of combinatorial theory, Series B**, Elsevier, v. 70, n. 1, p. 2–44, 1997. Citado na página 46.

ROGGIA, Karina Girardi. **Fusion of General Modal Logics Labelled with Truth Values**. Tese (Doutorado) — INSTITUTO SUPERIOR TÉCNICO, 2012. Citado 4 vezes nas páginas 11, 17, 27 e 33.

RUSSELL, Bertrand. **Principles of Mathematics**. Cambridge: Cambridge University Press, 1903. Citado 2 vezes nas páginas 11 e 47.

RUSSELL, Bertrand. Mathematical logic as based on the theory of types. **American Journal of Mathematics**, JSTOR, v. 30, n. 3, p. 222–262, 1908. Citado na página 47.

SILVA, Rafael Castro Goncalves. **Uma certificação em COQ do algoritmo W monádico**. Dissertação (Dissertação) — Universidade do Estado de Santa Catarina, Programa de Pós Graduação em Computação Aplicada, 2019. Citado 3 vezes nas páginas 46, 47 e 51.

SILVEIRA, Ariel Agne Da et al. A sound deep embedding of arbitrary normal modal logics in coq. In: **XXVI Brazilian Symposium on Programming Languages**. New York, NY, USA: Association for Computing Machinery, 2022. (SBLP 2022), p. 1–7. ISBN 9781450397445. Disponível em: <https://doi.org/10.1145/3561320.3561329>. Citado 3 vezes nas páginas 12, 51 e 59.

SILVEIRA, Ariel Agne da. **Implementação de uma biblioteca de lógica modal em Coq**. Dissertação (Projeto de Diplomação) — Bacharelado em Ciência da Computação–Centro de Ciências Tecnológicas, UDESC, Joinville, 2020. Citado 7 vezes nas páginas 12, 20, 22, 23, 24, 25 e 59.

TAIT, William W. Intensional interpretations of functionals of finite type i. **The journal of symbolic logic**, Cambridge University Press, v. 32, n. 2, p. 198–212, 1967. Citado na página 48.

TEAM, The Coq Development. **The Coq Reference Manual**. France, 2022. Citado 7 vezes nas páginas 12, 47, 49, 50, 52, 57 e 58.

THOMASON, Richmond H. Combinations of tense and modality. In: **Handbook of philosophical logic**. Dordrecht: Springer, 1984. p. 135–165. Citado 2 vezes nas páginas 11 e 33.

WADLER, Philip. Propositions as types. **Communications of the ACM**, Association for Computing Machinery, New York, NY, USA, v. 58, n. 12, p. 75–84, nov 2015. ISSN 0001-0782. Disponível em: <https://doi.org/10.1145/2699407>. Citado na página 48.

WIND, Paulien de. **Modal logic in coq**. Dissertação (Dissertação) — Vrije Universiteit, 2001. Citado 3 vezes nas páginas 17, 19 e 51.

WRIGHT, Georg Henrik von. **An Essay in Modal Logic**. Amsterdam: North-Holland Amsterdam, 1951. Citado na página 11.

ZALTA, Edward N. **Basic Concepts In Modal Logic**. Center for the Study of Language and Information Stanford University, 1995. Disponível em: <http://mally.stanford.edu/notes.pdf>. Citado 5 vezes nas páginas 11, 15, 17, 19 e 24.

ZORN, Max. A remark on method in transfinite algebra. **Bulletin of the American Mathematical Society**, v. 41, n. 10, p. 667–670, 1935. Citado 2 vezes nas páginas 43 e 89.

APÊNDICE A – PROVA FORMAL DA TRANSFERÊNCIA DE COMPLETUDE PELA FUSÃO DE LÓGICAS MODAIS

Prova do Teorema 4. Sendo $\Gamma \subseteq \Theta$ um conjunto $\mathcal{L}_{12}$-consistente, devemos provar que Γ é verdadeiro em algum mundo de um 12-modelo baseado em um 12-frame para $\mathcal{L}_{12}$. Como Γ é $\mathcal{L}_{12}$-consistente, Γ também é $\mathcal{L}_1$-consistente e $\mathcal{L}_2$-consistente. Ademais, como $\Gamma \subseteq \Theta \subseteq \mathrm{DC}_\pi(\Theta)$, existe um π-modelo, baseado em um π-frame para $\mathcal{L}_\pi$, onde Γ é verdadeiro em algum mundo (devido à premissa de completude de $\mathcal{L}_1$ e $\mathcal{L}_2$). Este π-modelo será chamado de *modelo inicial*, e será apresentado formalmente à frente.

Inicialmente, devemos definir formalmente os conceitos de π-teorias, modelos etiquetados e elementos de modelos.

Definição 46 (π-Teorias, π-Modelos Etiquetados e π-Elementos de Modelos)**.**

46.1 Sendo Δ um conjunto de fórmulas, a π-*teoria de* Δ é definida como, onde $\mathcal{L}_{12}$ é uma lógica bimodal:

$$\mathrm{T}_\pi(\Delta) = \{\Box_\pi^n \delta \mid \delta \in \mathrm{B}(\mathrm{S}_\pi(\Delta)) \cap \mathcal{L}_{12} \text{ e } \mathrm{d}_\pi(\Box_\pi^n \delta) \leq \mathrm{d}_\pi(\Delta)\}$$

46.2 Um π-*modelo etiquetado* é uma tripla $\langle \mathcal{M}, w_\mathcal{M}, \Sigma(\mathcal{M})\rangle$, onde $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}_\pi, \mathcal{V}_\pi\rangle$ e:

(i) $\mathcal{M}$ é um π-modelo para $\mathcal{L}_\pi$;

(ii) $w_\mathcal{M} \in \mathcal{W}$;

(iii) $\Sigma(\mathcal{M})$ é um conjunto de elementos (fórmulas interpretadas como átomos), fechado para a operação de sub-elementos sobre elementos;

(iv) $\mathrm{T}_\pi(\Sigma(\mathcal{M}))$ é verdadeiro em $w_\mathcal{M}$ em $\mathcal{M}$.

46.3 Dado um π-modelo etiquetado $\mathcal{E} = \langle \mathcal{M}, w_\mathcal{M}, \Sigma(\mathcal{M})\rangle$ onde $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}_\pi, \mathcal{V}_\pi\rangle$, para cada mundo $w \in \mathcal{W}$ *o conjunto de elementos* $\Sigma_\mathcal{M}(w)$ *de* w é definido indutivamente como:

(i) $\Sigma_\mathcal{M}(w_\mathcal{M}) = \Sigma(\mathcal{M})$;

(ii) Sendo $w_i, w_j \in \mathcal{W}$, se $w_i \mathcal{R}_\pi w_j$ e $\Box_\pi \varphi \in \Sigma_\mathcal{M}(w_i)$, então $\mathrm{SC}(\varphi) \in \Sigma_\mathcal{M}(w_j)$. ■

O mundo $w_\mathcal{M}$ é chamado de mundo base do modelo $\mathcal{M}$, $\Sigma(\mathcal{M})$ é chamado de conjunto de elementos do modelo $\mathcal{M}$ e $\Sigma_\mathcal{M}(w)$ é o conjunto de elementos de w em $\mathcal{M}$. Sendo $\mathcal{E} = \langle \mathcal{M}, w_\mathcal{M}, \Sigma(\mathcal{M})\rangle$ um modelo etiquetado, escreveremos $w \in \mathcal{E}$ para indicar que $w \in \mathcal{W}$, sendo $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V}\rangle$.

Com isso, podemos definir formalmente os conceitos de modelo e mundo iniciais, a partir de algum conjunto de fórmulas Γ que deve ser satisfeito.

Definição 47 (Modelo Inicial e Mundo Inicial)**.** É chamado de *modelo inicial* o π-modelo etiquetado:

$$\boldsymbol{\mathcal{I}} = \langle \mathcal{I}, \boldsymbol{i}, \mathrm{SC}(\Gamma)\rangle$$

onde o mundo base ***i*** deste modelo é chamado de *mundo inicial*. O modelo $\mathcal{I}$ é o modelo que satisfaz Γ e o mundo ***i*** é um mundo onde Γ é verdadeiro.

Podemos então provar o seguinte resultado auxiliar:

Lema 1. *Sendo $\mathcal{E} = \langle \mathcal{M}, w_{\mathcal{M}}, \Sigma(\mathcal{M}) \rangle$ um modelo etiquetado e $w_i \in \mathcal{E}$, então:*

1.1 Se $\Sigma_{\mathcal{M}}(w_i) \neq \varnothing$ então $\mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_i)) = (\mathrm{d}_\pi(\Sigma(\mathcal{M})) - \mathrm{dist}(w_{\mathcal{M}}, w_i))$;

1.2 Sendo $w_i \neq w_{\mathcal{M}}$, $\Sigma_{\mathcal{M}}(w_i) = \varnothing$ sse $\mathrm{dist}(w_{\mathcal{M}}, w_i) > \mathrm{d}_\pi(\Sigma(\mathcal{M}))$. ■

Prova do Lema 1. Temos dois casos, um para 1.1 outro para 1.2:

Caso 1.1 Prova-se por indução em $\mathrm{dist}(w_{\mathcal{M}}, w_j)$:

Base: Pela definição de dist, temos que $w_{\mathcal{M}} = w_j$, logo, $\mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_{\mathcal{M}})) = \mathrm{d}_\pi(\Sigma(\mathcal{M}))$, pela definição de $\Sigma_{\mathcal{M}}$, temos $\mathrm{d}_\pi(\Sigma(\mathcal{M})) = \mathrm{d}_\pi(\Sigma(\mathcal{M}))$.

Hipotese: Assumindo $\forall w_k, \mathrm{dist}(w_{\mathcal{M}}, w_k) = k$, temos que $\mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_k)) = \mathrm{d}_\pi(\Sigma(\mathcal{M})) - \mathrm{dist}(w_{\mathcal{M}}, w_k)$.

Passo: Pelas definições de $\Sigma_{\mathcal{M}}$ e d_π, temos que $\forall w_a, w_b$, onde $w_a \mathcal{R} w_b, \mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_a)) = \mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_b)) + 1$. Assumindo que $\mathrm{dist}(w_{\mathcal{M}}, w_j) = k+1$, sabemos que existe algum w_x onde $w_x \mathcal{R} w_j$ e $\mathrm{dist}(w_{\mathcal{M}}, w_x) = k$. Logo, pela hipótese podemos concluir que:

$$\mathrm{d}_\pi(\Sigma(w_x)) = \mathrm{d}_\pi(\Sigma(\mathcal{M})) - \mathrm{dist}(w_{\mathcal{M}}, w_x)$$

Mais ainda, temos:

- Pela definição de $\Sigma_{\mathcal{M}}$: $\Sigma(\mathcal{M}) = \Sigma_{\mathcal{M}}(w_{\mathcal{M}})$
- Pelas conclusões anteriores: $\mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_x)) = \mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_j)) + 1$

Portanto temos:

$$\mathrm{d}_\pi(\Sigma(w_j)) + 1 = \mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_{\mathcal{M}})) - \mathrm{dist}(w_{\mathcal{M}}, w_x)$$

Ou seja:

$$\mathrm{d}_\pi(\Sigma(w_j)) = \mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_{\mathcal{M}})) - (\mathrm{dist}(w_{\mathcal{M}}, w_x) + 1)$$

Logo:

$$\mathrm{d}_\pi(\Sigma(w_j)) = \mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_{\mathcal{M}})) - \mathrm{dist}(w_{\mathcal{M}}, w_j)$$

Caso 1.2 Consequência imediata do caso anterior e da definição de $\Sigma_{\mathcal{M}}$. ■

Este lema demonstra que, dado $\mathrm{d}_\pi(\Sigma(\mathcal{M}))$ finito, $\mathrm{d}_\pi(\Sigma_{\mathcal{M}}(w_j))$ irá diminuir com o aumento de $\mathrm{dist}(w_{\mathcal{M}}, w_j)$ até $\Sigma_{\mathcal{M}}(w_j)$ se tornar vazio, ou seja, o conjunto de elementos de um dado mundo em um modelo depende da distância do mundo até o mundo base do modelo.

Com isso, podemos formalmente enunciar o conceito de ancoramento de modelos:

Definição 48 (Ancoramento de Modelos)**.** Sendo $\mathcal{E}_0 = \langle \mathcal{M}, w_{\mathcal{M}}, \Sigma(\mathcal{M}) \rangle$ e $\mathcal{E}_1 = \langle \mathcal{N}, w_{\mathcal{N}}, \Sigma(\mathcal{N}) \rangle$ modelos etiquetados, onde $\mathcal{E}_0$ tem tipo π, é dito que $\mathcal{E}_1$ *está ancorado em* $\mathcal{E}_0$ se, e somente se:

48.1 $\mathcal{E}_1$ tem tipo $\overline{\pi}$;

48.2 Sendo $\mathcal{M} = \langle \mathcal{W}_m, \mathcal{R}_m, \mathcal{V}_m \rangle$ e $\mathcal{N} = \langle \mathcal{W}_n, \mathcal{R}_n, \mathcal{V}_n \rangle$ então, $\mathcal{W}_m \cap \mathcal{W}_n = \{w_{\mathcal{N}}\}$ e $\mathcal{E}_0 = \boldsymbol{\mathcal{I}}$ ou $w_{\mathcal{N}}$ não é o mundo base de $\mathcal{E}_0$;

48.3 $\Sigma(\mathcal{N}) = \mathrm{S}_\pi(\Sigma_{\mathcal{M}}(w_{\mathcal{N}}))$;

48.4 Para todo $\varphi \in \Sigma(\mathcal{N}) : \mathcal{M}, w_{\mathcal{N}} \Vdash \varphi$ sse $\mathcal{N}, w_{\mathcal{N}} \Vdash \varphi$ ■

É fácil de observar que a definição de ancoramento é irreflexiva e antissimétrica, mais ainda, se $\mathcal{E}_1$ está ancorado em $\mathcal{E}_0$ e w é o mundo base de $\mathcal{E}_1$, é dito que $\mathcal{E}_1$ *está ancorando em* $\mathcal{E}_0$ *no mundo* w. Para cada π-modelo $\mathcal{E}_0$ e mundo $w \in \mathcal{E}_0$, o conjunto $\mathrm{S}_\pi(\Sigma_{\mathcal{M}}(w))$ é chamado de conjunto de concordância de w em $\mathcal{E}_0$ pois, para cada $\overline{\pi}$-modelo $\mathcal{E}_1$ ancorado em w, $\mathcal{E}_0$ e $\mathcal{E}_1$ devem concordar com a valoração deste conjunto em w.

Intuitivamente, podemos interpretar a operação de ancoramento de modelos como uma forma de "alternar o contexto" durante a valoração de uma fórmula que contenha múltiplas modalidades, pois permite que fórmulas com modalidades de um tipo sejam valoradas em um modelos de outro tipo.

Definição 49 (Diagramas)**.** Para cada conjunto $\mathcal{L}_\pi$-consistente Δ, π-modelo $\mathcal{M}$ e mundo w em $\mathcal{M}$, o *diagrama de* Δ *em* $\mathcal{M}$ *no mundo* w, denotado por $\mathrm{DG}(\Delta)$ é dado por

$$\mathrm{DG}(\Delta) = \{\delta \mid \delta \in \Delta \text{ e } \mathcal{M}, w \Vdash \delta\} \cup \{\neg\delta \mid \delta \in \Delta \text{ e } \mathcal{M}, w \nVdash \delta\}$$

O *diagrama de concordância* $\mathrm{D}_{\mathcal{M}}(w)$ *do mundo* w *em* $\mathcal{M}$ é definido como $\mathrm{D}_{\mathcal{M}}(w) = \mathrm{DG}(\mathrm{S}_\pi(\Sigma_{\mathcal{M}}(w)))$. ■

É fácil observar que a condição 48.4 é equivalente à impor a restrição que $\mathrm{D}_{\mathcal{M}}(w_{\mathcal{N}})$ seja verdadeiro no mundo $w_{\mathcal{N}}$ no modelo $\mathcal{N}$. Uma importante propriedade de π-modelos etiquetados é a seguinte:

Lema 2. *Para todo* π*-modelo etiquetado* $\mathcal{E}$ *e* $w \in \mathcal{E}$, $\mathrm{D}_{\mathcal{M}}(w)$ *é* $\mathcal{L}_{12}$*-consistente.* ■

Prova do Lema 2. Inicialmente, devemos assumir que $\mathrm{D}_{\mathcal{M}}(w)$ é $\mathcal{L}_{12}$-inconsistente, para algum $w \in \mathcal{E}$. Então, existe um conjunto finito e não vazio $\Delta \subseteq \mathrm{D}_{\mathcal{M}}(w)$ onde $\neg\bigwedge\Delta \in \mathcal{L}_{12}$[1], ou seja, existe algum δ_i tal que $\delta_i \in \Delta$ e $\neg\delta_i \in \mathcal{L}_{12}$.

Assumindo que $\mathrm{dist}(w_{\mathcal{M}}, w) = k$. Sabemos que, pela definição de $\mathrm{D}_{\mathcal{M}}$, todo $\delta \in \Delta$ é da forma φ ou $\neg\varphi$, sendo $\varphi \in \mathrm{S}_\pi(\Sigma_{\mathcal{M}}(w))$. Portanto $\varphi \in \mathrm{S}_\pi(\Sigma(\mathcal{M}))$ e $\Delta \subseteq \mathrm{B}(\mathrm{S}_\pi(\Sigma(\mathcal{M})))$, mais ainda $\mathrm{d}_\pi(\varphi) \leq \mathrm{d}_\pi(\Sigma(\mathcal{M})) - k$ e, portanto, $\mathrm{d}_\pi(\Delta) \leq \mathrm{d}_\pi(\Sigma(\mathcal{M})) - k$ (isso se dá pelo Lema 1.1, no caso $\Sigma_{\mathcal{M}} \neq \varnothing$, pois $\Delta \neq \varnothing$).

[1] Onde $\bigwedge\Delta = (\delta_1 \wedge \cdots \wedge \delta_n)$.

Temos então que $\Box_\pi^k \neg \bigwedge \Delta \in \mathrm{T}_\pi(\Sigma(\mathcal{M}))$ (pela Definição 46.1), onde $\Box_\pi^k \neg \bigwedge \Delta$ é verdadeiro no mundo $w_\mathcal{M}$ em $\mathcal{E}$ (pela Definição 46.2.(iv)). Isso implica que $\neg \bigwedge \Delta$ é verdadeiro no mundo w em $\mathcal{E}$, contradizendo o fato que $\bigwedge \Delta$ é também verdadeiro em $\mathcal{E}$ (pela Definição 49).

Portanto, $\mathrm{D}_\mathcal{M}(w)$ não pode ser $\mathcal{L}_{12}$-inconsistente. ■

Definição 50 (Ancoramento Indireto)**.** Sendo **E** um conjunto de π-modelos e $\overline{\pi}$-modelos etiquetados e sendo $\mathcal{E}_0, \mathcal{E}_1 \in \mathbf{E}$. A relação de ancoramento indireto, dita "$\mathcal{E}_1$ *está indiretamente ancorado em* $\mathcal{E}_0$", é definida indutivamente como:

50.1 Se $\mathcal{E}_1$ está ancorado em $\mathcal{E}_0$, então $\mathcal{E}_1$ está indiretamente ancorado em $\mathcal{E}_0$;

50.2 Se, para algum $\mathcal{E}_2 \in \mathbf{E}$, onde $\mathcal{E}_2$ está ancorado em $\mathcal{E}_0$ e $\mathcal{E}_1$ está indiretamente ancorado em $\mathcal{E}_2$, então $\mathcal{E}_1$ está indiretamente ancorado em $\mathcal{E}_0$. ■

Com essa definição, podemos apresentar um dos conceitos mais importantes para a prova:

Definição 51 (Brotos)**.** Um *broto de* $\mathcal{I}$ é qualquer conjunto **B** de π-modelos e $\overline{\pi}$-modelos etiquetados tal que:

51.1 $\mathcal{I} \in \mathbf{B}$ e $\mathcal{I}$ não está ancorado em nada em **B**;

51.2 Para todo π-modelo etiquetado $\mathcal{E} \in \mathbf{B}$, se $\mathcal{E} \neq \mathcal{I}$ então $\mathcal{E}$ está indiretamente ancorado em $\mathcal{I}$;

51.3 Para cada $\mathcal{E}_0, \mathcal{E}_1 \in \mathbf{B}$, onde $\mathcal{E}_0 \neq \mathcal{E}_1$, $\mathcal{E}_0$ e $\mathcal{E}_1$ só compartilham um mundo se um está ancorado no outro. ■

Um broto pode ser entendido como um conjunto de modelos de tipos alternados, ancorados dois a dois, que inicia em $\mathcal{I}$ e cujo intuito é permitir a valoração de fórmulas com múltiplas modalidades distintas. Podemos então demonstrar algumas propriedades sobre brotos.

Lema 3. *Sendo* **B** *um broto de* $\mathcal{I}$*, então:*

3.1 Para todo $\mathcal{E}_0 \in \mathbf{B}$ e $w \in \mathcal{E}_0$, no máximo um $\mathcal{E}_1 \in \mathbf{B}$ está ancorado em $\mathcal{E}$ no mundo w ;

3.2 Para todo $\mathcal{E}_0 \in \mathbf{B}$, se $\mathcal{E}_0 \neq \mathcal{I}$ então, $\mathcal{E}_0$ está ancorado em exatamente um $\mathcal{E}_1 \in \mathbf{B}$;

3.3 Todo π-modelo etiquetado em **B** tem conjunto de mundos disjuntos[2];

3.4 É chamada de *cadeia de ancoramentos em* $\mathcal{E}$ *de comprimento* n a sequência finita $\langle \mathcal{E}_i \mid i \leq n, n \geq 1 \rangle$ de modelos etiquetados em **B** iniciando com $\mathcal{I}$ e terminando com $\mathcal{E}$ onde, para cada $1 < i \leq n$, $\mathcal{E}_i$ está ancorado em $\mathcal{E}_{i-1}$. Para todo $\mathcal{E} \in \mathbf{B}$, existe exatamente uma cadeia de ancoramentos em $\mathcal{E}$ onde os elementos são disjuntos dois a dois;

3.5 Se $\mathcal{E}_1 \in \mathbf{B}$ está ancorado em $\mathcal{E}_0 \in \mathbf{B}$ em w e $w \neq \boldsymbol{i}$, então w não é o mundo base de $\mathcal{E}_0$;

[2] Ou seja, apenas modelos etiquetados de tipos diferentes compartilham mundos.

3.6 Para todo π-modelo etiquetado $\mathcal{E}_0 = \langle \mathcal{M}, w_{\mathcal{M}}, \Sigma(\mathcal{M}) \rangle \in \mathbf{B}$ e $w \in \mathcal{E}_0$:

(i) Se $\mathcal{E}_0$ está ancorado em $\mathcal{E}_1 = \langle \mathcal{N}, w_{\mathcal{N}}, \Sigma(\mathcal{N}) \rangle$ então $\Sigma(\mathcal{M}) \subseteq \Sigma(\mathcal{N})$;

(ii) $\Sigma(\mathcal{M}) \subseteq \Sigma(\boldsymbol{\mathcal{I}}) \subseteq \Theta$;

(iii) $\mathrm{D}_{\mathcal{M}}(w) \subseteq \Theta$;

(iv) $\mathrm{T}_\pi(\Sigma(\mathcal{M})) \subseteq \mathrm{DC}_\pi(\Theta)$. ■

Prova do Lema 3.

Caso 3.1 Pela Definição 48.1, sabemos que só podemos ancorar um modelo etiquetado de tipo $\pi/\overline{\pi}$ em um modelo etiquetado de tipo $\overline{\pi}/\pi$, pela Definição 48.2 sabemos que estes dois modelos etiquetados compartilham apenas um mundo e, pela Definição 51.3 sabemos que dois modelos etiquetados distinto compartilham mundos apenas quando um está ancorado no outro. Logo, concluímos que, para um mundo $w \in \mathcal{E}_0$, há no máximo um modelo etiquetado $\mathcal{E}_1$ em **B** ancorado neste mundo;

Caso 3.2 Pela Definição 51.2 sabemos que todo π-modelo etiquetado diferente de $\boldsymbol{\mathcal{I}}$ está indiretamente ancorado em $\boldsymbol{\mathcal{I}}$, esse fato, juntamente com as Definições 51.3 e 48.1, nos diz que todo modelo etiquetado em **B** diferente de $\boldsymbol{\mathcal{I}}$ está ancorado em exatamente um outro modelo etiquetado em **B**;

Caso 3.3 Pelas Definições 51.3 e 48.1, temos que todo par de modelos etiquetados de mesmo tipo tem conjuntos de mundos disjuntos;

Caso 3.4 Pela definição indutiva de ancoramento indireto (Definição 50) e pela Definição 51.2, sabemos que essa cadeia de ancoramentos existe. Pelo Lema 3.2, sabemos que todo modelo etiquetado diferente de $\boldsymbol{\mathcal{I}}$ na cadeia tem exatamente um predecessor e, pela Definição 51.1, que nos diz que $\boldsymbol{\mathcal{I}}$ não está ancorado em nenhum modelo etiquetado, sabemos que $\boldsymbol{\mathcal{I}}$ não tem predecessor na cadeia, portanto, pode haver apenas uma cadeia de ancoramentos em $\mathcal{E}$;

Caso 3.5 Decorre diretamente da Definição 48.2;

Caso 3.6.(i) Decorre da Definição 48.3: $\Sigma(\mathcal{N}) = \mathrm{S}_\pi(\Sigma_{\mathcal{M}}(w_{\mathcal{N}}))$ e do fato que $\Sigma_{\mathcal{M}}(w) \subseteq \Sigma(\mathcal{M})$;

Caso 3.6.(ii) Pelos Lemas 3.4 e 3.6.(i) concluímos que $\Sigma(\mathcal{M}) \subseteq \Sigma(\boldsymbol{\mathcal{I}})$, o que é suficiente para provar o caso, pois $\Sigma(\boldsymbol{\mathcal{I}}) = \mathrm{SC}(\Gamma) \subseteq \Theta$;

Caso 3.6.(iii) Pelo Lema 3.6.(ii) e pelos fatos que $\mathrm{D}_{\mathcal{M}}(w) \subseteq \mathrm{B}(\Sigma_{\mathcal{M}}(w))$ e Θ é fechado para B, pela Definição 40;

Caso 3.6.(iv) Pelo Lema 3.6.(ii) e o fato que Θ é fechado para B, podemos concluir que $\mathrm{DC}_\pi(\mathrm{B}(\Sigma(\mathcal{M}))) \subseteq \mathrm{DC}_\pi(\Theta)$ e, pela Definição 46.2.(iv), que nos diz que $\mathrm{T}_\pi(\Sigma(\mathcal{M}))$ é verdadeiro em $w_{\mathcal{M}}$, temos que $\mathrm{T}_\pi(\Sigma(\mathcal{M})) \subseteq \mathrm{DC}_\pi(\mathrm{B}(\Sigma_{\mathcal{M}}))$. ■

Tendo apresentado estas propriedades, é possível observar que um broto de $\mathcal{I}$ pode ser visto como uma árvore, cuja raiz é $\mathcal{I}$ e uma cadeia de ancoramentos para algum modelo etiquetado $\mathcal{E}$ é um ramo na árvore, partindo da raiz e indo até a folha $\mathcal{E}$.

Definição 52 (Função de Seleção de Modelos)**.** Uma *função de seleção de modelo f* é uma função que associa, para cada π e para cada conjunto $\mathcal{L}_\pi$-consistente de fórmulas $\Delta \subseteq \mathrm{DC}_\pi(\Theta)$, um conjunto não vazio $f_\pi(\Delta)$ de pares da forma $\langle \mathcal{E}, w\rangle$ tal que $\mathcal{E}$ é um modelo etiquetado para $\mathcal{L}_\pi$ que satisfaz Δ em w. Para uma dada função de seleção *f*, definimos:

- $\mathbb{M}_f = \{\mathcal{E} \mid \langle \mathcal{E}, w\rangle \in f_\pi(\Delta), \Delta \subseteq \mathsf{LM}_{12} \text{ e } \Delta \text{ é } \mathcal{L}_\pi\text{-consistente}\}$
- $\mathbb{W}_f = \bigcup\{\mathcal{W} \mid \mathcal{W} \in \mathcal{M}, \mathcal{M} \in \mathcal{E}, \mathcal{E} \in \mathbb{M}_f\}$
- $\aleph_f = max\{\aleph_0, sup\left(\{|\mathcal{W}| \mid \mathcal{M} = \langle \mathcal{W}, \mathcal{R}, \mathcal{V}\rangle \in \mathbb{M}_f\}\right)\}$

As funções de seleção de modelos analisadas respeitam as seguintes propriedades:

1. $\mathbb{W}_f$ é um conjunto e $|\mathbb{W}_f| > \aleph_f$;
2. Para cada conjunto $\mathcal{L}_{12}$-consistente $\Delta \subseteq \mathrm{DC}_\pi(\Theta)$, $f_\pi(\Delta)$ é fechado para isomorfismo em (ou seja, entre objetos de) $\mathbb{W}_f$. ■

Para uma dada função *f*, iremos assumir que $\langle \mathcal{I}, i\rangle \in f_\pi(\Gamma \cup \mathrm{T}_\pi(\Sigma(\mathcal{I})))$ e que os mundos de todos os brotos de $\mathcal{I}$ estarão contidos no conjunto $\mathbb{W}_f$.

Definição 53 (Conjunto de Brotos)**.** Um broto de $\mathcal{I}$ é chamado de um *f-broto* de $\mathcal{I}$ se, para cada π-modelo etiquetado no broto, $\langle \mathcal{E}, w_\mathcal{M}\rangle \in f_\pi(\mathrm{DG}(\Sigma(\mathcal{M}) \cup \mathrm{T}_\pi(\Sigma(\mathcal{M}))))$.

Para cada *f*, chamaremos de $\mathfrak{B}_f$ *o conjunto de todos os f-brotos de* $\mathcal{I}$. ■

É importante ressaltar que $\mathfrak{B}_f$ é um conjunto de conjuntos de modelos etiquetados, em específico, é um conjunto de conjuntos que satisfazem as Definição 51 e 53.

Lema 4. *Para cada f, o conjunto* $\mathfrak{B}_f$ *tem um elemento máximo.* ■

Prova do Lema 4. O Lema de Zorn (ZORN, 1935) nos diz que, sendo *X* um conjunto parcialmente ordenado (por alguma relação de ordenamento <), se todo subconjunto totalmente ordenado de *X* tem um limite superior, então *X* tem um elemento máximo (com relação a <).

Considerando $\mathfrak{B}_f$ parcialmente ordenado por $\subseteq$ e sendo *C* um subconjunto[3] totalmente ordenado de $\mathfrak{B}_f$. O conjunto $\bigcup C$ é um limite superior de *C* pois $S \subseteq C$, para todo broto $S \in C$. Mais ainda, $\bigcup C$ satisfaz todas as condições para ser um *f*-broto (é fácil observar que a operação $\cup$ preserva as propriedades 51.1 – 51.3 e 53).

Portanto, $\mathfrak{B}_f$ tem um elemento máximo. ■

[3] Note que *C* é um conjunto de conjuntos.

É importante esclarecer o que significa $\mathfrak{B}_f$ ter um elemento máximo; como $\mathfrak{B}_f$ é um conjunto de brotos e cada broto pode ser entendido como uma árvore cuja raiz é $\mathcal{I}$, o elemento máximo de $\mathfrak{B}_f$ é a maior árvore no conjunto, ou seja, a maior sequência de modelos ancorados dois a dois partindo do modelo inicial. A importância de $\mathfrak{B}_f$ ter um elemento máximo ficará clara a frente.

O elemento máximo de $\mathfrak{B}_f$ respeita a seguinte importante propriedade:

Lema 5. *Sendo $\mathcal{S}^+$ um elemento máximo de $\mathfrak{B}_f$, então, para todo $\mathcal{E}_0 \neq \mathcal{I} \in \mathcal{S}^+$ e mundo não base $w \in \mathcal{E}_0$, então existe um $\mathcal{E}_1 \in \mathcal{S}^+$ ancorado em $\mathcal{E}_0$ no mundo w.* ■

Prova do Lema 5. Assumindo que $\mathcal{S}^+$ é um elemento máximo de $\mathfrak{B}_f$ e assumindo que $\mathcal{E}_0 \neq \mathcal{I} \in \mathcal{S}^+$ é um π-modelo etiquetado, $w \in \mathcal{E}_0$ um mundo não base de $\mathcal{E}_0$ e que não há qualquer $\mathcal{E}_1 \in \mathcal{S}^+$ ancorado em $\mathcal{E}_0$ no mundo w.

Pelo Lema 2, sabemos que o diagrama de concordância $\mathrm{D}_{\mathcal{M}}(w)$ de w em $\mathcal{E}_0$ é $\mathcal{L}_{12}$-consistente. Como $\mathrm{T}_{\overline{\pi}}(\mathrm{S}_{\pi}(\Sigma_{\mathcal{M}}(w)))$ é o um conjunto de teoremas de $\mathcal{L}_{12}$, sabemos que $\mathrm{D}_{\mathcal{M}}(w) \cup \mathrm{T}_{\overline{\pi}}(\mathrm{S}_{\pi}(\Sigma_{\mathcal{M}}(w)))$ é $\mathcal{L}_{12}$-consistente e, portanto, $\mathcal{L}_{\overline{\pi}}$-consistente.

Como $\mathrm{D}_{\mathcal{M}}(w) \cup \mathrm{T}_{\overline{\pi}}(\Sigma(\mathcal{M})) \subseteq \mathrm{DC}_{\overline{\pi}}(\Theta)$ (pelo Lema 3.6) sabemos que $\mathrm{D}_{\mathcal{M}}(w) \cup \mathrm{T}_{\overline{\pi}}(\mathrm{S}_{\pi}(\Sigma_{\mathcal{M}}(w)))$ é verdadeiro em algum mundo w_i de um $\overline{\pi}$-modelo etiquetado $\mathcal{E}_1$ tal que $\langle \mathcal{E}_1, w_i \rangle \in f_{\pi}(\mathrm{D}_{\mathcal{M}}(w) \cup \mathrm{T}_{\overline{\pi}}(\mathrm{S}_{\pi}(\Sigma_{\mathcal{M}}(w))))$. Portanto, sendo $\mathcal{N}$ um modelo onde $\mathcal{N} \in \mathcal{E}_1$ e $\mathcal{W}_{\mathcal{N}} \in \mathcal{N}$, temos que $\mathcal{W}_{\mathcal{N}} \subseteq \mathbb{W}_f$ e $|\mathcal{W}_{\mathcal{N}}| \leq \aleph_f$.

Como f é fechado sob isomorfismo em $\mathbb{W}_f$, podemos tomar w_i como sendo w, declarar w como o mundo base de $\mathcal{E}_1$ e tomar $\Sigma(\mathcal{N}) = \mathrm{S}_{\pi}(\Sigma_{\mathcal{M}}(w))$ (portanto $\mathrm{D}_{\mathcal{M}(w)} = \mathrm{DG}(\Sigma(\mathcal{N}))$).

Como $\mathcal{S}^+ \in \mathfrak{B}_f$, todo modelo etiquetado em $\mathcal{S}^+$ tem, no máximo, cardinalidade[4] $\aleph_f$. Pelo fato que $\mathcal{S}^+$ tem uma estrutura de árvore, sabemos que $\mathcal{S}^+$ contém, no máximo, $(\aleph_f)^{n-1} = \aleph_f$ modelos etiquetados com cadeias de ancoramento de comprimento n. Portanto, $\mathcal{S}^+$ é um união contável de conjuntos de modelos etiquetados cuja cardinalidade é no máximo $\aleph_f$, ou seja, $|\mathcal{S}^+| \leq \aleph_f$.

Portanto, sendo $\mathcal{X} = \bigcup\{\mathcal{W} \mid \mathcal{W} \in \mathcal{M}, \mathcal{M} \in \mathcal{E}, \mathcal{E} \in \mathcal{S}^+\}$, temos $|\mathcal{X}| \leq \aleph_f < |\mathbb{W}_f|$, adicionalmente $|\mathcal{W}_{\mathcal{N}}| \leq \aleph_f < |\mathbb{W}_f|$. Como $f_{\overline{\pi}}(\mathrm{DG}(\Sigma(\mathcal{N}) \cup \mathrm{T}_{\overline{\pi}}(\Sigma(\mathcal{N}))))$ é fechado sob isomorfismo em $\mathbb{W}_f$, podemos assumir que todos os mundos não base de $\mathcal{E}_1$ são elementos do conjunto $\mathbb{W}_f - \mathcal{X}$.

Assim, temos que $\mathcal{E}_1$ satisfaz todas as condições para ser um $\overline{\pi}$-modelo etiquetado ancorado em $\mathcal{E}_0$ no mundo w e que $\mathcal{E}_1$ está indiretamente ancorado em $\mathcal{I}$ (Definição 50). Os mundos não base de $\mathcal{E}_1$ são disjuntos do conjunto $\mathcal{X}$ e $\mathcal{E}_1$ não compartilha o mundo w com qualquer outro modelo diferente de $\mathcal{E}_0$ em $\mathcal{S}^+$ (Definição 51.1 e hipótese), portanto $\mathcal{S}^+ \cup \{\mathcal{E}_1\}$ é um f-broto que contém $\mathcal{S}^+$, ou seja, $\mathcal{S}^+$ não é máximo, o que é contraditório. ■

Note que essa prova não afirma que $\mathcal{S}^+$ será necessariamente infinito. Considere, por exemplo a cadeia de ancoramentos $\mathcal{I}, \ldots, \mathcal{A}, \mathcal{B}$ e considere $\mathcal{W}_{\mathcal{A}(\mathcal{B})}$ o conjunto de mundos

[4] A cardinalidade de um modelo etiquetado se refere a cardinalidade do conjunto de mundos do modelo associado ao modelo etiquetado.

do modelo associado ao modelo etiquetado $\mathcal{A}(\mathcal{B})$, sendo $\mathcal{W}_{\mathcal{A}} = \{w_{\mathcal{A}}, w_{\mathcal{B}}\}$, $\mathcal{W}_{\mathcal{B}} = \{w_{\mathcal{B}}\}$ e $\mathcal{I}, \ldots, \mathcal{A}, \mathcal{B} \in \mathcal{S}^+$. Portanto, existe um modelo etiquetado ancorado em um mundo não base de $\mathcal{A}$ onde não há nenhum outro modelo ancorado, pois seu único mundo é o mundo base.

Definição 54 (Modelos Máximos)**.** Para cada elemento máximo $\mathcal{S}^+$ em $\mathfrak{B}_f$, definimos o seu 12-modelo correspondente $\mathcal{M}(\mathcal{S}^+)$, sendo $\mathcal{E} = \langle \mathcal{M}, w_{\mathcal{M}}, \Sigma(\mathcal{M}) \rangle \in \mathcal{S}^+$ um modelo etiquetado qualquer, como:

$$\begin{aligned}
\mathcal{W}_{\mathcal{M}(\mathcal{S}^+)} &= \bigcup \{\mathcal{W} \mid \mathcal{W} \in \mathcal{M}\} \\
\mathcal{R}_{\pi,\mathcal{M}(\mathcal{S}^+)} &= \bigcup \{\mathcal{R}_\pi \mid \mathcal{R}_\pi \in \mathcal{M} \text{ e } \mathcal{M} \text{ é do tipo } \pi\} \\
\mathcal{V}_{\mathcal{M}(\mathcal{S}^+)}(p) &= \bigcup \{\mathcal{V}_\pi(p) \mid \mathcal{V}_\pi \in \mathcal{M}\}
\end{aligned}$$

Este modelo é chamado de *modelo $\mathfrak{B}_f$-máximo* ou simplesmente *modelo máximo*. ■

Com essa definição, temos a seguinte propriedade importante:

Lema 6. *Sendo $\mathcal{M}(\mathcal{S}^+) = \langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2, \mathcal{V} \rangle$ um modelo máximo, então:*

6.1 Todo $w \in \mathcal{W}$ pertence exatamente a um modelo $\mathcal{M}^1(w)$, este que pertence a um 1-modelo etiquetado $\mathcal{E}_0 \in \mathcal{S}^+$, e a um modelo $\mathcal{M}^2(w)$, este que pertence a um 2-modelo etiquetado $\mathcal{E}_1 \in \mathcal{S}^+$, onde $\mathcal{E}_0/\mathcal{E}_1$ está ancorado em $\mathcal{E}_1/\mathcal{E}_0$ no mundo w;

6.2 O π-frame $\langle \mathcal{W}, \mathcal{R}_\pi \rangle$ é a união disjunta de todos os π-frames $\langle \mathcal{W}, \mathcal{R}_\pi \rangle \in \mathcal{M}$ onde $\mathcal{M} \in \mathcal{E}$ e $\mathcal{E} \in \mathcal{S}^+$. ■

Prova do Lema 6.

Caso 6.1 Pela definição de $\mathcal{M}(\mathcal{S}^+)$, sabemos que w está em algum π-modelo $\mathcal{M}_\pi(w)$. Então temos dois casos para considerar:

(i) $w = \boldsymbol{i}$ ou $w \neq \boldsymbol{i}$ e w não é o mundo base de $\mathcal{M}_\pi(w)$.
Nesse caso, temos que w está em algum $\overline{\pi}$-modelo $\mathcal{M}_{\overline{\pi}}(w)$ que está ancorado em $\mathcal{M}_\pi(w)$ no mundo w, pelo Lema 5.

(ii) $w \neq \boldsymbol{i}$ e w é o mundo base de $\mathcal{M}_\pi(w)$ (portanto $\mathcal{M}_\pi(w) \neq \mathcal{I}$).
Nesse caso, w está em algum $\overline{\pi}$-modelo $\mathcal{M}_{\overline{\pi}}(w)$ tal que $\mathcal{M}_\pi(w)$ está ancorado em $\mathcal{M}_{\overline{\pi}}(w)$ no mundo w, pelo Lema 3.2.

Em ambos os casos, temos que w está em pelo menos um $\mathcal{M}^1(w)$ e um $\mathcal{M}^2(w)$, um ancorado no outro em w. A unicidade de $\mathcal{M}^1(w)$ e $\mathcal{M}^2(w)$ é consequência direta do Lema 3.3;

Caso 6.2 Pelo Lema 6.1, sabemos que $\mathcal{W} = \bigcup \{\mathcal{W} \mid \mathcal{M} \in \mathcal{E}, \mathcal{E} \in \mathcal{S}^+ \text{ e } \mathcal{M} \text{ é do tipo } \pi\}$ é valido para $\pi \in \{1, 2\}$, já o Lema 3.3 nos diz que os conjuntos de mundos de π-modelos são mutuamente disjuntos. ■

Os principais resultados relacionados ao modelo $\mathcal{M}(\mathcal{S}^+)$ são o *Lema de Concordância* e o *Lema do Frame*. O Lema da Concordância nos diz que o modelo "grande" $\mathcal{M}(\mathcal{S}^+)$ concorda com todos os modelos "pequenos" $\mathcal{M}_\pi(w)$ na valoração de todos os componentes booleanos de elementos em $\Sigma_{\mathcal{M}_\pi(w)}(w)$ (lembre-se das Definições 33 e 36). No que segue, escrevemos $\mathcal{M}_\pi$ ao invés de $\mathcal{M}_\pi(w)$ quando não for necessário indicar w.

Lema 7 (Lema de Concondância)**.** *Sendo $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2, \mathcal{V} \rangle$ um modelo máximo. Então, para cada $w \in \mathcal{W}$ e $\varphi \in \mathrm{B}(\Sigma_{\mathcal{M}_\pi}(w))$, φ será verdadeiro em w no π-modelo $\mathcal{M}_\pi = \langle \mathcal{W}_\pi, \mathcal{S}_\pi, \mathcal{V}_\pi \rangle$ se, e somente se, φ for verdadeiro em w no 12-modelo $\mathcal{M}$.*

Prova do Lema 7. Provaremos por indução em φ:

Caso Base $\varphi := p, p \in \mathbb{P}$: Dividimos essa prova em dois casos:

- ($\Rightarrow$) Como $\mathcal{M}_\pi, w \Vdash p$ temos que $w \in \mathcal{V}_\pi(p)$, logo $w \in \mathcal{V}(p)$ pela Definição 54;
- ($\Leftarrow$) Como $\mathcal{M}, w \Vdash p$ sabemos que $w \in \mathcal{V}(p)$, logo temos que $w \in \mathcal{V}^1_\pi(p)$ para algum $\mathcal{N} = \langle \mathcal{W}^1_\pi, \mathcal{S}^1_\pi, \mathcal{V}^1_\pi \rangle \in \mathcal{S}^+$, pela Definição 54, onde $\mathcal{N}$ deve ser o modelo $\mathcal{M}^1(w)$ ou o modelo $\mathcal{M}^2(w)$ no Lema 6.1. O Lema 6.1 também nos diz que um modelo deve estar ancorado no outro no mundo w e, qualquer que seja o caso, temos que $p \in \Sigma_{\mathcal{M}^1}(w)$ e $p \in \Sigma_{\mathcal{M}^2}(w)$, pela Definição 48.3. A Definição 48.4 nos diz que $\mathcal{M}^1, w \Vdash p$ sse $\mathcal{M}^2, w \Vdash p$, ou seja, temos $\mathcal{M}_\pi, w \Vdash p$.

Caso $\varphi := \neg\psi$ e $\varphi := \psi \vee \gamma$: Decorrem diretamente da hipótese de indução.

Caso $\varphi := \Box_\pi \psi$: Temos a seguinte hipótese de indução:

$$\forall w, \psi, \psi \in \mathrm{B}(\Sigma_{\mathcal{M}_\pi}(w)) \wedge \mathcal{M}_\pi, w \Vdash \psi \Leftrightarrow \mathcal{M}, w \Vdash \psi$$

Sabemos que $\mathcal{M}_\pi, w \Vdash \Box_\pi \psi$ será válido sse $\forall v, w \mathcal{S}_\pi v \rightarrow \mathcal{M}_\pi, v \Vdash \psi$ e, pelo Lema 6.2, sabemos que isso será válido sse $\forall v, w \mathcal{R}_\pi v \rightarrow \mathcal{M}_\pi, v \Vdash \psi$[5] for válido. Pelo Lema 6.1, sabemos que caso $w \mathcal{R}_\pi v$ então $\mathcal{M}_\pi = \mathcal{M}_\pi(w) = \mathcal{M}_\pi(v)$, ou seja, w e v estão no mesmo π-modelo.

Ademais, temos que $\psi \in \mathrm{B}(\Sigma_{\mathcal{M}_\pi}(v))$ pois $\mathrm{TC}(\psi) \subseteq \Sigma(\mathcal{M}_\pi(v))$ (no caso onde $w \neq v$, isso é devido a Definição 46.3.(ii), já no caso onde $w = v$, isso é devido as Definições 46.2.(iii) e 46.3.(i)). Como $\psi \in \mathrm{B}(\Sigma_{\mathcal{M}_\pi}(v))$, podemos aplicar a hipótese de indução em $\mathcal{M}_\pi, v \Vdash \psi$, portanto, temos: $\forall v, w \mathcal{R}_\pi v \rightarrow \mathcal{M}, v \Vdash \psi$, que será verdadeiro apenas se $\mathcal{M}, w \Vdash \Box_\pi \psi$, portanto provamos o caso.

Caso $\varphi := \Box_{\overline{\pi}} \psi$: Temos a seguinte hipótese de indução:

$$\forall w, \psi, \psi \in \mathrm{B}(\Sigma_{\mathcal{M}_\pi}(w)) \wedge \mathcal{M}_\pi, w \Vdash \psi \Leftrightarrow \mathcal{M}, w \Vdash \psi$$

[5] Note que $\mathcal{R}_\pi$ se refere às relações $\mathcal{R}_1$ e $\mathcal{R}_2$ do modelo $\mathcal{M}$.

Vamos assumir que $\Box_{\overline{\pi}}\psi \in \Sigma_{\mathcal{M}_{\pi}}(w)$. Temos que ou $\mathcal{M}_{\pi}$ está ancorado em $\mathcal{M}_{\overline{\pi}}$ em w ou vice versa. Em ambos os casos, o fato que $\Box_{\overline{\pi}}\psi \in \Sigma_{\mathcal{M}_{\overline{\pi}}}(w)$ decorre da Definição 48.3 (no primeiro caso, pelo Lema 3.6.(i), no segundo caso pelo fato que $\Box_{\overline{\pi}}\psi$ é um π-elemento). Pela Definição 48.4 temos que $\mathcal{M}_{\pi}, w \Vdash \Box_{\overline{\pi}}\psi$ sse $\mathcal{M}_{\overline{\pi}}, w \Vdash \Box_{\overline{\pi}}\psi$, sendo que $\mathcal{M}_{\overline{\pi}}, w \Vdash \Box_{\overline{\pi}}\psi$ será verdadeiro sse $\mathcal{M}, w \Vdash \Box_{\overline{\pi}}\psi$, o que pode ser provado de forma análoga ao caso anterior, se substituirmos $\Box_{\pi}$ por $\Box_{\overline{\pi}}$. ■

Definição 55 (Condição de Frame)**.** Uma função *f satisfaz a condição de frame* se, para cada conjunto de fórmulas $\mathcal{L}_{\pi}$-consistente $\Delta \subseteq \mathrm{DC}_{\pi}(\Theta)$, os modelos em $f_{\pi}(\Delta)$ são baseados em frames para $\mathcal{L}_{\pi}$. ■

Uma função f que satisfaz a condição de frame existe quando $\mathcal{L}_1$ e $\mathcal{L}_2$ são completas com relação à $\mathrm{DC}_{\pi}(\Theta)$. A existência dessa função se dá pela completude de $\mathcal{L}_1$ e $\mathcal{L}_2$ - como $\mathcal{L}_{\pi}$ é completa, temos que todo $\Delta \subseteq \mathrm{DC}_{\pi}(\Theta)$ é satisfazível em algum modelo baseado num frame para $\mathcal{L}_{\pi}$, logo, pela definição de função de seleção de modelos, sabemos que todos os modelos em $f_{\pi}(\Delta)$ são baseados em frames para $\mathcal{L}_{\pi}$.

Lema 8 (Lema do Frame)**.** *Sendo $\mathcal{M} = \langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2, \mathcal{V} \rangle$ um modelo $\mathfrak{B}_f$-máximo onde f satisfaz a Definição 55. Então $\langle \mathcal{W}, \mathcal{R}_1, \mathcal{R}_2 \rangle$ é um 12-frame para $\mathcal{L}_{12}$* ■

Prova do Lema 8. Sabemos, pela definição de união disjunta e pela definição de frames, que conjuntos de frames são fechados para a operação de união disjunta[6]. Com isso e o Lema 6.2, temos que o par $\langle \mathcal{W}, \mathcal{R}_1 \rangle$ descreve um frame para a lógica $\mathcal{L}_1$ e o par $\langle \mathcal{W}, \mathcal{R}_2 \rangle$ descreve um frame para a lógica $\mathcal{L}_2$. Pela definição de 12-frames e pelo Teorema 3, concluímos a prova. ■

Como $\Gamma \subseteq \mathrm{B}(\Sigma_{\mathcal{I}}(\boldsymbol{i}))$ é verdadeiro em $\boldsymbol{i}$ no modelo $\mathcal{I} = \mathcal{M}^1(\boldsymbol{i})$, o Lema 7 nos diz que Γ é verdadeiro em $\boldsymbol{i}$ no 12-modelo $\mathcal{M}$. Como $\mathcal{L}_1$ e $\mathcal{L}_2$ são completas, podemos assumir alguma função de seleção de modelos f que satisfaz a Definição 55; portanto, o frame que define $\mathcal{M}$ é um frame para $\mathcal{L}_{12}$, pelo Lema 8.

Portanto, com este método conseguimos construir um modelo que satisfaz todo subconjunto $\mathcal{L}_{12}-$consistente do espaço de fórmulas Θ. Assim, concluímos a prova do Teorema 4. ■

[6] Por exemplo: $\mathcal{F}_1 = \langle \{w_0, w_1\}, \{\langle w_0, w_1\rangle, \langle w_0, w_0\rangle, \langle w_1, w_1\rangle\}\rangle$ e $\mathcal{F}_2 = \langle \{w_2, w_3\}, \{\langle w_2, w_3\rangle\}\rangle$, sua união disjunta será o frame $\mathcal{F}_{12} = \langle \{w_{0_1}, w_{1_1}, w_{2_2}, w_{3_2}\}, \{\langle w_{0_1}, w_{1_1}\rangle, \langle w_{0_1}, w_{0_1}\rangle, \langle w_{1_1}, w_{1_1}\rangle, \langle w_{2_2}, w_{3_2}\rangle\}\rangle$.