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

Implementação da Combinação de Métodos de Prova Automática de Teoremas para Lógicas Modais

Karen Lima Macêdo

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

Orientadora
Prof.a Dr.a Cláudia Nalon

Brasília
2023

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

Implementação da Combinação de Métodos de Prova Automática de Teoremas para Lógicas Modais

Karen Lima Macêdo

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

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

Prof. Dr. Flávio Leonardo Cavalcanti de Moura
CIC/UnB

Prof. Dr. Bruno Lopes Vieira
IC/UFF

Prof. Dr. Marcelo Grandi Mandelli
Coordenador do Bacharelado em Ciência da Computação

Brasília, 21 de julho de 2023

Dedicatória

Dedico este trabalho à memória da minha avó Djanira que não conseguiu vê-lo finalizado mas enquanto pôde esteve ao meu lado em todas as comemorações, momentos importantes e difíceis durante a minha vida, e nunca mediu esforços para me ver bem e feliz. Dedico também à minha namorada Maria Antônia que me apoiou durante todo o desenvolvimento deste projeto, se faz presente em cada passo, me inspira e me ajuda a superar toda dificuldade.

Agradecimentos

Agradeço à minha mãe por todo esforço realizado para que eu pudesse realizar meus sonhos e pelo amor incondicional, sem ela eu não teria chegado até aqui. Agradeço também à minha irmã por todos os momentos de distração e alegria que me reenergizaram a continuar o desenvolvimento deste trabalho.

Agradeço à minha orientadora Cláudia Nalon por toda paciência, compreensão, por todo o aprendizado e por ter acreditado em mim.

Agradeço aos meus familiares, em especial minha tia Dalva e minhas primas Tainá e Maria Clara, que acompanharam de perto minha evolução e fizeram todo o possível para me ajudar e apoiar quando eu precisei.

Agradeço também aos meus amigos Andrey e Matheus que mesmo distantes se fizeram presentes e que eu sempre pude contar independente do momento e da situação.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Este trabalho apresenta as modificações feitas no KSP, um provador de teoremas baseado em resolução para a Lógica Multimodal K_n, para permitir que o provador também lide com a combinação dos raciocínios local e global. Para o completo entendimento do leitor, a sintaxe e semântica da Lógica Multimodal K_n são descritas, assim como o cálculo utilizado no KSP e sua implementação. Uma avaliação experimental foi executada durante o desenvolvimento deste projeto para garantir que a performance do provador não sofresse alteração significativa, que o provador continuasse lidando com os raciocínios local e global corretamente e que a implementação da combinação dos dois raciocínios resultasse em concordância com o cálculo. Os resultados obtidos mostram que o tempo de execução para os cálculos local e global separadamente foi semelhante antes e após as modificações do provador para o mesmo conjunto de testes. Também foi possível concluir que as saídas sobre a satisfatibilidade foram iguais às esperadas para este conjunto de testes. Em relação aos testes executados para a combinação dos dois cálculos, parte do conjunto de cláusulas cumpriram a expectativa porém um dos testes apresentou erros. Por isso, a refatoração do provador e uma nova experimentação são necessárias.

Palavras-chave: Lógica Modal, Resolução, Prova Automática de Teoremas, Implementação, KSP

Abstract

In this work we present the modifications made to KSP, a resolution-based theorem prover for Multimodal Logic K_n, to allow for the combination of local and global reasoning. For the complete understanding of the reader, the syntax and semantics of Multimodal Logic K_n is described, as well as the calculus and its implementation in KSP. An experimental evaluation was carried out during the development of this project to ensure that the performance of the prover had not changed significantly, that the prover continued to handle local and global reasoning correctly, and that the implementation of the combination of the two forms of reasoning was faithful to the calculus. The results obtained show that the execution time for the local and global calculations separately was similar to the time of the original prover for the same set of tests. It was also possible to conclude that the outputs on satisfiability were the same as expected when considering the same benchmark. Regarding the tests performed for the combination of the two calculations, part of the set of clauses met the expectation but one of the tests presented errors. Because of this, refactoring of the prover and new experimentation are necessary.

Keywords: Modal Logic, Resolution, Automatic Theorem Proving, Implementation, KSP

Sumário

1 Introdução **1**

2 A Lógica Multimodal K_n **3**
2.1 Sintaxe 4
2.2 Semântica 8

3 Resolução **12**
3.1 Forma Normal Separada com Níveis Modais 12
3.2 Regras de Inferência 17

4 $\mathbf{K_S P}$ **23**
4.1 Laço Principal 23
4.2 Processamento da Entrada 25
4.3 Tradução para a Forma Normal 27
4.4 Pré-processamento de Cláusulas 27
4.5 Seleção de Cláusulas 28
4.6 Refinamentos 28
4.7 Eliminação de Redundância 29

5 Implementação e Avaliação Experimental **30**
5.1 Implementação 30
5.2 Experimentação 32

6 Conclusão **37**

Referências **38**

Anexo **39**

I Exemplo **40**

Lista de Figuras

I.1 Arquivo de entrada para o KSP [1, Exemplo 4.6, Páginas 12 a 14] 40

Lista de Tabelas

3.1 Regras de reescrita para obtenção da NNF 13
3.2 Regras de simplificação 14
3.3 Regras de simplificação para cláusulas rotuladas, onde $ml \in \mathbb{N} \cup \{*\}$, l é um literal e D é uma cláusula proposicional (possivelmente vazia) 15
3.4 Função de tradução para Forma Normal Separada 15
3.5 Função de tradução para SNF^{+}_{ml} 16
3.6 Função de tradução para SNF^{++}_{ml} 16
3.7 Regras de Inferência, onde $ml = \sigma(\{ml_1, ..., ml, ml_{m+1}, ml_{m+2} - 1\})$ em GEN1, GEN3, onde $m \geq 0$; $ml = \sigma(\{ml_1, ml_2\})$ em LRES, MRES; e $ml = \sigma(\{ml_1, ml_2, ml_3\})$ em GEN2. 18
3.8 Regra da Resolução Semântica, onde l é um literal, e D e D' são disjunções de literais 21

4.1 Função de tradução para Forma Normal Antiprenex 26
4.2 Regra de propagação para os literais l e l', nível modal ml, agente a, com a condição que existe somente uma cláusula modal negativa em Λ^{modal}_{ml}. . . 28

5.1 Arquivos de Configuração 36
5.2 Média das diferenças e diferença máxima entre os testes 36

Capítulo 1

Introdução

Raciocínio Automatizado é a área da inteligência artificial que estuda formas de automatizar o processo de dedução por meio de sistemas computacionais, isto é, o processo de inferir a partir de premissas por meio de um raciocínio lógico e chegar a uma conclusão. Para isso, são utilizados símbolos e fórmulas para representar fatos. As fórmulas e seus significados compreendem o que é chamado de linguagem lógica. A atribuição de significado a uma formula é denominada interpretação. O problema da satisfatibilidade consiste em determinar se existe uma interpretação sob a qual a fórmula é verdadeira.

A verificação da implementação de sistemas é um exemplo de problema de satisfatibilidade e um caso de uso do Raciocínio Automatizado. Quando é possível derivar as fórmulas das propriedades a partir do conjunto de sentenças que representam a implementação usando um método de prova, pode-se considerar que o sistema foi verificado e cumpre seu objetivo. Para a construção do processo dedutivo são usados os cálculos. Eles são constituídos por um conjunto de axiomas, ou seja, um conjunto de fórmulas consideradas verdadeiras, e um conjunto de regras de inferência. Denomina-se prova à sequência de fórmulas obtidas a partir de axiomas ou da aplicação das regras de inferência; teoremas são fórmulas para as quais existe uma prova.

A Lógica Modal possibilita fazer a caracterização de sistemas complexos e sistemas distribuídos [2]. Ela estende a Lógica Proposicional e com isso permite considerar múltiplas interpretações dentro de um determinado cenário. A Lógica Multimodal K_n [3, 4] é uma generalização da Lógica Modal que adiciona ao conjunto de operadores da Lógica Clássica Proposicional um conjunto de operadores modais: $[a]$, o operador de necessidade, e $\langle a \rangle$, o operador de possibilidade, com $a \in \mathcal{A} = \{1, ..., n\}$. A fórmula $[a]\varphi$ lê-se "a proposição φ é necessária segundo a interpretação do agente a" e $\langle a \rangle \varphi$ lê-se "a proposição φ é possível segundo a interpretação do agente a".

Na Lógica Multimodal, a interpretação é dada por uma estrutura relacional, ou seja um conjunto e um conjunto de relações binárias sobre este conjunto, chamada *modelo*.

Os elementos do conjunto são chamados *mundos* e a relação, para cada agente $a \in A_n$, é dita *acessibilidade* entre mundos. Os operadores de necessidade e possibilidade são interpretados da seguinte forma: a sentença $[a]\varphi$ é verdadeira no mundo w se φ é verdadeira em todos os mundos acessíveis a partir de w segundo a interpretação do agente a; $\langle a\rangle\varphi$ é verdadeira no mundo w se φ é verdadeira em algum mundo acessível a partir de w segundo a interpretação do agente a. A satisfação local de uma fórmula determina o significado dessa fórmula a partir de um único mundo do modelo, enquanto satisfação global faz esta determinação a partir de todos os mundos do modelo. O problema de satisfatibilidade local para K_n é PSPACE-completo [5], ou seja, a Lógica Multimodal pode ser usada para representar qualquer problema pertencente à classe PSPACE. O problema de satisfatibilidade global para K_n é EXPTIME-completo [6]. A combinação dos dois tipos de raciocínio preserva a complexidade do problema de satisfatibilidade global, isto é, é EXPTIME-completo.

Resolução é um exemplo de cálculo refutacional [7]. Esse cálculo faz uso do metateorema: uma fórmula φ é insatisfatível se, e somente se, é refutável. Ou seja, nega-se a fórmula e tenta-se achar uma contradição. O KSP [8] é um provador de teoremas baseado em resolução para a Lógica Multimodal Básica K_n que implementa cálculos baseados em resolução para lidar com o problema de satisfatibilidade local e global. Porém a combinação entre esses dois tipos de raciocínios ainda não está implementada.

O objetivo deste projeto é continuar a implementação de interfaces e módulos para viabilizar essa combinação que iniciou-se como um projeto de iniciação científica (2020-2021 e 2021-2022). Para isso, foi necessário refatorar as funções do provador que implementam as regras de inferência do cálculo baseado em resolução para que elas recebam como parâmetro os conjuntos em que as fórmulas candidatas devem ser procuradas. Além disso, essas funções devem continuar capazes de lidar com os raciocínios local e global separadamente.

O Capítulo 2 apresenta a sintaxe e semântica da Lógica Multimodal K_n. Em seguida, no Capítulo 3, introduz-se cálculo baseado em resolução que é implementado no KSP. No Capítulo 4 descreve-se a implementação do provador. O Capítulo 5 descreve as modificações feitas e relata os experimentos e resultados obtidos. No último capítulo, as conclusões são apresentadas e sugestões para trabalhos futuros são descritas.

Capítulo 2

A Lógica Multimodal K_n

Utilizando a Lógica Proposicional, as sentenças "Está chovendo" e "É seguro dirigir" podem ser utilizadas na formação de sentenças mais complexas, como, por exemplo, "Se está chovendo, então não é seguro dirigir" da seguinte forma "$p \rightarrow \neg q$", sendo que p representa a primeira sentença e q, a segunda. Nesse caso, apenas um contexto é representado: não é seguro dirigir durante a chuva. Porém, se a intensidade da chuva é fraca, é seguro dirigir e a Lógica Modal permite formalizar esses outros contextos utilizando, por exemplo, a sentença "Se está chovendo, então é possível que seja seguro dirigir" formalizada da seguinte forma: "$p \rightarrow \langle\rangle q$", onde $\langle\rangle$ representa "possibilidade".

Além disso, sabe-se também que independente da intensidade, algumas pessoas não consideram que dirigir é seguro durante a chuva. Para esses casos, pode-se utilizar a Lógica Multimodal e formalizar a sentença anterior da seguinte forma: $p \rightarrow \langle a\rangle q$, que representa o fato "Se está chovendo, então o agente a considera que é seguro dirigir".

A expressividade da Lógica Clássica Proposicional é definida através do problema da satisfatibilidade booleana, que pertence à classe NP [9]. Já a expressividade da Lógica Multimodal pode ser caracterizada através dos problemas de satisfatibilidade local e global, sendo o primeiro pertencente à classe PSPACE [5] e o segundo à classe EXPTIME [6]. A combinação de raciocínio local e global preserva a complexidade do problema global, isto é, está em EXPTIME.

Por esse motivo, a Lógica Multimodal é considerada mais expressiva que a Lógica Clássica Proposicional. Todos os problemas representados através da Lógica Proposicional podem ser representados utilizando a Lógica Multimodal. Isso fica mais claro sabendo que a Lógica Multimodal é uma extensão da Lógica Proposicional com a adição de dois novos símbolos na sua sintaxe. A apresentação dessa sintaxe, isto é, os símbolos e as combinações consideradas bem formadas, e da semântica, a interpretação das sentenças bem formadas, é o intuito desse capítulo.

2.1 Sintaxe

Como dito acima, a sintaxe é responsável por apresentar os símbolos da linguagem e definir regras sintáticas para construção das fórmulas. Isso garante que o conjunto de sentenças bem formadas seja definido de forma não ambígua e, consequentemente, possam ser interpretadas.

Sejam $\mathcal{P} = \{p, q, r, p', q', ...\}$ um conjunto enumerável de símbolos proposicionais, $\mathcal{A} = \{1, ..., n\}, n \in \mathbb{N}$ um conjunto fixo de índices, o conjunto de operadores da Lógica Clássica Proposicional $\{\wedge, \neg, \vee, \rightarrow$ e $\leftrightarrow\}$, e o conjunto de operadores modais, $[a]$ e $\langle a \rangle$, com $a \in \mathcal{A}$.

Definição 1 O conjunto de fórmulas bem formadas, denotado por $\mathsf{WFF}_{\mathsf{K}_n}$, é o menor conjunto tal que:

1. se $p \in \mathcal{P}$, então $p \in \mathsf{WFF}_{\mathsf{K}_n}$
2. se φ e $\psi \in \mathsf{WFF}_{\mathsf{K}_n}$, então $\neg\varphi$, $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, $[a]\varphi$ e $\langle a \rangle \varphi \in \mathsf{WFF}_{\mathsf{K}_n}$, para cada $a \in \mathcal{A}$.

As seguintes abreviações também serão usadas, além dos operadores dados na Definição 1:

1. **false** $= (\varphi \wedge \neg\varphi)$
2. **true** $= \neg$**false**
3. $(\varphi \rightarrow \psi) = (\neg\varphi \vee \psi)$
4. $(\varphi \leftrightarrow \psi) = ((\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi))$

Por esse motivo, os operadores proposicionais $\rightarrow$ e $\leftrightarrow$, e os símbolos $\top$ e $\bot$ não foram incluídos na Definição 1.

Exemplo 1 Um exemplo de fórmula bem formada seria: $([1](p \wedge [2]q) \rightarrow [2](p \wedge q))$

Exemplo 2 Um exemplo de fórmula mal formada seria: $([1](p \wedge [2]q) \rightarrow \vee[2](p \wedge q))$

Quando o conjunto $\mathcal{A}$ de agentes da lógica é unitário, o agente é omitido, ou seja, escreve-se $[]$ e $\langle\rangle$.

Parênteses podem ser omitidos das fórmulas. Nesses casos, obedeceremos a seguinte ordem de precedência, dos operadores de maior precedência para os operadores de menor precedência:

1. $\neg, [a], \langle a \rangle$
2. $\wedge$
3. $\vee$
4. $\rightarrow$
5. $\leftrightarrow$

Exemplo 3 Obedecendo a ordem de precedência acima, a fórmula:
$[][]\,(p \vee \neg[]p \wedge \neg[]\neg q\,)$
seria reescrita, exatamente em acordo com a Definição 1, da seguinte forma:
$[][]\,(p \vee\ (\neg[]p \wedge \neg[]\neg q\,)\,)$.

Outro conceito importante para este trabalho é a definição de cláusulas proposicionais, visto que o cálculo, apresentado no Capítulo 3 e que é implementado no KSP, é clausal.

Definição 2 Um *literal* é um símbolo proposicional ou sua negação. Denota-se por $\mathcal{L}$ o conjunto de literais. O *complemento* de um literal l é denotado por $\neg l$. Um *literal modal* é $[a]l$ ou $\langle a \rangle l$, onde $l \in \mathcal{L}$ e $a \in \mathcal{A}$.

Definição 3 Uma *cláusula proposicional* é uma disjunção de literais $\bigvee_{b=1}^{r} l_b$.

Exemplo 4 Segundo as Definições 2 e 3 acima e sendo que p e $q \in \mathcal{P}$, exemplos de literal, literal modal e cláusula proposicional seriam:
literal: q e $\neg p$
literal modal: $[a]q$ e $[a]\neg p$
cláusula proposicional: $\neg q \vee p \vee r$

O conjunto de subfórmulas de uma fórmula é definido como usual.

Definição 4 O conjunto de subfórmulas da fórmula proposicional φ é definido recursivamente de acordo com os seguintes itens:

- φ é uma subfórmula dela mesma.
- Se $\varphi = (\,\neg\gamma)$ então todas as subfórmulas de γ são subfórmulas de φ.

- Se $\varphi = (\,\psi \circ \gamma\,)$, onde $\circ \in \{\vee, \wedge, \rightarrow\}$, então todas as subfórmulas de ψ e γ são subfórmulas de φ.
- Se $\varphi = (\,\circ\gamma\,)$, onde $\circ \in \{[a], \langle a\rangle\}$, então todas as subfórmulas de γ são subfórmulas de φ.

Como dito anteriormente, o objetivo desse trabalho é lidar com o problema da satisfatibilidade utilizando um cálculo baseado em resolução. As regras de inferência desse cálculo requerem cláusulas identificadas por níveis modais. O nível modal de uma subfórmula é dado de acordo com sua posição na árvore sintática da superfórmula.

Definição 5 Sejam φ, φ' fórmulas bem-formadas. Seja Σ o alfabeto $\{1,\ 2,\ .\}$ e Σ^* o conjunto de todas as sequências finitas sobre Σ. A sequência vazia em Σ^* é denotada por ε. Seja $\tau : \mathsf{WFF}_{\mathsf{K}_n} \times \Sigma^* \times \{+,-\} \times \mathbb{N} \rightarrow 2^{WFF_{K_n} \times \Sigma^* \times \{+,-\} \times \mathbb{N}}$ a função parcial definida indutivamente como segue (onde $\lambda \in \Sigma^*$, $pol \in \{+,-\}$, $ml \in \mathbb{N}$ e o complemento do símbolo em $\{+,-\}$ é dado por $\mathsf{comp}(+) = -$, $\mathsf{comp}(-) = +$):

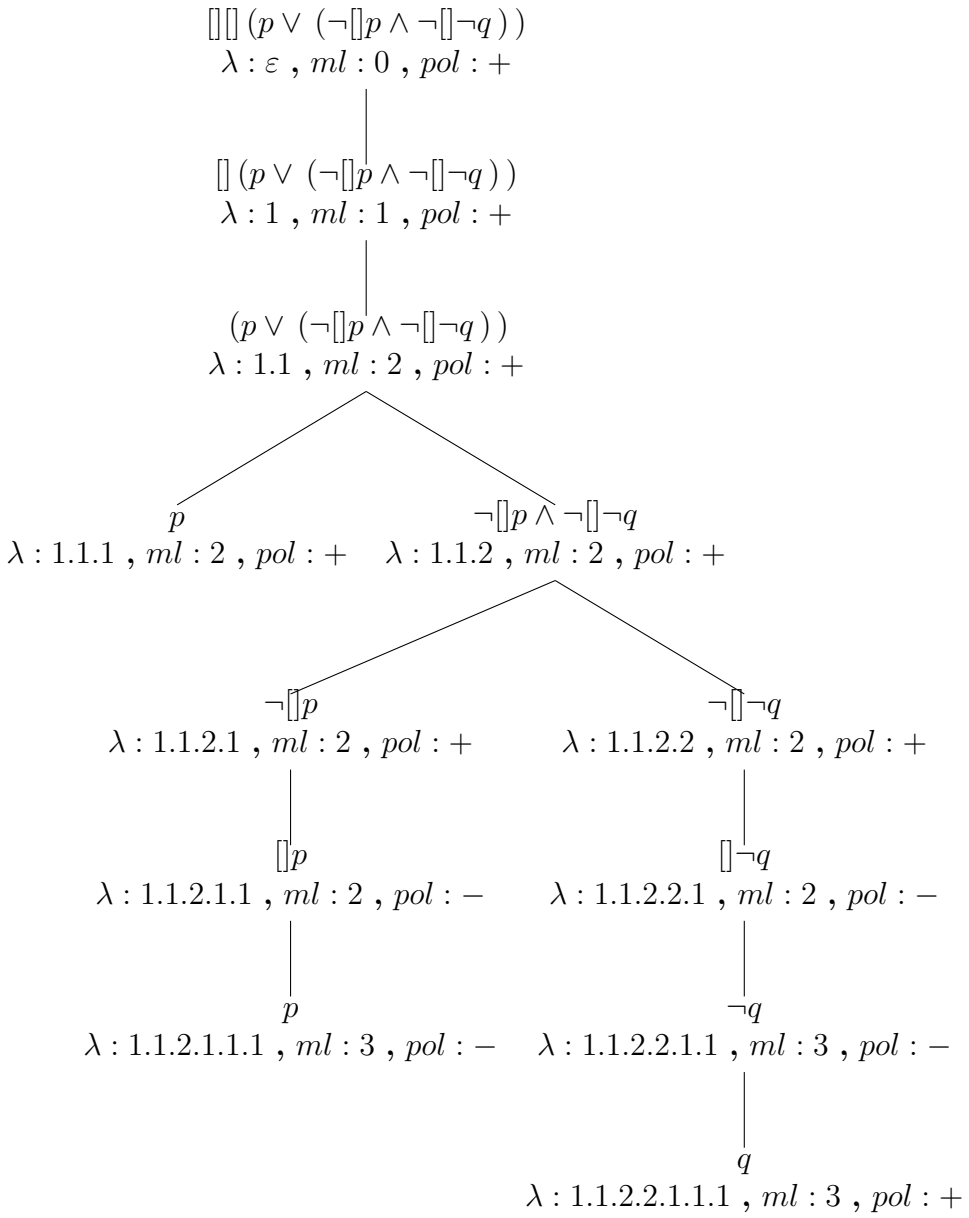
- $\tau\,(p, \lambda, pol, ml\,) = \{\,(p, \lambda, pol, ml\,)\}$, para $p \in \mathcal{P}$;
- $\tau\,(\neg\varphi, \lambda, pol, ml\,) = \{\,(\neg\varphi, \lambda, pol, ml\,)\} \cup \tau\,(\varphi, \lambda.1, \mathsf{comp}(pol)\,, ml)$;
- $\tau\,([a]\varphi, \lambda, pol, ml\,) = \{\,([a]\varphi, \lambda, pol, ml\,)\} \cup \tau\,(\varphi, \lambda.1, pol, ml + 1)$;
- $\tau\,(\langle a\rangle\varphi, \lambda, pol, ml\,) = \{\,(\langle a\rangle\varphi, \lambda, pol, ml\,)\} \cup \tau\,(\varphi, \lambda.1, pol, ml + 1)$;
- $\tau\,(\varphi \ \wedge\ \varphi', \lambda, pol, ml\,) = \{\,(\varphi \ \wedge\ \varphi', \lambda, pol, ml\,)\} \cup \tau\,(\varphi, \lambda.1, pol, ml) \cup \tau\,(\varphi', \lambda.2, pol, ml)$;
- $\tau\,(\varphi \ \vee\ \varphi', \lambda, pol, ml\,) = \{\,(\varphi \ \vee\ \varphi', \lambda, pol, ml\,)\} \cup \tau\,(\varphi, \lambda.1, pol, ml) \cup \tau\,(\varphi', \lambda.2, pol, ml)$;

O resultado de $\tau\,(\varphi, \varepsilon, +, 0\,)$ é a *árvore sintática anotada de* φ. Cada nó da árvore sintática anotada é identificado unicamente por uma subfórmula, sua posição na árvore, sua polaridade e seu nível modal.

Definição 6 Seja φ uma fórmula e $\tau\,(\varphi, \varepsilon, +, 0\,)$ sua árvore sintática anotada. Se $(\varphi', \lambda, pol, ml\,) \in \tau\,(\varphi, \varepsilon, +, 0\,)$, então o nível modal de φ' em φ na posição λ é dado por $\mathsf{mlevel}(\varphi, \varphi', \lambda) = ml$ e sua polaridade é dada por $\mathsf{pol}(\varphi, \varphi', \lambda) = pol$.

O seguinte exemplo apresenta a árvore anotada de $[][]\,(p \vee\ (\neg[]p \wedge \neg[]\neg q\,)\,)$, ilustrando os conceitos acima.

Exemplo 5 A árvore sintática anotada da fórmula $[][]\,(p \vee\ (\neg[]p \wedge \neg[]\neg q\,)\,)$ é:



Como $\{[]\neg q, 1.1.2.2.1, -\} \in \tau\,([][]\,(p \vee (\neg[]p \wedge \neg[]\neg q\,)\,), , 0, +\,)$, então a subfórmula $[]\neg q$ na posição 1.1.2.2.1 da superfórmula $[][]\,(p \vee (\neg[]p \wedge \neg[]\neg q\,)\,)$ ocorre no nível modal 2 e tem polaridade negativa. Note que p na posição 1.1.1 ocorre com polaridade positiva no nível modal 2 e na posição 1.1.2.1.1.1 com polaridade negativa no nível modal 3.

Se $\mathsf{pol}(\varphi, \varphi', \lambda) = +$, dizemos que a subfórmula φ' na posição λ da superfórmula φ tem

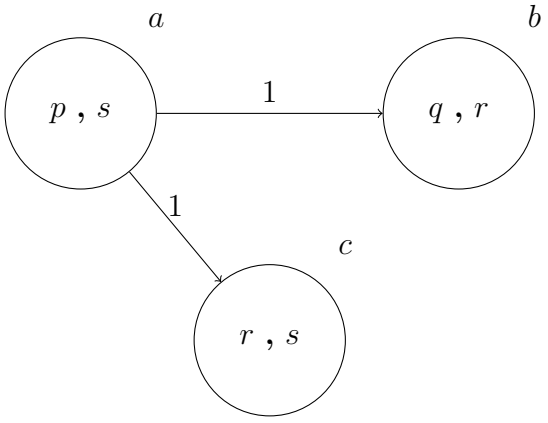
polaridade positiva. Se para todas as posições λ do nível modal ml, temos $\mathsf{pol}(\varphi, \varphi', \lambda) = +$ ou $\mathsf{pol}(\varphi, \varphi', \lambda) = -$, dizemos que φ' é pura no nível modal ml. Finalmente, se φ' é pura em todos os níveis modais, então dizemos que φ' é pura. No Exemplo 5 acima, a subfórmula p é pura no nível modal 2, com polaridade positiva; é também pura no nível modal 3, com polaridade negativa; mas não é pura em toda a fórmula. Já a subfórmula q ocorre tão somente com polaridade positiva (no nível modal 3), portanto é pura em toda a fórmula. Se um literal l é puro em todos os níveis modais, dizemos que l é um literal puro.

2.2 Semântica

Como dito no início do capítulo, a Lógica Modal permite representar fatos em diferentes contextos de acordo com a visão de vários agentes. Para interpretar essas fórmulas usamos o conceito de mundos e relações binárias de acessibilidade entre esses mundos. Cada mundo é um contexto, logo uma fórmula φ pode ser verdadeira em um mundo e falsa em outro mundo.

No exemplo a seguir, os mundos são representados pelos círculos. Dentro de cada mundo temos os símbolos proposicionais considerados verdadeiros naquele contexto. Os símbolos que não constam do círculo são considerados falsos. As arestas são as relações de acessibilidade R e o rótulo acima delas indica o agente a, ou seja, representam a relação entre elementos em R_a.

Exemplo 6



O mundo a está relacionado com b e c através de arestas rotuladas por 1, ou seja, representa aR_1b e aR_1c. Como para todos os mundos acessíveis a partir de a, temos que r é verdade, então $[1]r$ é verdadeiro em a. Além disso, $\langle 1 \rangle s$ é verdadeiro em a pois em um, ou mais mundos, acessíveis a partir de a, temos que s é verdade.

O exemplo apresenta uma forma intuitiva de entender a semântica da Lógica Multimodal K_n, para determinar a satisfatibilidade das fórmulas modais usamos *modelos de Kripke*, cuja definição formal é dada a seguir.

Definição 7 Um *modelo de Kripke* $\mathcal{M}$ para n agentes sobre $\mathcal{P}$ é dada pela tupla $\langle W, w_0, R_1, ..., R_n, \pi \rangle$, onde:

- W é um conjunto não vazio
- w_0 é um elemento distinto em W
- R_a é uma relação binária sobre W, denominada *relação de acessibilidade*, tal que $R_a \subseteq W \times W$, para todo $a \in \mathcal{A}$
- $\pi : W \times \mathcal{P} \rightarrow \{false, true\}$ é uma função que associa para cada mundo $w \in W$ um valor de verdade para os símbolos proposicionais

Denotamos por wR_av que v é acessível a partir de w, ou seja, $(w, v) \in R_a$. A satisfatibilidade de uma fórmula em um mundo é definida abaixo.

Definição 8 A satisfatibilidade de uma fórmula em mundo w de um modelo $\mathcal{M}$ é definida indutivamente, como segue:

- $(\mathcal{M}, w) \vDash p$ se, e somente se, $\pi(w)(p) = true$, onde $p \in \mathcal{P}$
- $(\mathcal{M}, w) \vDash \neg\varphi$ se, e somente se, $(\mathcal{M}, w) \nvDash \varphi$
- $(\mathcal{M}, w) \vDash (\varphi \wedge \psi)$ se, e somente se, $(\mathcal{M}, w) \vDash \varphi$ e $(\mathcal{M}, w) \vDash \psi$
- $(\mathcal{M}, w) \vDash (\varphi \vee \psi)$ se, e somente se, $(\mathcal{M}, w) \vDash \varphi$ ou $(\mathcal{M}, w) \vDash \psi$
- $(\mathcal{M}, w) \vDash [a]\varphi$ se, e somente se, para todo w', wR_aw' implica $(\mathcal{M}, w') \vDash \varphi$
- $(\mathcal{M}, w) \vDash \langle a \rangle\varphi$ se, e somente se, existe w' tal que wR_aw' e $(\mathcal{M}, w') \vDash \varphi$

Esse trabalho tem como foco o problema da satisfatibilidade local e global. Seja $\mathcal{M} = (W, w_0, R_1, ..., R_n, \pi)$ um modelo:

- Uma fórmula φ é satisfeita localmente se existe um modelo e a fórmula é satisfeita em w_0, ou seja, se $(M, w_0) \models \varphi$. Nesse caso, diz-se que φ é localmente satisfeita em $\mathcal{M}$ e denotado da seguinte forma: $\mathcal{M} \vDash_L \varphi$.

- Uma fórmula φ é satisfeita globalmente se existe um modelo tal que φ é satisfeita em todos os seus mundos. Ou seja, φ é globalmente satisfeita em $\mathcal{M}$, denotado por $\mathcal{M} \vDash_G \varphi$, se para todo $w \in W$, $(\mathcal{M}, w) \vDash \varphi$.
- Dado um conjunto de fórmulas $\Gamma = \{\gamma_1, ..., \gamma_m\}$, $m \in \mathbb{N}$, uma fórmula φ é satisfeita localmente com restrições globais Γ se existe um modelo que satisfaz globalmente todas as fórmulas em Γ e w_0 satisfaz φ. Ou seja, se existe um modelo $\mathcal{M}$ tal que $\mathcal{M} \vDash_G \bigwedge_{i=1}^{m} \gamma_i$ e $\mathcal{M} \vDash_L \varphi$.

Definição 9 Um modelo $\mathcal{M} = (W, w_0, R_1, ..., R_n, \pi)$ está no formato de árvore se $\bigcup_{a=1}^{n} R_a$ é uma árvore, isto é um grafo acíclico direcionado (com raiz w_0).

Definição 10 Sejam $\varphi, \psi \in \mathsf{WFF}_{\mathsf{K}_n}$ fórmulas bem-formadas. A profundidade modal de uma fórmula é dada pela função mdepth: $\mathsf{WFF}_{\mathsf{K}_n} \to \mathbb{N}$, onde:

- $\mathsf{mdepth}(p) = \mathbf{true} = \mathbf{false} = 0$, para $p \in \mathcal{P}$;
- $\mathsf{mdepth}(\neg\varphi) = \mathsf{mdepth}(\varphi)$;
- $\mathsf{mdepth}(\varphi \wedge \psi) = \mathsf{max}(\mathsf{mdepth}(\varphi), \mathsf{mdepth}(\psi))$;
- $\mathsf{mdepth}(\varphi \vee \psi) = \mathsf{max}(\mathsf{mdepth}(\varphi), \mathsf{mdepth}(\psi))$;
- $\mathsf{mdepth}([a]\varphi) = 1+\mathsf{mdepth}(\varphi)$;
- $\mathsf{mdepth}(\langle a\rangle\varphi) = 1+\mathsf{mdepth}(\varphi)$.

sendo max: $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ a função que determina o máximo entre dois números naturais.

Iremos denotar a profundidade de um mundo em um modelo em formato de árvore $\mathcal{M}$ por $\mathsf{mdepth}(w)$, que corresponde ao comprimento do caminho único entre w_0 e w através da união das relações de acessibilidade em $\mathcal{M}$.

Definição 11 *Camada modal* é a classe de equivalência de mundos de uma mesma profundidade em um modelo.

Com a ajuda destas definições, prova-se que satisfatibilidade local depende somente do nível modal em que subfórmulas ocorrem [10]. Ou seja, pode-se intercambiar os termos nível modal e camada modal.

Também podemos trocar a verificação da satisfatibilidade global de φ pela verificação da satisfatibilidade local de φ em todas as camadas modais de um modelo [8].

Definição 12 Seja K_n^* a extensão de K_n com um operador adicional $[*]$, o operador universal. Seja $\mathcal{M} = (W, w_0, R_1, ..., R_n, \pi)$ um modelo para K_n. O modelo $\mathcal{M}^*$ é a tupla $(W, w_0, R_1, ..., R_n, R_*, \pi)$, onde $R_* = W \times W$. A fórmula $[*]\varphi$ é localmente satisfeita no mundo w_0 no modelo $\mathcal{M}^*$, denotado por $(\mathcal{M}^*, w_0) \vDash_L [*]\varphi$, se, e somente se, para todo $w' \in W$, temos que $(\mathcal{M}^*, w') \vDash \varphi$.

A prova do seguinte teorema é dada em [11].

Teorema 1 Para $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$, decidir se $\mathcal{M} \vDash_G \varphi$ é equivalente a decidir se $\mathcal{M}^* \vDash_L [*]\varphi$.

Segue que decidir se a fórmula φ é satisfatível sob restrições globais $\Gamma = \{\gamma_1, ..., \gamma_m\}, m \in \mathbb{N}$, é equivalente a decidir se $\mathcal{M}^* \vDash_L \varphi \wedge [*](\gamma_1 \wedge ... \wedge \gamma_m)$.

O KSP usa essas abordagens baseadas em níveis modais para lidar com o problema da satisfatibilidade global e local. No próximo capítulo iremos apresentar o cálculo implementado no KSP.

Capítulo 3

Resolução

O cálculo utilizado no $K_{\mathrm{S}}P$ é clausal logo, antes de uma fórmula ser testada para satisfatibilidade global e local, ela é traduzida para uma forma normal denominada *Forma Normal Separada com Níveis Modais*, SNF_{ml}. Uma fórmula em SNF_{ml} é uma conjunção de cláusulas rotuladas pelos níveis modais nos quais elas ocorrem. Denotamos por $ml : \varphi$ a fórmula φ que se encontra no nível modal $ml \in \mathbb{N} \cup \{*\}$, onde $* : \varphi$ significa que φ ocorre em todos os níveis modais. Formalmente, $\mathsf{WFF}^{ml}_{\mathsf{K}_n}$ é o conjunto de fórmulas $ml : \varphi$ tal que $ml \in \mathbb{N} \cup \{*\}$ e $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$. A interpretação de fórmulas nesta nova linguagem é dada a seguir.

Definição 13 Seja $\mathcal{M}^* = (W, w_0, R_1, ..., R_n, R_*, \pi)$ um modelo e $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$. Satisfatibilidade de uma fórmula rotulada é dada por:

- $\mathcal{M}^* \vDash ml : \varphi$ se, e somente se, para todos os mundos $w \in W$ tal que $\mathsf{mdepth}(w) = ml$, temos que $\langle \mathcal{M}^*, w \rangle \vDash \varphi$;
- $\mathcal{M}^* \vDash * : \varphi$ se, e somente se, $\mathcal{M}^* \vDash [*]\varphi$.

3.1 Forma Normal Separada com Níveis Modais

Seja $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$. Uma fórmula em SNF_{ml} é uma conjunção de cláusulas rotuladas pelos níveis modais nos quais elas ocorrem, isto é, cláusulas em SNF_{ml} são da forma:

- *Cláusula literal*: $ml : \bigvee_{b=1}^{r} l_b$
- *a-cláusula positiva*: $ml : l' \Rightarrow [a]l$
- *a-cláusula negativa*: $ml : l' \Rightarrow \langle a \rangle l$

onde $ml \in \mathbb{N} \cup \{*\}$ e $l, l', l_b \in L$. A cláusula $ml : D$, onde D é uma disjunção vazia, é chamada *cláusula vazia* e denotada por ml : **false**.

A tradução de φ em SNF_{ml} tem duas etapas, sendo a primeira a transformação na *Forma Normal Negada* (NNF) e a segunda a tradução da fórmula em NNF para a *Forma Normal Separada com Níveis Modais*.

Uma fórmula está na *Forma Normal Negada* se contém apenas os conectivos $\neg$, $\wedge$, $\vee$, $\langle a \rangle$ e $[a]$, com $a \in \mathcal{A}$; e somente há variáveis proposicionais no escopo do operador $\neg$. Para transformar φ em NNF, aplica-se as regras de reescrita da Tabela 3.1 [1].

Tabela 3.1: Regras de reescrita para obtenção da NNF

$\psi \leftrightarrow \gamma \Rightarrow \ (\,(\psi \rightarrow \gamma\,) \wedge (\gamma \rightarrow \psi\,)\,)$
$\psi \rightarrow \gamma \Rightarrow \neg\psi \vee \gamma$
$\neg\,(\psi \wedge \gamma\,) \Rightarrow \ (\neg\psi \vee \neg\gamma\,)$
$\neg\,(\psi \vee \gamma\,) \Rightarrow \ (\neg\psi \wedge \neg\gamma\,)$
$\neg\neg\psi \Rightarrow \psi$
$\neg[a]\psi \Rightarrow \langle a\rangle\neg\psi$
$\neg\langle a\rangle\psi \Rightarrow [a]\neg\psi$

As regras de simplificação presentes na Tabela 3.2 podem ser utilizadas em qualquer momento durante o processo de tradução [8, 1].

A seguir apresentamos um exemplo da transformação de uma fórmula em sua forma normal negada.

Exemplo 7 Usando as regras das Tabelas 3.1 e 3.2, o processo de reescrita da fórmula $[1][1]\,(\neg p \rightarrow \neg\,(\neg[1]p \vee \langle 1\rangle q\,)\,)$ para a forma normal negada é:

$[1][1]\,(\neg p \rightarrow \neg\,(\neg[1]p \vee \langle 1\rangle q\,)\,)$
$\Rightarrow [1][1]\,(\neg\neg p \vee \neg\,(\neg[1]p \vee \langle 1\rangle q\,)\,)$
$\Rightarrow [1][1]\,(p \vee \neg\,(\neg[1]p \vee \langle 1\rangle q\,)\,)$
$\Rightarrow [1][1]\,(p \vee \ (\neg\neg[1]p \wedge \neg\langle 1\rangle q\,)\,)$
$\Rightarrow [1][1]\,(p \vee \ ([1]p \wedge \neg\langle 1\rangle q\,)\,)$
$\Rightarrow [1][1]\,(p \vee \ ([1]p \wedge [1]\neg q\,)\,)$

Além das regras de simplificação da Tabela 3.2, as regras de simplificação da Tabela 3.3 também podem ser usadas em cláusulas rotuladas.

Tabela 3.2: Regras de simplificação

$\psi \vee \psi \Rightarrow \psi$
$\psi \wedge \psi \Rightarrow \psi$
$\psi \vee \neg\psi \Rightarrow \mathbf{true}$
$\psi \wedge \neg\psi \Rightarrow \mathbf{false}$
$\psi \wedge \mathbf{true} \Rightarrow \psi$
$\psi \wedge \mathbf{false} \Rightarrow \mathbf{false}$
$\psi \vee \mathbf{true} \Rightarrow \mathbf{true}$
$\psi \vee \mathbf{false} \Rightarrow \psi$
$[a]\mathbf{true} \Rightarrow \mathbf{true}$
$\langle a\rangle\mathbf{false} \Rightarrow \mathbf{false}$
$\neg\mathbf{true} \Rightarrow \mathbf{false}$
$\neg\mathbf{false} \Rightarrow \mathbf{true}$
$([a]\varphi \wedge \langle a\rangle\neg\varphi\,) \Rightarrow \mathbf{false}$
$([a]\varphi \wedge [a]\neg\varphi\,) \Rightarrow [a]\mathbf{false}$
$([a]\mathbf{false} \wedge\langle a\rangle\varphi\,) \Rightarrow \mathbf{false}$
$([a]\mathbf{false} \wedge[a]\varphi\,) \Rightarrow [a]\mathbf{false}$
$(\langle a\rangle\varphi \vee \langle a\rangle\neg\varphi\,) \Rightarrow \langle a\rangle\mathbf{true}$
$(\langle a\rangle\mathbf{true} \vee[a]\varphi\,) \Rightarrow \mathbf{true}$
$(\langle a\rangle\mathbf{true} \vee\langle a\rangle\varphi\,) \Rightarrow \langle a\rangle\mathbf{true}$

A tradução da fórmula em NNF para SNF_{ml} é feita através da aplicação recursiva de reescrita e renomeamento [12]. Sejam φ e $\varphi' \in \mathsf{WFF}_{\mathsf{K}_n}$, t' um novo símbolo proposicional, e $* + 1 = *$, a função de tradução $\rho : \mathsf{WFF}^{ml}_{\mathsf{K}_n} \rightarrow \mathsf{WFF}^{ml}_{\mathsf{K}_n}$ é definida na Tabela 3.4.

Para satisfatibilidade local, a tradução de φ é dada por $(0 : t\,) \wedge \rho\,(0 : t \rightarrow \varphi\,)$ e para satisfatibilidade global, a tradução de φ é dada por $(* : t) \wedge \rho\,(* : t \rightarrow \varphi\,)$, sendo que t é um novo símbolo proposicional. Para satisfatibilidade local de φ com restrições globais $\Gamma = \gamma_1, ..., \gamma_m$, a tradução é dada por $(* : t) \wedge \rho(* : t \rightarrow \bigwedge_{i=1}^{m} \gamma_i\,) \wedge \rho\,(0 : t \rightarrow \varphi\,)$, sendo que t é um novo símbolo proposicional. Como o operador de conjunção é comutativo, associativo, e idempotente, uma fórmula em SNF_{ml} pode ser referenciada

Tabela 3.3: Regras de simplificação para cláusulas rotuladas, onde $ml \in \mathbb{N} \cup \{*\}$, l é um literal e D é uma cláusula proposicional (possivelmente vazia)

$ml : D \vee l \vee l \Rightarrow ml : D \vee l$
$ml : D \vee \mathbf{false} \Rightarrow ml : D$
$ml : D \vee l \vee \neg l \Rightarrow ml : D \vee \mathbf{true} \Rightarrow ml : \mathbf{true}$
$ml : \mathbf{true} \wedge D \Rightarrow ml : D$

$$
\begin{array}{rcl}
\rho\,(ml : t \to \varphi \wedge \varphi'\,) & = & \rho\,(ml : t \to \varphi\,) \wedge \rho\,(ml : t \to \varphi'\,), \\
\rho\,(ml : t \to [a]\varphi\,) & = & \rho\,(ml : t \to [a]\varphi\,), \text{ se } \varphi \text{ é um literal} \\
 & = & (ml : t \to [a]t'\,) \wedge \rho\,(ml + 1 : t' \to \varphi\,), \text{ caso contrário,} \\
\rho\,(ml : t \to \langle a\rangle\varphi\,) & = & \rho\,(ml : t \to \langle a\rangle\varphi\,), \text{ se } \varphi \text{ é um literal} \\
 & = & (ml : t \to \langle a\rangle t'\,) \wedge \rho\,(ml + 1 : t' \to \varphi\,), \text{ caso contrário,} \\
\rho\,(ml : t \to \varphi \vee \varphi'\,) & = & (ml : \neg t \vee \varphi \vee \varphi'\,), \text{ se } \varphi \text{ e } \varphi' \text{ são disjunções de literais ou constantes,} \\
 & & \text{sendo que } \varphi \text{ é possivelmente vazia,} \\
 & = & \rho\,(ml : t \to \varphi \vee t'\,) \wedge \rho\,(ml : t' \to \varphi'\,), \\
 & & \text{se } \varphi' \text{ não é uma disjunção de literais ou constantes.}
\end{array}
$$

Tabela 3.4: Função de tradução para Forma Normal Separada

como um conjunto de cláusulas.

Exemplo 8 Usando as regras da Tabela 3.4, o processo de tradução, para satisfatibilidade global, da fórmula $[1][1]\,(p \vee\ ([1]p \wedge [1]\neg q\,)\,)$, obtida no processo de tradução do Exemplo 7, para a SNF_{ml} seria:

$(* : t_0\,) \wedge \rho\,(* : t_0 \to [1][1]\,(p \vee\ ([1]p \wedge [1]\neg q\,)\,)\,)$

$\Rightarrow\ (* : t_0\,) \wedge\ (* : t_0 \to [1]t_1\,) \wedge \rho(* : t_1 \to [1]\,(p \vee\ ([1]p \wedge [1]\neg q\,)\,)\,)$

$\Rightarrow\ (* : t_0\,) \wedge\ (* : t_0 \to [1]t_1\,) \wedge\ (* : t_1 \to [1]t_2\,) \wedge \rho(* : t_2 \to\ (p \vee\ ([1]p \wedge [1]\neg q\,)\,)\,)$

$\Rightarrow\ (* : t_0\,) \wedge\ (* : t_0 \to [1]t_1\,) \wedge\ (* : t_1 \to [1]t_2\,) \wedge \rho(* : t_2 \to p \vee t_3\,) \wedge \rho\,(* : t_3 \to ([1]p \wedge [1]\neg q\,)\,)$

$\Rightarrow\ (* : t_0\,) \wedge\ (* : t_0 \to [1]t_1\,) \wedge\ (* : t_1 \to [1]t_2\,) \wedge\ (* : \neg t_2 \vee p \vee t_3\,) \wedge \rho\,(* : t_3 \to [1]p\,) \wedge \rho\,(* : t_3 \to [1]\neg q\,)$

$\Rightarrow\ (* : t_0\,) \wedge\ (* : t_0 \to [1]t_1\,) \wedge\ (* : t_1 \to [1]t_2\,) \wedge\ (* : \neg t_2 \vee p \vee t_3\,) \wedge\ (* : t_3 \to [1]p\,) \wedge\ (* : t_3 \to [1]\neg q\,)$

Logo, a fórmula traduzida em SNF_{ml} é:

$(* : t_0) \wedge (* : t_0 \rightarrow [1]t_1) \wedge (* : t_1 \rightarrow [1]t_2) \wedge (* : \neg t_2 \vee p \vee t_3) \wedge (* : t_3 \rightarrow [1]p) \wedge (* : t_3 \rightarrow [1]\neg q)$

Além de cláusulas em SNF_{ml}, o $K\mathcal{S}P$ também utiliza cláusulas em SNF^{+}_{ml} e em SNF^{++}_{ml} para resolução negativa e ordenada, respectivamente.

Para a completude da resolução negativa, os literais, que ocorrem no escopo dos operadores modais, devem ser positivos [13]. A reescrita das cláusulas para a forma SNF^{+}_{ml} garante que isso aconteça.

A tradução para a forma SNF^{+}_{ml} é feita com a aplicação sucessiva das regras de reescrita da Tabela 3.5 (onde $ml \in \mathbb{N}$, $t, p \in \mathcal{P}$, e t' é símbolo proposicional novo) uma vez que a fórmula já esteja em SNF_{ml} .

$$\begin{aligned} \rho(ml : t \rightarrow [a]\neg p) &= (ml : t \rightarrow [a]t') \wedge \rho(ml + 1 : t' \rightarrow \neg p), \\ \rho(ml : t \rightarrow \langle a\rangle\neg p) &= (ml : t \rightarrow \langle a\rangle t') \wedge \rho(ml + 1 : t' \rightarrow \neg p) \end{aligned}$$

Tabela 3.5: Função de tradução para SNF^{+}_{ml}

A tradução para a forma SNF^{++}_{ml} é feita com a aplicação sucessiva das regras de reescrita da Tabela 3.6 (onde $ml \in \mathbb{N}$, $t, p \in \mathcal{P}$, e t' é símbolo proposicional novo) uma vez que a fórmula já esteja em SNF_{ml}.

$$\begin{aligned} \rho(ml : t \rightarrow [a]l) &= (ml : t \rightarrow [a]t') \wedge \rho(ml + 1 : t' \rightarrow l), \\ \rho(ml : t \rightarrow \langle a\rangle l) &= (ml : t \rightarrow \langle a\rangle t') \wedge \rho(ml + 1 : t' \rightarrow l) \end{aligned}$$

Tabela 3.6: Função de tradução para SNF^{++}_{ml}

As seguintes definições são necessárias para o entendimento de resolução ordenada.

Definição 14 Seja Φ um conjunto de cláusulas e P_Φ o conjunto de símbolos proposicionais que ocorrem em Φ. Seja $\succ$ uma ordenação total e bem fundada em Φ. Essa ordenação pode ser estendida para literais L_Φ sobre P_Φ definindo $\neg p \succ p$ e $p \succ \neg q$ sempre que $p \succ q$, para todo $p, q \in P_\Phi$.

Definição 15 Um literal l é considerado *maximal* com respeito a uma cláusula $ml : C \vee l$ se, e somente se, não existe l' em C tal que $l' \succ l$.

Seja Φ um conjunto de cláusulas em SNF_{ml} e uma ordenação dos literais ocorrendo em Φ, a aplicação exaustiva das regras na Tabela 3.6, onde $p \succ t'$, para todo p que

ocorre em Φ, traduz o conjunto de cláusulas para a forma SNF^{++}_{ml}. Isso garante que os literais, que ocorrem no escopo dos operadores modais, sejam "pequenos o suficientes" com respeito a uma dada ordenação original dos literais, requisito para garantir a completude da resolução ordenada [1].

Os teoremas abaixo garantem que a transformação na forma clausal preserva satisfatibilidade. As provas podem ser encontradas em [1].

Teorema 2 Seja $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ uma fórmula e $t \in \mathcal{P}$ um símbolo proposicional que não ocorre em φ. Então φ é localmente satisfeita se, e somente se, $(0 : t) \wedge \rho(0 : t \rightarrow \varphi)$ é satisfatível.

Teorema 3 Seja $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ uma fórmula e $t \in \mathcal{P}$ um símbolo proposicional que não ocorre em φ. Então φ é globalmente satisfeita se, e somente se, $(* : t) \wedge \rho(* : t \rightarrow \varphi)$ é satisfatível.

Teorema 4 Seja $\varphi \in \mathsf{WFF}_{\mathsf{K}_n}$ uma fórmula, $\Gamma \subseteq \mathsf{WFF}_{\mathsf{K}_n}$ um conjunto de fórmulas, e $t \in \mathcal{P}$ um símbolo proposicional que não ocorre em φ ou em Γ. Então φ é localmente satisfeita com restrições globais Γ se, e somente se, $(* : t) \wedge \rho(* : t \rightarrow \Lambda_{\gamma \in \Gamma} \gamma) \wedge \rho(0 : t \rightarrow \varphi)$ é satisfatível.

3.2 Regras de Inferência

Na Tabela 3.7 estão as regras de inferência do cálculo baseado em resolução para K_n, denotado por RES_{ml}. Na tabela, as premissas são apresentadas acima da linha e a conclusão, chamada de resolvente, é apresentada abaixo da linha. As regras só podem ser aplicadas se a unificação entre os rótulos, como dada na Definição 16, não for indefinida.

Definição 16 Unificação é dada pela função parcial $\sigma : 2^{\mathbb{N} \cup \{*\}} \rightarrow \mathbb{N} \cup \{*\}$, onde: $\sigma(\{ml, *\}) = ml$ e $\sigma(\{ml\}) = ml$, nos outros casos σ é indefinido.

Definição 17 Seja Φ um conjunto de cláusulas em SNF_{ml}. Uma *derivação* a partir de Φ é uma sequência de conjuntos $\Phi_0, \Phi_1, ...$, onde $\Phi_0 = \Phi$, e para cada $i > 0, \Phi_{i+1} = \Phi_i \cup \{D\}$, onde $D \notin \Phi_i$ é o resolvente obtido de Φ_i através da aplicação de LRES, MRES, GEN1, GEN2 ou GEN3. É necessário que D esteja na forma simplificada e não seja uma tautologia (ou seja, equivalente a **true**).

[LRES]

$$\frac{\begin{array}{rrcl} ml_1: & D & \vee & l \\ ml_2: & D' & \vee & \neg l \end{array}}{\begin{array}{rrcl} ml: & D & \vee & D' \end{array}}$$

[MRES]

$$\frac{\begin{array}{rrcl} ml_1: & l_1 & \rightarrow & [a]l \\ ml_2: & l_2 & \rightarrow & \langle a\rangle\neg l \end{array}}{\begin{array}{rrcl} ml: & \neg l_1 & \vee & \neg l_2 \end{array}}$$

[GEN2]

$$\frac{\begin{array}{rrcl} ml_1: & l'_1 & \rightarrow & [a]l_1 \\ ml_2: & l'_2 & \rightarrow & [a]\neg l_1 \\ ml_3: & l'_3 & \rightarrow & \langle a\rangle l_2 \end{array}}{\begin{array}{rl} ml: & \neg l'_1 \vee \neg l'_2 \vee \neg l'_3 \end{array}}$$

[GEN1]

$$\frac{\begin{array}{rrcl} ml_1: & l'_1 & \rightarrow & [a]\neg l_1 \\ & \vdots & & \\ ml_m: & l'_m & \rightarrow & [a]\neg l_m \\ ml_{m+1}: & l' & \rightarrow & \langle a\rangle\neg l \\ ml_{m+2}: & l_1 & \vee & \ldots \vee l_m \vee l \end{array}}{\begin{array}{rl} ml: & \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l' \end{array}}$$

[GEN3]

$$\frac{\begin{array}{rrcl} ml_1: & l'_1 & \rightarrow & [a]\neg l_1 \\ & \vdots & & \\ ml_m: & l'_m & \rightarrow & [a]\neg l_m \\ ml_{m+1}: & l' & \rightarrow & \langle a\rangle l \\ ml_{m+2}: & l_1 & \vee & \ldots \vee l_m \end{array}}{\begin{array}{rl} ml: & \neg l'_1 \vee \ldots \vee \neg l'_m \vee \neg l' \end{array}}$$

Tabela 3.7: Regras de Inferência, onde $ml = \sigma(\{ml_1, ..., ml, ml_{m+1}, ml_{m+2} - 1\})$ em GEN1, GEN3, onde $m \geq 0$; $ml = \sigma(\{ml_1, ml_2\})$ em LRES, MRES; e $ml = \sigma(\{ml_1, ml_2, ml_3\})$ em GEN2.

Definição 18 Um conjunto Φ de cláusulas em SNF_{ml} é *saturado* se toda cláusula que é um resolvente obtido de Φ através da aplicação de LRES, MRES, GEN1, GEN2 ou GEN3 é uma tautologia ou já está contida em Φ.

Definição 19 Uma *refutação local* para um conjunto de cláusulas Φ é uma derivação $\Phi_0, ..., \Phi_k, k \in \mathbb{N}$, onde $0 : \mathbf{false} \in \Phi_k$. Uma *refutação global* para Φ é uma derivação $\Phi_0, ..., \Phi_k, k \in \mathbb{N}$, onde $* : \mathbf{false} \in \Phi_k$.

Definição 20 Uma derivação $\Phi_0, ..., \Phi_k$ de Φ é *terminante* se é uma refutação local ou global para Φ ou se existe um $\Phi_i, i \in \mathbb{N}$, tal que Φ_i está saturado.

Apresentamos a seguir alguns exemplos da aplicação do cálculo a conjuntos de cláusulas.

Exemplo 9 Um exemplo possível da aplicação das regras de inferência presentes na Tabela 3.7 no seguinte conjunto de fórmulas em SNF_{ml} seria:

1. $0 : \mathbf{true} \rightarrow t_0$

2. $0 : t_0 \rightarrow \langle 1 \rangle \neg q$

3. $* : \mathbf{true} \rightarrow t_2$

4. $* : t_2 \rightarrow [1]q$

5. $0 : \mathbf{true} \rightarrow \neg t_2 \vee \neg t_0$ [MRES, 2, 4, $\neg q$]

6. $0 : \mathbf{true} \rightarrow \neg t_2$ [LRES, 5, 1, $\neg t_0$]

7. $0 : \mathbf{true} \rightarrow \mathbf{false}$ [LRES, 6, 3, $\neg t_2$]

Note que ao lado direito do operador $\rightarrow$ tem-se $\langle a \rangle \neg q$ na cláusula da Linha 2 e $[a]q$ na cláusula da Linha 4, e $\sigma(\{0, *\}) = 0$. Por esse motivo, pode-se aplicar MRES nessas cláusulas gerando a cláusula da Linha 5 como resolvente. No caso das cláusulas das Linhas 5 e 1 tem-se, respectivamente, t_0 e $\neg t_0$, e como $\sigma(\{0, 0\}) = 0$, LRES foi aplicada gerando como resolvente a cláusula da Linha 6. Nessa nova cláusula e na cláusula da Linha 3 aplicou-se novamente LRES visto que elas apresentam, ao lado direito do operador $\rightarrow$, os literais $\neg t_1$ e t_2, e $\sigma(\{*, 0\}) = 0$. Como resolvente a cláusula vazia, na Linha 7 foi gerada. Portanto, a derivação a partir do conjunto de cláusulas dado entre as Linhas 1 a 7, resulta no conjunto dado acima, ou seja, é uma *refutação local*.

Exemplo 10 A seguir temos um exemplo da aplicação das regras de inferência presentes na Tabela 3.7 em um conjunto de fórmulas em SNF_{ml} que gera uma derivação que não é uma *refutação global* mas é *terminante*:

1. $* : \mathbf{true} \rightarrow t_0$

2. $* : t_1 \rightarrow \langle 1 \rangle \neg p$

3. $* : \mathbf{true} \rightarrow t_1 \vee \neg t_0 \vee q$

4. $* : \mathbf{true} \rightarrow t_2 \vee \neg t_0 \vee \neg r$

5. $* : \mathbf{true} \rightarrow t_3$

6. $* : \mathbf{true} \rightarrow \neg t_3 \vee p$

7. $* : \mathbf{true} \rightarrow q \vee t_1$ [LRES, 1, 3, $\neg t_0$]

8. $* : \mathbf{true} \rightarrow \neg r \vee t_2$ [LRES, 1, 4, $\neg t_0$]

9. $* : \mathbf{true} \rightarrow p$ [LRES, 5, 6, $\neg t_3$]

10. $* : \mathbf{true} \rightarrow \neg t_1$ [GEN1, 2, 9, $\neg p$]

Note que a cláusula $* : false$ não foi gerada mas não é possível produzir uma nova derivação.

Exemplo 11 A seguir temos um exemplo da aplicação das regras de inferência presentes na Tabela 3.7 em um conjunto de fórmulas em SNF_{ml} que gera uma derivação que não é uma *refutação local* mas é *terminante*:

1. $0 : t_0 \rightarrow \langle 1 \rangle q$
2. $0 : t_3 \rightarrow [1] r$
3. $0 : t_1 \rightarrow [1] q$
4. $0 : t_2 \rightarrow [1] p$
5. $1 : \mathbf{true} \rightarrow \neg p \vee \neg q \vee \neg r$
6. $0 : \mathbf{true} \rightarrow \neg t_3 \vee \neg t_2 \vee \neg t_1 \vee \neg t_0$ [GEN3, 1, 2, 3, 4, 5, q, p, r, q]
7. $0 : \mathbf{true} \rightarrow \neg t_3 \vee \neg t_2 \vee \neg t_1 \vee \neg t_0$ [GEN1, 1, 2, 4, 5, q, p, r, q]

Note que a cláusula $0 : false$ não foi gerada, mas não é possível produzir uma nova derivação.

Os teoremas a seguir garantem que o cálculo apresentando é correto e completo. As provas podem ser encontradas em [1].

Teorema 5 Seja Φ um conjunto de cláusulas em SNF_{ml} e $\Phi_0, ..., \Phi_k, k \in \mathbb{N}$, uma derivação a partir de Φ. Se Φ é satisfatível, então todo $\Phi_i, 0 \leq i \leq k$, é satisfatível.

Teorema 6 Seja Φ um conjunto de cláusulas insatisfatível em SNF_{ml}. Então existe uma refutação para Φ.

Como dito na seção anterior, o KSP também aplica resolução negativa e ordenada. Como resolução negativa é um caso especial de resolução semântica [1], a regra da resolução semântica a seguir é fundamental para entender resolução negativa.

Para o caso clássico, dada uma interpretação π, resolução pode ser aplicada somente se uma das cláusulas da premissa é um *núcleo*, isto é, uma cláusula com valor verdade igual

$$\text{[S-RES]}\quad \frac{D \vee l \qquad D' \vee \neg l,}{D \vee D'}$$

Tabela 3.8: Regra da Resolução Semântica, onde l é um literal, e D e D' são disjunções de literais

a *true* interpretada sobre π. Tomando $\pi(p) = false$, para todo símbolo proposicional p, resolução semântica corresponde a resolução negativa. Em outras palavras, a regra só pode ser aplicada se uma das cláusulas que estão sendo resolvidas é negativa.

Definição 21 Um literal é dito negativo se é a negação de um símbolo proposicional. Uma cláusula é considerada negativa se contém somente literais negativos.

No provador KSP, resolução negativa é usada como um método de refinamento que restringe a seleção de cláusulas. Nesse caso, uma cláusula literal $ml : D$ é considerada negativa se, e somente se, D é uma cláusula negativa. Ou seja, restringir o cálculo modal para resolução negativa significa que pelo menos uma das cláusulas das premissas das regras de inferência deve ser uma cláusula literal negativa. O cálculo de resolução que resulta da restrição de RES_{ml} pela resolução negativa é denotado por RES^{neg}_{ml}.

O teorema a seguir garante que resolução negativa é um cálculo completo. Assim como nos teoremas anteriores, a prova pode ser encontrada em [1].

Teorema 7 Seja Φ um conjunto de cláusulas em SNF^{+}_{ml}. Se Φ é insatisfatível, então existe uma refutação por RES^{neg}_{ml} para Φ.

Resolução ordenada também é um refinamento de resolução onde a aplicação das regras de inferências são restringidas aos literais maximal em uma cláusula, com respeito a uma ordenação bem fundada dos literais. No caso clássico de resolução binária, o refinamento ordenado restringe a aplicação para cláusulas $C \vee l$ e $D \vee \neg l$ onde l é maximal com respeito à C e $\neg l$ é maximal com respeito à D.

O teorema a seguir garante que resolução ordenada também é um cálculo completo. A prova pode ser encontrada em [1].

Teorema 8 Seja Φ um conjunto de cláusulas em SNF_{ml}^{++}. Se Φ é insatisfatível, então existe uma refutação por resolução ordenada para Φ.

Capítulo 4

KSP

Como dito anteriormente, o objetivo desse trabalho é refatorar a implementação do KSP para que o provador possa lidar também com a combinação dos raciocínios local e global. Nesse capítulo apresentaremos a arquitetura do provador e suas principais características. Para uma descrição detalhada, ver [8].

4.1 Laço Principal

O KSP foi escrito em C e implementa o cálculo apresentado no capítulo anterior. O Algoritmo 1 mostra o algoritmo do laço principal do provador, utilizado para raciocínio local [8].

O provador recebe como entrada um conjunto de declarações e um conjunto de fórmulas modais ou cláusulas, onde o conjunto de declarações é formado pelas opções do usuário. O usuário pode escolher, através da linha de comando ou por meio de um arquivo de configuração, os procedimentos, métodos, simplificações e refinamentos que deseja que sejam executados, bem como informações de saída que deseja que sejam retornadas.

O processamento da entrada foi construído com o gerador de analisadores léxicos Flex e com o gerador de analisadores sintáticos Bison. Esse processamento está indicado na Linha 2 do Algoritmo 1. O usuário pode fornecer o problema na forma de cláusulas ou fórmulas. Se fórmula são fornecidas, o procedimento dado na Linha 3 realiza a transformação na forma clausal apresentada na Tabela 3.4. Cláusulas são armazenadas em tabelas de dispersão para evitar que todo o conjunto de cláusulas seja percorrido quando uma cláusula está sendo procurada.

Nas Linhas 5 a 20 é apresentado o laço principal. O KSP utiliza a estratégia do conjunto de suporte para evitar inferências irrelevantes. Essa estratégia requer que o conjunto de cláusulas seja particionado em dois conjuntos, e restringe que o conjunto de cláusulas possíveis que participarão no passo seguinte. Para a lógica clássica [14], esses

Algoritmo 1: KSP

```
1  início
2      processamento_entrada;
3      transformacao_snf;
4      preprocessamento_clausula;
5      enquanto (Γ^lit_ml ≠ 0) faça
6          para todos os níveis modais ml faça
7              (clausula ← escolha(ml));
8              se não redundante(clausula) então
9                  GEN1 (clausula, ml, ml − 1);
10                 GEN3 (clausula, ml, ml − 1);
11                 LRES (clausula, ml, ml);
12                 Λ^lit ← Λ^lit ∪ {clausula};
13             fim
14             Γ^lit_ml ← Γ^lit_ml/{clausula};
15             se (0 : false ∈ Γ^lit_0) então
16                 retorna insatisfatível;
17             fim
18         fim
19         Γ^lit_ml ← ∪Γ^lit_ml;
20     fim
21     retorna satisfatível;
22 fim
```

dois conjuntos são Γ e Λ, sendo Δ o conjunto de cláusulas a ser particionado e $\Lambda = \Delta\ \Gamma$, onde Γ deve ser satisfatível para que a completude seja garantida. Para a lógica modal, com a forma normal apresentada no Capítulo 3, é possível particionar de acordo com os níveis modais, o que detalharemos a seguir.

O conjunto Γ é o conjunto de suporte (o *sos*, passivo ou conjunto não processado), formado pelos subconjuntos Γ^{lit}_{ml}; e Λ é chamado de *usable* (ativo ou conjunto processado), formado pelos subconjuntos Λ^{lit}_{ml} e Λ^{modal}_{ml}. Para o cálculo modal, o subconjunto de cláusulas é particionado de acordo com o nível modal em que as cláusulas ocorrem: Γ^{lit}_{ml} é o conjunto de suporte para cláusulas literais no nível modal ml; $\Gamma^{lit} = \bigcup_{ml} \Gamma^{lit}_{ml}$, o conjunto de suporte de todas as cláusulas literais; Λ^{lit}_{ml}, o conjunto ativo de cláusulas literais no nível modal ml e Λ^{mod}_{ml}, o conjunto ativo de cláusulas modais no nível modal ml. Como o cálculo não gera novas cláusulas modais e o conjunto de cláusulas modais por si só é satisfatível, visto que são implicações e no lado esquerdo do operador são proposições positivas, não há necessidade de um conjunto para cláusulas modais não processadas.

Os laços entre as Linhas 5 e 20 são responsáveis pela saturação das cláusulas. Na execução do laço de saturação das cláusulas, a *cláusula dada* é escolhida em Γ^{lit}_{ml}, resolvida com cláusulas de Λ^{lit}_{ml} e de Λ^{modal}_{ml-1}, e então movida de Γ^{lit}_{ml} para Λ^{lit}_{ml}.

Durante a aplicação das regras de inferência no laço principal, se a cláusula vazia $0 : \mathbf{true} \rightarrow \mathbf{false}$ é gerada no nível modal 0, então o procedimento retorna que o conjunto de cláusulas é insatisfatível. Se o provador foi configurado para satisfatibilidade global, a condição muda para $* : \mathbf{true} \rightarrow \mathbf{false}$. Se a cláusula vazia não é encontrada, as cláusulas são saturadas ou o conjunto de cláusulas não processadas se torna vazio, o procedimento retorna que o conjunto de cláusulas é satisfatível.

As regras de inferência MRES e GEN2 são aplicadas em *preprocessamento_clausula*, antes do laço principal ser executado porque, durante a busca pela prova, cláusulas modais não são geradas, como pode ser visto na Tabela 3.7.

4.2 Processamento da Entrada

Como dito na seção anterior, o usuário pode fornecer a entrada para o provador através da linha de comando ou de um arquivo de configuração. Caso sejam fornecidas opções tanto por linha de comando quanto por um arquivo, as opções dadas através da linha de comando sobrepõem as opções dadas através do arquivo de configuração. Existem dois tipos de arquivos de entrada aceitos pelo KSP, sendo um deles um arquivo com um conjunto de declarações que indicam as opções para o provador e o outro um arquivo com conjuntos de fórmulas ou cláusulas modais. No segundo tipo, o usuário pode especificar se o conjunto de fórmulas e cláusulas são processadas ou não processadas.

A saída do analisador sintático é constituída por uma árvore abstrata anotada duplamente encadeada e uma tabela de símbolos. Para cada nó da árvore é atribuído um número único. A tabela de símbolos contém informações sobre os símbolos proposicionais, constantes e operadores modais: seu tipo, identificador, número de ocorrências, número de ocorrências negativas e positivas. Uma tabela de dispersão dupla contém a localização da posição dos símbolos proposicionais e das constantes na árvore: o primeiro nível armazena o nível modal em que eles ocorrem e o segundo nível o endereço, dessa forma as exclusões na árvore podem ser feitas, geralmente, em tempo constante.

Uma das opções disponíveis para o usuário é a eliminação de literal puro por nível modal (*mlple*) que substitui todo símbolo proposicional p que é puro no nível modal ml por uma constante. Se p ocorre somente com polaridade positiva em ml, então p é substituído por **true**; se ocorre somente com polaridade negativa, então p é substituído por **false**.

Se a entrada é um conjunto de fórmulas, o usuário possui algumas opções de traduções para formas normais, uma delas é primeiro transformada em NNF ou em BNF. A tradução em BNF aplica todas as regras da reescrista para NNF e remove o operador $\langle a \rangle$, ou seja, o operador $[a]$ é permitido no escopo de negações. A reescrita da fórmula $\neg[a]\varphi$ para NNF

seria $\langle a\rangle \mathrm{NNF}(\neg\varphi)$, mas a tradução dessa fórmula em BNF seria $\neg[a]\mathrm{BNF}(\neg\varphi)$. Esse é o único caso que difere a tradução em NNF para a tradução em BNF.

A transformação em *prenex*, *antiprenex*, ou uma seguida da outra, também são opções disponíveis ao usuário nessa etapa de processamento [15].

Definição 22 Um termo modal é um literal ou fórmula na forma $M_1...M_k l$, onde l é um literal e M_i, $1 \leq i \leq k$, é $[a]$ ou $\neg[a]$ para algum $a \in \mathcal{A}$.

Definição 23 Seja φ e $\psi \in \mathsf{WFF}_{\mathsf{K}_n}$. Uma fórmula γ está na Forma Normal Antiprenex se, e somente se,

1. γ é um termo modal; ou
2. γ é da forma $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$ ou $(\varphi \rightarrow \psi)$, e φ e ψ estão na Forma Normal Antiprenex; ou
3. γ é da forma $[a]\varphi$, φ é uma disjunção, e/ou φ está na Forma Normal Antiprenex; ou
4. γ é da forma $\neg[a]\varphi$, φ é uma conjunção, e/ou φ está na Forma Normal Antiprenex.

A tradução de uma fórmula φ para a Forma Normal Antiprenex é definida pela função $\alpha : \mathsf{WFF}_{\mathsf{K}_n} \rightarrow \mathsf{WFF}_{\mathsf{K}_n}$ apresentada na Tabela 4.1.

$$
\begin{aligned}
\alpha(\varphi) &= \varphi, \text{ se } \varphi \text{ é um termo modal,} \\
\alpha([a](\varphi \wedge \psi)) &= \alpha([a]\varphi \wedge [a]\psi), \\
\alpha([a]\neg(\varphi \rightarrow \psi)) &= \alpha([a]\varphi \wedge [a]\neg\psi), \\
\alpha([a]\neg(\varphi \vee \psi)) &= \alpha([a]\neg\varphi \wedge [a]\neg\psi), \\
\alpha(\neg[a]\neg(\varphi \rightarrow \psi)) &= \alpha([a]\varphi \rightarrow \neg[a]\neg\psi), \\
\alpha(\neg[a]\neg(\varphi \vee \psi)) &= \alpha(\neg[a]\neg\varphi \vee \neg[a]\neg\psi), \\
\alpha([a]\varphi) &= \alpha([a]\alpha(\varphi)), \text{ se } \varphi \text{ é da forma } [a]\psi \text{ ou } \neg[a]\psi, \\
\alpha(\neg[a]\varphi) &= \alpha(\neg[a]\alpha(\varphi)), \text{ se } \varphi \text{ é da forma } [a]\psi \text{ ou } \neg[a]\psi, \\
\alpha([a]\varphi) &= [a]\alpha(\varphi), \text{ se } \varphi \text{ não é da forma de nenhuma das formas acima,} \\
\alpha(\neg[a]\varphi) &= \neg[a]\alpha(\varphi), \text{ se } \varphi \text{ não é da forma de nenhuma das formas acima}
\end{aligned}
$$

Tabela 4.1: Função de tradução para Forma Normal Antiprenex

A função de tradução de uma fórmula φ para a Forma Normal Prenex é similar a função de tradução para forma normal antiprenex.

Além das opções já citadas, *nnfsimp* e *bnfsimp* também estão disponíveis, nesse caso as regras de simplificação presentes na Tabela 3.2 são aplicadas nas fórmulas em NNF e BNF, respectivamente; com as opções *early_ple* e *early_mlple*, eliminação dos literais puros é aplicado globalmente ou em todos os níveis modais, respectivamente.

4.3 Tradução para a Forma Normal

Dentre as opções para transformação em SNF_{ml}, existem quatro possíveis: SNF^{+}_{ml}, SNF^{++}_{ml}, SNF^{-}_{ml} e SNF^{--}_{ml}. A tradução para as formas SNF^{+}_{ml} e SNF^{++}_{ml} foi apresentada nas Tabelas 3.5 e 3.6. O processo de reescrista para SNF^{-}_{ml} e SNF^{--}_{ml} é similar, a diferença se resume aos literais no escopo dos operadores modais que são renomeados por novos literais negativos.

Todos esses processos de tradução requerem renomeamento. Dentre as opções de renomeamento, o provador dispõe de quatro: *normal_renaming*, onde toda subfórmula é renomeada por um novo símbolo proposicional; *limited_reuse_renaming*, onde o novo símbolo proposicional é usado para todas as ocorrências da mesma subfórmula sendo renomeada em um determinado nível modal; *extensive_reuse_renaming* faz o mesmo que a opção anterior, e além disso, se uma fórmula φ foi renomeada por um símbolo proposicional novo t, então a NNF de $\neg\varphi$ é renomeada para $\neg t$; e *conjunct_renaming* que renomeia as subfórmulas modais que ocorrem em conjunções, em vez de aplicar a regra de reescrita usual.

O conjunto de cláusulas na Forma Normal Separada é armazenado em uma estrutura do tipo *trie* implementada como tabelas de dispersão multi-nível, de acordo com o conjunto que ele pertence (processado ou não processado), seu tipo (inicial, literal, modal positivo ou modal negativo), seu nível modal, o índice do operador modal (caso sejam cláusulas modais), seu literal máximo, e seu tamanho (caso sejam cláusulas literais ou iniciais).

4.4 Pré-processamento de Cláusulas

Várias tarefas de escolha do usuário são executadas nessa etapa. A opção *check_full_repeated* verifica a duplicidade de uma cláusula em todos os conjuntos de cláusulas no mesmo nível modal. Caso essa opção não fosse configurada, a verificação seria feita apenas no conjunto em que a cláusula está armazenada. A opção *propdia* aplica a regra de propagação da Tabela 4.2.

$$\frac{ml: \quad l' \rightarrow \langle a \rangle l}{ml+1: \quad l}$$

Tabela 4.2: Regra de propagação para os literais l e l', nível modal ml, agente a, com a condição que existe somente uma cláusula modal negativa em Λ_{ml}^{modal}.

A opção *mle* deleta todas as cláusulas positivas modais do nível modal ml que não contêm cláusula modal negativa, e todas as cláusulas nos níveis modais superiores. Isso pode ser feito porque se existe um mundo w que satisfaz $[a]l$, para um literal l e agente a, e não existe nenhum mundo w' tal que $(w, w') \in R_a$, a relação vazia para todos os mundos no nível modal ml satisfaz todas as cláusulas modais positivas. Depois da exclusão das cláusulas modais positivas do nível modal ml, nenhum mundo em um nível modal maior é acessível, por isso todas as cláusulas nos níveis modais superiores também podem ser deletadas.

Nessa etapa as regras de inferência MRES e GEN2 são aplicadas. Através das opções *mres* e *gen2* essas regras são forçosamente aplicadas, mesmo em caso de bloqueio por causa das transformações para SNF_{ml}^{+}, SNF_{ml}^{++}, SNF_{ml}^{-} e SNF_{ml}^{--}. O bloqueio acontece porque esses processos de renomeamento fazem com que todos os literais no escopo de operadores modais tenham a mesma polaridade.

Outras opções podem ser vistas em [8].

4.5 Seleção de Cláusulas

Além da estratégia do conjunto de suporte explicada anteriormente, há mais cinco heurísticas disponíveis para configuração pelo usuário se tratando da escolha de cláusula literal: *shortest*, onde a cláusula com menor tamanho em um determinado nível modal é escolhida; *newest*, onde a seleção de cláusulas simula uma estrutura tipo pilha; *oldest*, onde a seleção de cláusulas simula uma estrutura tipo fila; *smallest*, onde a cláusula escolhida é a cláusula de menor tamanho com o menor literal maximal; e *greatest*, onde a cláusula escolhida é a cláusula de menor tamanho com o maior literal maximal.

4.6 Refinamentos

Além das restrições mencionadas na seção anterior, o usuário também pode restringir a aplicação de LRES ao configurar as opções *ordered*, onde as cláusulas só podem ser

resolvidas em seus literais maximais com respeito a uma ordenação escolhida pelo provador para preservar completude; *negative*, onde uma das premissas é uma cláusula negativa; *positive*, onde uma das premissas é uma cláusula positiva; e *negord*, onde tanto resolução ordernada (*ordered*) como negativa (*negative*) são aplicadas.

Porém, para manter completude, resolução negativa requer SNF^{+}_{ml} ou SNF^{++}_{ml}, resolução ordenada requer SNF^{++}_{ml} e resolução positiva (*positive*) requer SNF^{-}_{ml} ou SNF^{--}_{ml}.

4.7 Eliminação de Redundância

Durante o processamento de cláusulas, para eliminação de redundância, há cinco opções disponíveis para o usuário: *ple*, *mlple*, *fsub*, *bsub* e *sos_sub*.

A eliminação de literal puro por nível modal (*mlple*) substitui todo literal l em um nível modal ml por **true**, se a polaridade de l for positiva, e por **false**, se a polaridade for negativa. No caso em que a polaridade de l no nível modal ml é positiva, todas as cláusulas literais em ml que l ocorre podem ser excluídas. Se l ocorre no lado direito do escopo do operador $[a]$ de uma cláusula modal positiva, a cláusula modal positiva pode ser deletada porque ela é substituída por $[a]\mathbf{true}$ - uma tautologia. Se l ocorre no lado direito do escopo do operador $\langle a\rangle$ de uma cláusula modal positiva $ml - 1 : l' \rightarrow \langle a\rangle l$, a cláusula é substituída por $ml - 1 : l' \rightarrow \langle a\rangle\mathbf{true}$. Mas diferente do caso anterior, $\langle a\rangle\mathbf{true}$ não é uma tautologia, então ela é mantida no conjunto de cláusulas. O procedimento para a opção *ple* é similar, a diferença é que a eliminação do literal puro é aplicada globalmente.

A Definição 24 a seguir define subsunção, o que será importante para o entendimento das opções *fsub*, *bsub* e *sos_sub*.

Definição 24 Seja C e D disjunções de literais. A cláusula $ml : C$ subsume uma cláusula $ml' : D$ se, e somente se, $\sigma(\{ml, ml'\}) = ml'$ e D é da forma $C \vee D$, onde D é uma disjunção de literais (possivelmente vazia).

A opção *fsub* deleta uma cláusula literal $ml : C$ se ela é subsumida por qualquer cláusula literal mais antiga. Enquanto a opção *bsub* deleta uma cláusula literal $ml : C$ se ela é subsumida por qualquer cláusula literal mais nova. Nos dois casos, subsunção é aplicada em modo "preguiçoso" [16]: uma cláusula somente é testada para subsunção quando é selecionada em Γ^{lit}_{ml} e contra cláusulas em Λ^{lit}_{ml}. A opção *sos_sub* força a verificação em todo o conjunto de cláusulas.

Capítulo 5

Implementação e Avaliação Experimental

Neste capítulo iremos descrever a implementação e experimentos realizados.

5.1 Implementação

Na implementação original do provador, a busca por cláusulas candidatas a premissas era restringida pelo nível modal da cláusula escolhida. Dessa forma, se o raciocínio fosse local, cláusulas em Λ_*^{lit} e em Λ_*^{modal} não eram consideradas. Porém, de acordo com a definição das regras de inferência, que pode ser vista na Tabela 3.7, a restrição sobre os níveis modais das premissas é imposta pela função de unificação. Isto é, resolução pode ser aplicada entre cláusulas de nível modal global e local se a função de unificação que recebe como entrada seu níveis modais for diferente de indeterminado.

Portanto, o primeiro passo da refatoração foi implementar a função de unificação no KSP de acordo com o Algoritmo 2, que recebe como entrada ml e ml', o nível modal de duas cláusulas.

A implementação da função de unificação recebe como parâmetro somente dois níveis modais, porém as regras de inferência GEN1, GEN2 e GEN3 têm um conjunto de premissas que pode ter tamanho maior que dois. Nesse caso, a função é chamada quantas vezes for necessário passando como parâmetro a saída anterior, caso seja diferente de *indeterminado*, e o nível modal da próxima cláusula até que todas as premissas tenham sido verificadas. Seja Ψ o conjunto de candidatos, *nivel_modal* a função que devolve o nível modal da cláusula passada como parâmetro, e *escolha* a função que retorna uma cláusula do conjunto passado como parâmetro, o Algoritmo 3 apresenta esse procedimento de verificação.

Algoritmo 2: Função de Unificação

```
1  início
2  |  se (ml = ml′) então
3  |  |  retorna ml;
4  |  fim
5  |  se (ml = ∗ ∧ ml′ ≠ ∗) então
6  |  |  retorna ml′;
7  |  fim
8  |  se (ml ≠ ∗ ∧ ml′ = ∗) então
9  |  |  retorna ml;
10 |  fim
11 |  retorna indeterminado;
12 fim
```

Algoritmo 3: Verificação das premissas de GEN1, GEN2 e GEN3

```
1  início
2  |  clausula ← escolha(Ψ);
3  |  ml ← nivel_modal(clausula);
4  |  Ψ ← Ψ/{clausula};
5  |  enquanto (Ψ ≠ 0) faça
6  |  |  clausula ← escolha(Ψ);
7  |  |  ml′ ← nivel_modal(clausula);
8  |  |  ml ← unificacao(ml, ml′);
9  |  |  se (ml = indeterminado) então
10 |  |  |  retorna indeterminado;
11 |  |  fim
12 |  |  Ψ ← Ψ/{clausula};
13 |  fim
14 |  retorna ml;
15 fim
```

Em seguida, as funções que implementam as regras de inferência foram modificadas para receber como parâmetro, além da cláusula escolhida, o conjunto, ou conjuntos nos casos de GEN1, GEN2 e GEN3, em que as outras premissas devem ser procuradas.

No primeiro laço, entre as Linhas 5 e 17 do Algoritmo 4, o nível modal da cláusula escolhida é global, nesse caso as cláusulas modais candidatas com nível modal local podem ter qualquer valor, contanto que todas tenham o mesmo, então nesse caso a busca percorre todos os níveis modais. No segundo laço, entre as Linhas 18 e 34, seja ml o nível modal da cláusula escolhida, as funções foram modificadas para buscar cláusulas modais candidatas com nível modal global ou nível modal igual a $ml - 1$, nos casos de GEN1 e GEN3, e ml no caso de LRES. A busca nos outros níveis modais locais não é necessária visto

que a saída da função de unificação entre qualquer outro nível modal diferente de $ml - 1$ e ml resultaria em *indeterminado*. As modificações em MRES foram equivalentes as modificações em LRES e as alterações em GEN2 foram similares as modificações em GEN1 e GEN3.

5.2 Experimentação

Além de garantir que a implementação da combinação dos raciocínios local e global seja correta, também era necessário garantir que a aplicação de resolução em conjuntos de cláusulas somente com nível modal local ou global continuasse correta, e que o tempo de execução não tenha sofrido alteração significativa.

O experimento foi conduzido em um ambiente com as seguintes características:

- Linux Ubuntu 20.04.2 LTS
- Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz × 8
- 8GB RAM

No experimento foi utilizado um conjunto de fórmulas formado por 378 fórmulas divididas em 9 famílias com 42 arquivos cada, sendo que 21 contém fórmulas satisfatíveis e 21 insatisfatíveis. Mais informações sobre o conjunto podem ser adquiridas em [17]. Além disso, foram utilizados também 8 arquivos de configuração: `snf+#negative_jar`, `snf++#ordered_jar`, `snf#plain_jar`, `snf-#positive_jar`, `snf+#negative_jar_global`, `snf++#ordered_jar_global`, `snf#plain_jar_global` e `snf-#positive_jar_global`. Esses arquivos são distribuídos junto com o KSP. As opções de cada arquivo de configuração podem ser vistas na Tabela 5.1.

As diferenças entre os arquivos para raciocínio local (terminados em `_jar`) e seus correspondentes para raciocínio global (terminados em `_jar_global`) são as opções *local* que é substituída por *global*, *early_mlple* que é substituída por *early_ple* e *early_ple* que é substituída por *ple*.

Todos os 42 arquivos de teste foram executados para cada um dos 8 arquivos de configuração no $K_{\mathrm{S}}P$ pré-modificações e no $K_{\mathrm{S}}P$ pós-modificações, em seguida os tempos de execução foram comparados e as diferenças calculadas. A Tabela 5.2 mostras as médias das diferenças de tempo obtidas e a diferença máxima de acordo com cada arquivo de configuração. Um tempo limite de 30 segundos foi estabelecido, fórmulas cuja execução ultrapassou esse tempo limite não foram consideradas.

Para as fórmulas cuja execução não ultrapassou o tempo limite, as saídas possíveis eram **satisfatível** e **insatisfatível**. A partir dos resultados apresentados na Tabela 5.2

pode-se concluir que o tempo de execução da aplicação de resolução em conjuntos de cláusulas somente com nível modal local ou global não sofreu alteração significativa em comparação com o tempo de execução do KSP pré-modificações.

A segunda parte do experimento consistiu na execução de exemplos para testar a combinação dos raciocínios local e global. Quatro conjuntos de cláusulas foram criados sendo três deles satisfatíveis e um insatisfatível, em todos eles a saída obtida estava correta. Por último, o exemplo apresentado em [1], que é insatisfatível, foi utilizado como teste e que produziu erros. O conjunto de cláusulas deste exemplo é reproduzido a seguir.

1. $*: \ t_0$
2. $*: \ \neg t_0 \vee t_1 \vee t_2$
3. $*: \ t_1 \rightarrow [c]t_f$
4. $*: \ t_2 \rightarrow \langle d \rangle t_t$
5. $*: \ \neg t_f$
6. $0: \ t_0 \rightarrow \langle c \rangle t_3$
7. $1: \ t_3 \rightarrow \langle c \rangle t_t$
8. $0: \ t_0 \rightarrow [d]t_4$
9. $1: \ t_4 \rightarrow [d]t_f$
10. $1: \ \neg t_4 \vee nec_{d,t_f}$
11. $*: \ nec_{d,t_f} \rightarrow [d]t_f$
12. $*: \ \neg nec_{d,t_f} \rightarrow \langle d \rangle \neg nec_{d,t_f}$
13. $*: \ nec_{d,t_f} \rightarrow [d]nec_{d,t_f}$

A codificação deste exemplo na linguagem do provador, bem com a saída completa gerada durante a sua execução, são apresentados no Anexo I. O resultado produzido difere do esperado: para a codificação deste exemplo, o provador forneceu "satisfatível" como resposta. Abaixo serão apresentados alguns dos problemas observados e que colaboram para a saída incorreta do provador.

- Resolução modal está sendo aplicada quando a função de unificação retorna um inteiro negativo. Observe na Definição 16 que o domínio é um conjunto de rótulos e resulta em um rótulo, onde rótulos pertencem a $\mathbb{N} \cup \{*\}$. Portanto, unificação que não seja aplicada a conjunto de tais elementos ou que não resulte em um número natural (ou $*$) não é definida. Exemplos incorretos produzidos pela implementação da função de unificação são encontrados nas Cláusulas 54 e 56:

  53. $0: \ nec_{d,t_f} \quad [\text{LRES}, 35, 1, \ t_0]$

  54. $-1: \ nec_{d,t_f} \quad [\text{GEN1}, 12, 53, \ nec_{d,t_f}]$

  56. $-2: \ nec_{d,t_f} \quad [\text{GEN1}, 12, 54, \ nec_{d,t_f}]$

  Observe que para aplicação de GEN1 às Cláusulas 12 e 53 é preciso determinar o resultado da unificação a $\{-1, *\}$, onde -1 corresponde a $ml - 1$ com $ml = 0$ na Cláusula 53 e $*$ é o rótulo da Cláusula 12, que é modal. Como este conjunto não

está no domínio da função, o resultado seria indeterminado e a regra não poderia ter sido aplicada. Para a definição das restrições de unificação para cada uma das regras de inferencia, ver Tabela 3.7.

- Resolução modal está sendo aplicada a cláusulas modais relativas a diferentes agentes. Observe na Tabela 3.7, que as cláusulas modais nas premissas das regras sempre referem ao mesmo agente a. Entretanto, a seguinte cláusula é produzida:

$$16. \quad *: \quad \neg t_1 \vee \neg t_2 \quad [\text{GEN3}, 3, 5, 4, t_f, t_t]$$

  mas a Cláusula 3 é uma cláusula c-positiva, enquanto a Cláusula 4 é uma cláusula d-negativa; ou seja, não podem ser resolvidas.

- Resolução não está sendo aplicada exaustivamente. Por exemplo, a cláusula $* : \neg nec_{d,t_f} \vee \neg t_2$ pode ser obtida pela aplicação de GEN3 às Cláusulas 4, 5 e 11. Entretanto, esta cláusula, que é utilizada para a obtenção da prova apresentada em [1], não é gerada.

- Resolução está sendo aplicada mais de uma vez sobre o mesmo conjunto de premissas. O problema ocorre em todas as regras de inferência que foram utilizadas. Exemplos são apresentados abaixo:

$$\begin{array}{llll}
14 & *: & \textbf{true} & [\text{MRES}, 12, 13, \ nec_{d,t_f}] \\
15 & *: & \textbf{true} & [\text{MRES}, 12, 13, \ nec_{d,t_f}] \\
 & & & \\
16 & *: & t_1 \vee \ t_2 & [\text{GEN3}, 3, 5, 4, t_f, t_t] \\
19 & *: & t_1 \vee \ t_2 & [\text{GEN3}, 3, 5, 4, t_f, t_t] \\
 & & & \\
17 & *: & t_1 \vee nec_{d,t_f} & [\text{GEN3}, 3, 5, 12, t_f, \ nec_{d,t_f}] \\
20 & *: & t_1 \vee nec_{d,t_f} & [\text{GEN3}, 3, 5, 12, t_f, \ nec_{d,t_f}] \\
 & & & \\
35 & 0: & t_0 \vee nec_{d,t_f} & [\text{GEN1}, 8, 12, 10, t_4, \ nec_{d,t_f}] \\
36 & 0: & t_0 \vee nec_{d,t_f} & [\text{GEN1}, 8, 12, 10, t_4, \ nec_{d,t_f}] \\
 & & & \\
37 & 0: & t_2 \vee \ t_0 & [\text{LRES}, 18, 2, \ t_1] \\
38 & 0: & t_2 \vee \ t_0 & [\text{LRES}, 18, 25, \ t_1]
\end{array}$$

Conclui-se que embora os testes mostrem que os processos isolados de raciocínio, local ou global, não tenham sido afetados, ainda existem alguns problemas na implementação para a combinação. Em particular, é necessário revisar as implementações de GEN1 e GEN3 para corrigir o problema de geração indevida de cláusulas e também para garantir que todas as cláusulas devidas sejam efetivamente geradas.

Algoritmo 4: KSP pós-modificações

```
1  início
2      processamento_entrada;
3      transformacao_snf;
4      preprocessamento_clausula;
5      enquanto (Γ*^lit ≠ 0) faça
6          (clausula ← escolha(∗));
7          se não redundante(clausula) então
8              GEN1 (clausula, Λ_ml^modal);
9              GEN3 (clausula, Λ_ml^modal);
10             LRES (clausula, Λ_ml^lit);
11             Λ^lit ← Λ^lit ∪ {clausula};
12         fim
13         Γ*^lit ← Γ*^lit/{clausula};
14         se ((∗ : false ∈ Γ*^lit) ∨ (0 : false ∈ Γ_0^lit)) então
15             retorna insatisfatível;
16         fim
17     fim
18     enquanto (Γ_ml^lit ≠ 0) faça
19         para todos os níveis modais ml faça
20             (clausula ← escolha(ml));
21             se não redundante(clausula) então
22                 GEN1 (clausula, Λ_{ml−1}^modal, Λ*^modal);
23                 GEN3 (clausula, Λ_{ml−1}^modal, Λ*^modal);
24                 LRES (clausula, Λ_ml^lit, Λ_ml^modal);
25                 Λ_ml^lit ← Λ_ml^lit ∪ {clausula};
26             fim
27             Γ_ml^lit ← Γ_ml^lit/{clausula};
28             se (0 : false ∈ Γ_0^lit) então
29                 retorna insatisfatível;
30             fim
31         fim
32         Γ_ml^lit ← ∪Γ_ml^lit;
33     fim
34     retorna satisfatível;
35 fim
```

Tabela 5.1: Arquivos de Configuração

`snf+#negative_jar`	`snf++#ordered_jar`	`snf#plain_jar`	`snf-#positive_jar`
early_mlple	*early_mlple*	*early_mlple*	*early_mlple*
snf+	*snf++*		*snf-*
negative	*ordered*		*positive*
local	*local*	*local*	*local*
shortest	*shortest*	*shortest*	*shortest*
unit	*unit*	*unit*	*unit*
lhs_unit	*lhs_unit*	*lhs_unit*	*lhs_unit*
propdia	*propdia*	*propdia*	*propdia*
mres	*mres*	*mres*	*mres*
mlple	*mlple*	*mlple*	*mlple*
fsub	*fsub*	*fsub*	*fsub*
bsub	*bsub*	*bsub*	*bsub*
limited_reuse _renaming	*limited_reuse _renaming*	*limited_reuse _renaming*	*limited_reuse _renaming*
prenex	*prenex*	*prenex*	*prenex*
maxproof,1	*maxproof,1*	*maxproof,1*	*maxproof,1*

Tabela 5.2: Média das diferenças e diferença máxima entre os testes

Arquivo de configuração	Média das diferenças em segundos	Diferença Máxima em segundos
`snf+#negative_jar`	-0,00293651	0,59
`snf++#ordered_jar`	0,004285714	1,65
`snf#plain_jar`	-0,00132275	0,66
`snf-#positive_jar`	-0,00243386	1,24
`snf+#negative_jar_global`	0,002566138	0,79
`snf++#ordered_jar_global`	0,010661376	1,15
`snf#plain_jar_global`	0,083862434	4,88
`snf-#positive_jar_global`	0,061984127	5,58

Capítulo 6

Conclusão

Este trabalho tinha como objetivo a refatoração das regras de inferência do provador baseado em resolução para a Lógica Multimodal K_n KSP para possibilitar a combinação dos raciocínios local e global.

Após as modificações feitas no KSP, duas experimentações foram executadas para garantir que os raciocínios local e global, considerados isoladamente, permaneciam corretos e apresentavam performance semelhante ao provador pré-modificações, e que a combinação dos dois cálculos estivesse correta e completa.

A primeira avaliação experimental foi feita sobre um conjunto de 378 fórmulas e 8 arquivos de configuração. Os resultados mostraram que as saídas da aplicação dos cálculos local e global separadamente condisseram com o esperado. Também foi possível concluir que a alteração no tempo de execução em comparação ao tempo pré-modificações não foi significativa, visto que a média das diferenças foi menor que um segundo em todos os casos e a diferença máxima obtida foi aproximadamente 5 segundos. Isso indica que a performance do KSP é semelhante à performance do provador pré-modificações.

A segunda etapa de testes foi executada sobre cinco conjuntos de cláusulas. Apesar de quatro delas retornarem resultados esperados, uma delas revelou que a implementação da combinação dos dois raciocínios ainda necessita de refatoração para correção de erros. O cálculo indevido de cláusulas por GEN1 e GEN3 e a não-saturação do conjunto de cláusulas foram os erros encontrados. As causas ainda não foram descobertas. A correção desses erros e a execução de testes sobre um número significativo de conjuntos de cláusulas são sugestões para um trabalho futuro.

Referências

[1] Nalon, Cláudia, Clare Dixon e Ullrich Hustadt: *Modal resolution: Proofs, layers, and refinements*. ACM Transactions on Computational Logic, 20(4):23:1–23:38, agosto 2019, ISSN 1529-3785. `http://doi.acm.org/10.1145/3331448`. viii, 13, 17, 20, 21, 22, 33, 34, 40

[2] Hailpern, Brent T: *Verifying concurrent processes using temporal logic*. Número 129 em *Lecture Notes in Computer Science*. Springer Science & Business Media, 1982. 1

[3] Fitting, Melvin e Richard L Mendelsohn: *First-Order Modal Logic*, volume 277 de *Synthese Library*. Springer Science & Business Media, 1998. 1

[4] Blackburn, Patrick, Marteen de Rijke e Yde Venema: *Modal logic*. Cambridge University Press, 2001. 1

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

[6] Spaan, Edith: *Complexity of modal logics*. Tese de Doutoramento, University of Amsterdam, The Netherlands, 1993. 2, 3

[7] Robinson, John Alan: *A machine-oriented logic based on the resolution principle*. Journal of the ACM (JACM), 12(1):23–41, 1965. 2

[8] Nalon, Cláudia, Hustadt Ullrich e Clare Dixon: *$K_S P$: A resolution-based theorem prover for K_n: Architecture, refinements, strategies and experiments*. Journal of Automated Reasoning, 64(3):461–484, 2020. 2, 11, 13, 23, 28

[9] Cook, Stephen A: *The complexity of theorem-proving procedures*. Em *Proceedings of the third annual ACM symposium on Theory of computing*, páginas 151–158, 1971. 3

[10] Areces, Carlos, Rosella Gennari, Juan Heguiabehere e Maarten de Rijke: *Tree-based heuristics in modal theorem proving*. Em Horn, W (editor): *Proceedings of ECAI'2000*, páginas 199–203, 2000. 10

[11] Goranko, Valentin e Solomon Passy: *Using the universal modality: gains and questions*. Journal of Logic and Computation, 2(1):5–30, 1992. 11

[12] Plaisted, David A e Steven Greenbaum: *A structure-preserving clause form translation*. Journal of Symbolic Computation, 2(3):293–304, 1986. 14

[13] Slagle, James R.: *Automatic theorem proving with renamable and semantic resolution.* Journal of the ACM, 14(4):687–697, outubro 1967, ISSN 0004-5411. 16

[14] Wos, Lawrence, George A Robinson e Daniel F Carson: *Efficiency and completeness of the set of support strategy in theorem proving.* Journal of the ACM (JACM), 12(4):536–541, 1965. 23

[15] Nalon, Cláudia e Clare Dixon: *Anti-prenexing and prenexing for modal logics.* Em *Logics in Artificial Intelligence: 10th European Conference, JELIA 2006 Liverpool, UK, September 13-15, 2006 Proceedings 10*, páginas 333–345. Springer, 2006. 26

[16] Schulz, Stephan: *Simple and efficient clause subsumption with feature vector indexing.* Automated Reasoning and Mathematics: Essays in Memory of William W. McCune, páginas 45–67, 2013. 29

[17] Balsiger, Peter, Alain Heuerding e Stefan Schwendimann: *A benchmark method for the propositional modal logics K, KT, S4.* Journal of Automated Reasoning, 24(3):297–317, 2000. 32

Anexo I

Exemplo

Na Figura I.1 é apresentado o conteúdo do arquivo de entrada para o KSP, com as clásulas apresentadas no Exemplo 4.6 em [1, Páginas 12 a 14]. Na linguagem do provador, os símbolos `=>`, `~` e `|` representam implicação, negação e disjunção, respectivamente. Na entrada, a variável `nec_d_tf` representa nec_{d,t_f} do exemplo original.

```
sos(clauses).
* : nec_d_tf => [d] nec_d_tf.
* : ~nec_d_tf => <d> ~nec_d_tf.
* : nec_d_tf => [d] tf.
1: ~t4 | nec_d_tf.
1 : t4 => [d] tf.
0 : t0 => [d] t4.
1 : t3 => <c> tt.
0 : t0 => <c> t3.
* : ~tf.
* : t2 => <d> tt.
* : t1 => [c] tf.
* : ~t0 | t1 | t2.
* : t0.
end_of_list.
```

Figura I.1: Arquivo de entrada para o KSP [1, Exemplo 4.6, Páginas 12 a 14]

Na Listagem I.1 é apresentada a saída do KSP para este exemplo. A linha de comando utilizada foi:

```
./ksp -full_check_repeated -i my_global_example.ksp -pgen -pdel
```

onde `-full_check_repeated` é a opção para verificação de repetição de cláusulas em ambos os conjuntos, passivo e ativo; e `-pgen` e `-pdel` são as opções para impressão das cláusulas geradas e removidas durante o processamento. Na saída, os operadors $[a]$ e $\langle a \rangle$, para um índice a, são representados por `box a` e `~box a~`, respectivamente. Cada cláusula é numerada por um par, onde o primeiro valor corresponde ao número da cláusula e o segundo valor, ao seu nível modal. Após cada cláusula é apresentada a justificativa: `Given` corresponde a uma cláusula da entrada; as demais justificativas compreendem a regra de inferência utilizada, suas premissas e literais resolvidos. Se houver, a segunda justificativa corresponde à razão para remoção da cláusula do conjunto.

Listagem I.1: Saída do KSP para a entrada dada na Figura I.1

```
[1,*]. true => t0  [Given]
[2,*]. true => t1 | t2 | ~t0  [Given]
[3,*]. t1 => box c tf [Given]
[4,*]. t2 => ~box d~ tt  [Given]
[5,*]. true => ~tf [Given]
[6,0]. t0 => ~box c~ t3  [Given]
[7,1]. t3 => ~box c~ tt  [Given]
[8,0]. t0 => box d t4 [Given]
[9,1]. t4 => box d tf [Given]
[10,1]. true => ~t4 | nec_d_tf [Given]
[11,*]. nec_d_tf => box d tf  [Given]
[12,*]. ~nec_d_tf => ~box d~ ~nec_d_tf [Given]
[13,*]. nec_d_tf => box d nec_d_tf [Given]
[14,*]. true => true  [MRES,12,13,~nec_d_tf]
Deleted:
[14,*]. true => true  [MRES,12,13,~nec_d_tf][Tautology]
[15,*]. true => true  [MRES,12,13,~nec_d_tf]
Deleted:
[15,*]. true => true  [MRES,12,13,~nec_d_tf][Tautology]
[16,*]. true => ~t1 | ~t2  [GEN3,3,5,4,tf,tt]
[17,*]. true => ~t1 | nec_d_tf [GEN3,3,5,12,tf,~nec_d_tf]
[18,0]. true => ~t1 | ~t0  [GEN3,3,5,6,tf,t3]
Deleted:
[19,*]. true => ~t1 | ~t2  [GEN3,3,5,4,tf,tt][Repeated,16]
Deleted:
[20,*]. true => ~t1 | nec_d_tf [GEN3,3,5,12,tf,~nec_d_tf][Repeated,17]
```

```
[21,1]. true => ~t1 | ~t3  [GEN3,3,5,7,tf,tt]
[22,*]. true => true  [LRES,2,16,t1]
Deleted:
[22,*]. true => true  [LRES,2,16,t1][Tautology]
[23,*]. true => t2 | ~t0 | nec_d_tf  [LRES,2,17,t1]
[24,*]. true => true  [LRES,2,16,t2]
Deleted:
[24,*]. true => true  [LRES,2,16,t2][Tautology]
[25,*]. true => t1 | t2  [LRES,2,1,~t0]
[26,*]. true => true  [LRES,25,16,t1]
Deleted:
[26,*]. true => true  [LRES,25,16,t1][Tautology]
[27,*]. true => t2 | nec_d_tf [LRES,25,17,t1]
[28,*]. true => true  [LRES,25,16,t2]
Deleted:
[28,*]. true => true  [LRES,25,16,t2][Tautology]
Deleted:
[29,*]. true => ~t1 | nec_d_tf [LRES,27,16,t2][Repeated,17]
[30,*]. true => ~t1 | ~t0 | nec_d_tf  [LRES,23,16,t2]
Deleted:
[31,*]. true => t2 | nec_d_tf [LRES,23,1,~t0][Repeated,27]
Deleted:
[32,*]. true => t2 | ~t0 | nec_d_tf  [LRES,30,2,~t1][Repeated,23]
Deleted:
[33,*]. true => t2 | ~t0 | nec_d_tf  [LRES,30,25,~t1][Repeated,23]
Deleted:
[34,*]. true => ~t1 | nec_d_tf [LRES,30,1,~t0][Repeated,17]
[35,0]. true => ~t0 | nec_d_tf [GEN1,8,12,10,t4,~nec_d_tf]
Deleted:
[36,0]. true => ~t0 | nec_d_tf [GEN1,8,12,10,t4,~nec_d_tf][Repeated,35]
[37,0]. true => t2 | ~t0 [LRES,18,2,~t1]
Deleted:
[38,0]. true => t2 | ~t0 [LRES,18,25,~t1][Repeated,37]
[39,0]. true => ~t1  [LRES,18,1,~t0]
[40,1]. true => t2 | ~t3 | ~t0 [LRES,21,2,~t1]
[41,1]. true => t2 | ~t3 [LRES,21,25,~t1]
Deleted:
```

```
[42,0]. true => t2 | ~t0 [LRES,39,2,~t1][Repeated,37]
[43,0]. true => t2 [LRES,39,25,~t1]
Deleted:
[44,1]. true => ~t1 | ~t3  [LRES,41,16,t2][Repeated,21]
Deleted:
[45,0]. true => ~t1  [LRES,43,16,t2][Repeated,39]
[46,1]. true => ~t1 | ~t3 | ~t0  [LRES,40,16,t2]
Deleted:
[47,1]. true => t2 | ~t3 [LRES,40,1,~t0][Repeated,41]
Deleted:
[48,0]. true => ~t1 | ~t0  [LRES,37,16,t2][Repeated,18]
Deleted:
[49,0]. true => t2 [LRES,37,1,~t0][Repeated,43]
Deleted:
[50,1]. true => t2 | ~t3 | ~t0 [LRES,46,2,~t1][Repeated,40]
Deleted:
[51,1]. true => t2 | ~t3 | ~t0 [LRES,46,25,~t1][Repeated,40]
Deleted:
[52,1]. true => ~t1 | ~t3  [LRES,46,1,~t0][Repeated,21]
[53,0]. true => nec_d_tf [LRES,35,1,~t0]
[54,-1]. true => nec_d_tf  [GEN1,12,53,~nec_d_tf]
Deleted:
[55,-1]. true => nec_d_tf  [GEN1,12,53,~nec_d_tf][Repeated,54]
[56,-2]. true => nec_d_tf  [GEN1,12,54,~nec_d_tf]
Deleted:
[57,-2]. true => nec_d_tf  [GEN1,12,54,~nec_d_tf][Repeated,56]
Satisfiable.
```