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

Um Provador de Teoremas baseado em Tableaux para Verificação de Propriedades Temporais de Conhecimento ou Crença

Thiago Coelho Vieira

Documento apresentado para o
exame de Mestrado em Informática

Orientadora
Prof.ª Dr.ª Cláudia Nalon

Brasília
2015

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Mestrado em Informática

Coordenadora: Prof.ª Dr.ª Alba C. M. A. Melo

Banca examinadora composta por:

Prof.ª Dr.ª Cláudia Nalon (Orientadora) — CIC/UnB
Prof.ª Dr.ª Daniele Nantes Sobrinho — MAT/UnB
Prof. Dr. Flávio Leonardo Cavalcanti de Moura — CIC/UnB

CIP — Catalogação Internacional na Publicação

Coelho Vieira, Thiago.

Um Provador de Teoremas baseado em Tableaux para Verificação de Propriedades Temporais de Conhecimento ou Crença / Thiago Coelho Vieira. Brasília : UnB, 2015.
57 p. : il. ; 29,5 cm.

Documento (Mestrado) — Universidade de Brasília, Brasília, 2015.

1. Provadores de Teoremas, 2. Tableaux, 3. Verificação Formal,
4. Lógicas Epistêmicas, 5. Lógicas de Crença, 6. Lógicas Temporais,
7. Lógica Modal.

CDU 004

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

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

Um Provador de Teoremas baseado em Tableaux para Verificação de Propriedades Temporais de Conhecimento ou Crença

Thiago Coelho Vieira

Documento apresentado para o
exame de Mestrado em Informática

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

Prof.ª Dr.ª Daniele Nantes Sobrinho
MAT/UnB

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

Prof.ª Dr.ª Alba C. M. A. Melo
Coordenadora do Mestrado em Informática

Brasília, 07 de Janeiro de 2015

Dedicatória

Dedico este trabalho à minha esposa Milena que sempre esteve ao meu lado me apoiando em mais uma etapa da minha vida. Também dedico à minha avó Zenaide (em memória) pelo apoio fundamental dado em minha vida acadêmica. Por fim, dedico também aos meus pais.

Agradecimentos

Agradeço a Deus por me dar a oportunidade de concluir esta etapa da minha vida. Agradeço também à minha orientadora, Dr.ª Cláudia Nalon, pela perseverança, paciência e apoio dado durante toda essa jornada.

Resumo

Diversos tipos de lógicas são usadas como linguagens para descrever sistemas complexos e suas propriedades com a finalidade de serem verificadas formalmente. Provadores de teoremas baseados em tableaux são ferramentas computacionais capazes de realizar esta tarefa de verificação. Em (WDF98) é proposto um método de prova baseado em tableaux para duas lógicas epistêmico-temporais, $KL_{(n)}$ e $BL_{(n)}$. Neste trabalho implementamos o método de prova baseado em tableaux descrito em (WDF98) e apresentamos um algoritmo para verificação de propriedades epistêmicas e temporais sobre a estrutura do tableau construída por este método.

Palavras-chave: Provadores de Teoremas, Tableaux, Verificação Formal, Lógicas Epistêmicas, Lógicas de Crença, Lógicas Temporais, Lógica Modal.

Abstract

Logics are used as languages to describe complex systems and their properties in order to be formally verified. Tableaux-based theorem-provers are computational tools which can be used to perform this verification task. (WDF98) propose a proof method based on tableaux for both the epistemic-temporal logics $KL_{(n)}$ and $BL_{(n)}$. In this work we implement the tableaux-based proof method described in (WDF98) and present an algorithm for verification of epistemic-temporal properties over the structure of the tableau built by this method.

Keywords: Automated Theorem-Proving, Tableaux, Formal Verification, Epistemic Logics, Logics of Belief, Temporal Logics, Modal Logic.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Em Ciência da Computação, lógica pode ser utilizada como linguagem para descrever propriedades de sistemas complexos. Além disso, é também um instrumento para raciocinar sobre estes sistemas e suas propriedades. Lógicas modais, que são extensões da lógica clássica com alguns novos operadores, têm sido frequentemente utilizadas com este propósito (HM92; MP92; FHMV95; DLKH96; DFGFvdH07; Bou11; DWFZ12; GL13).

Em lógicas modais, além dos operadores convencionais da lógica clássica, temos os operadores de necessidade e possibilidade, geralmente denotados por "$\Box$" e "$\Diamond$", respectivamente. Tais operadores podem caracterizar e representar diferentes modos de avaliação da verdade de proposições, tais como: crença e conhecimento, usados para representar noções epistêmicas sobre informações que caracterizam o estado de um sistema; temporais, utilizados para representar o comportamento do sistema com o passar do tempo; deônticos, onde os operadores descrevem modos como obrigação e permissão; entre outros (Che80). Com o auxílio de lógicas modais é possível descrever propriedades tais como: "é possível que esteja chovendo lá fora", "acredita-se que esteja chovendo lá fora", "sabe-se que está chovendo lá fora", "amanhã vai chover", "eu sei que vai chover todo o mês de janeiro", "ninguém acredita que irá chover em agosto", etc. Pode-se falar, portanto, já que fala sobre a *possibilidade* de um fato ser ou não verdadeiro, uma *modalidade* qualifica a verdade do julgamento. Neste trabalho, serão consideradas combinações específicas de lógicas epistêmicas e temporais, que são apresentadas a seguir.

Ambas as lógicas epistêmicas que serão aqui tratadas, $KD45_{(n)}$ e $S5_{(n)}$, buscam caraterizar uma forma de "conhecimento" de um "agente" sobre "fatos". Apesar de serem sintaticamente iguais, a noção semântica de cada lógica é diferente. $S5_{(n)}$ é conhecida com a lógica do conhecimento *idealizado* (FHMV95), devido a sua característica reflexiva e introspectiva, determinando que o conhecimento de um fato requer a sua realização no mundo atual. Já $KD45_{(n)}$ é conhecida como a lógica da crença *idealizada* (HM92), uma vez que o conhecimento dos agentes pode não corresponder ao que é verdadeiro no mundo atual. Estes dois operadores permitem, portanto, que se possa descrever tanto o que o agente sabe quanto o que o agente acredita.

A lógica temporal com que iremos trabalhar é a proposicional linear ($PLTL$): os modelos são discretos e compostos por linhas de tempo com passado finito e futuro infinito (GPSS80). Seu caráter linear deve-se ao fato de para cada instante do tempo existe somente um futuro possível. Com o auxílio desta linguagem podemos raciocinar sobre fatos no presente, no momento futuro e sobre a ocorrência de fatos em intervalos de tempo.

A combinação da lógica temporal $PLTL$ e de conhecimento $S5_{(n)}$, denominada $KL_{(n)}$ (FHMV95), tem se mostrado adequada para a especificação de sistemas complexos e foi, por exemplo, utilizada para descrever sistemas distribuídos (GL13), multiagentes (Bou11) e protocolos (DFGFvdH07). A combinação de $PLTL$ com a lógica de crença $KD45_{(n)}$ é denominada $BL_{(n)}$. Lógicas epistêmicas consideradas isoladamente têm caráter estático. A fusão de lógicas temporais e epistêmicas é interessante porque, ao juntar o aspecto temporal, ganhamos uma componente dinâmica que é inerente aos sistemas complexos, distribuídos e concorrentes que desejamos formalizar e sobre os quais desejamos raciocinar.

O problema de verificação, que é uma das aplicações de raciocínio lógico a partir de uma formalização, pode ser definido como se segue. Se Δ é o conjunto de fórmulas que representa a especificação e Γ é o conjunto de fórmulas que caracteriza uma propriedade, a verificação consiste em mostrar que a relação $\Delta \models \Gamma$ é satisfeita, ou seja, que o conjunto Γ é consequência lógica de Δ. Este problema pode ser resolvido com o auxílio de ferramentas semânticas, como, verificadores de modelo e SAT-*solvers*, por exemplo, ou com o auxílio de ferramentas que implementem métodos de prova completos para a lógica considerada. Dado um método de prova correto e completo para uma determinada lógica, o problema de consequência lógica e o problema de dedutibilidade são redutíveis entre si. Assim, provadores automáticos de teoremas podem ser utilizados para prover resposta ao problema de verificação.

Provadores de teoremas são ferramentas computacionais que automatizam a aplicação dos métodos de prova e são variadas as técnicas em que são baseados. Por exemplo, um provador automático de teoremas pode basear-se em resolução (Rob65), tableaux (Smu95), sequentes (Sistemas de Gentzen), dedução natural (Gen69), entre outros. Neste trabalho iremos trabalhar com o método baseado em tableaux para $KL_{(n)}$ e $BL_{(n)}$ apresentado em (WDF98). O método do tableaux é baseado em prova por contradição em que regras de inferência são aplicadas a um conjunto de fórmulas. O método é também analítico, ou seja, aplicações das regras decompõem fórmulas em suas subfórmulas (ou equivalentes) até que não seja mais possível aplicar alguma regra. Caso seja encontrada uma contradição, o conjunto de fórmulas não pode ser satisfeito. Provadores baseados em tableaux são bastante utilizados devido à sua simplicidade e fácil implementação, sendo frequentemente usados para lidar com o problema de dedutibilidade em lógicas não-clássicas, como a lógica modal (DGHP99).

A construção de provadores de teoremas baseados em tableaux pode ser realizada tendo em mente uma lógica específica ou um grupo de lógicas. O método apresentado em (WDF98), com o qual iremos trabalhar, lida especificamente com duas lógicas, $KL_{(n)}$ e $BL_{(n)}$. Dentre os provadores baseados em tableaux para lógicas específicas, podemos citar o *MOLTAP* (vL14) e *OOPS* (vVvdVV10), que lidam com problemas descritos na lógica epistêmica $S5_{(n)}$. Provadores lógicos genéricos como *Isabelle* (Pau90) provêm mecanismos metalógicos que permitem a codificação de métodos de prova incluindo o tableaux. Entre essas duas abordagens encontramos ferramentas que implementam métodos para determinados grupos de lógicas e que também permitem que o usuário, a partir de uma especificação, crie seu próprio provador. Exemplos de tais ferramentas são: MLTP (Li08), LoTREC (FdCFG$^+$01) e MetTeL^2 (TSK13). É interessante mencionar que o provador LoTREC utiliza uma estrutura de grafo como representação do tableaux (CFdCGH97), assim como em (WDF98). Um pouco mais próximo da abordagem feita pelo *Isabelle* temos o *Tableau WorkBench* (TWB) (TAB03), que é uma ferramenta genérica e intuitiva

para construção de provadores de teorema baseados em tableaux para lógicas (modais) proposicionais. Nele, os usuários definem as regras do tableaux e especificam as estratégias que guiam a busca da prova. Um dos principais objetivos do TWB é dar aos usuários que não possuem conhecimento em programação a possibilidade de construir provadores de teoremas para suas lógicas. Uma listagem, não exaustiva, de diversos provadores de teorema pode ser encontrada em (Sch14).

O método apresentado em (WDF98) é específico e, portanto, otimizado para lidar com o problema de satisfatibilidade em $KL_{(n)}$ e $BL_{(n)}$. Um dos objetivos deste trabalho foi exatamente a implementação deste método de prova, para o qual, até onde vai nosso conhecimento, não existia implementação . A nossa implementação do método foi escrita na linguagem de programação Python (Pyt14b) e está publicamente disponível[1] para utilização e testes. O outro objetivo deste trabalho é aprimorar a aplicação deste método de prova para o fim específico de verificação de propriedades. Para atingir este objetivo, apresentamos um algoritmo para verificação de propriedades que reaproveita a estrutura já construída pelo método em (WDF98) para um conjunto de propriedades que represente uma especificação. O algoritmo proposto, que é correto, procura evitar a busca exaustiva na estrutura dada e minimizar expansões desta estrutura. É objetivo futuro demonstrar a completude do método aqui apresentado, bem como sua implementação.

O trabalho está organizado da seguinte forma. O Capítulo 2 introduz a sintaxe e semântica das lógicas multimodais de conhecimento $S5_{(n)}$ e de crença $KD45_{(n)}$, bem como da lógica temporal $PLTL$ e das lógicas resultantes da combinação (fusão) entre a lógica multimodal epistêmica e temporal ($S5_{(n)} + PLTL$) e a lógica multimodal de crença de temporal ($KD45_{(n)} + PLTL$).

No Capítulo 3 será apresentado o método de prova baseado em tableaux para $S5_{(n)} + PLTL$ e $KD45_{(n)} + PLTL$ que inspirou esta dissertação (WDF98) e para o qual foi realizada a implementação, descrita no Capítulo 4. O método é correto e completo. Se a entrada for um conjunto inconsistente de fórmulas, a estrutura resultante da aplicação do algoritmo é vazia. Caso contrário, um modelo pode ser construído a partir da estrutura produzida pelo método. No Capítulo 4 é descrita a arquitetura e tecnologias utilizadas na implementação do tableau. A implementação foi construída visando a facilidade de extensão para outras lógicas e a capacidade de exportação das estruturas do tableau já construídas.

O Capítulo 5 apresenta a proposta de modificação ao método de prova baseado em tableaux, juntamente com as provas de correção e completude. Vários exemplos são apresentados para ilustrar a aplicação do método. A modificação proposta procura fazer o reaproveitamento da estrutura já construída pelo método apresentado em (WDF98) e minimizar expansões na construção desta estrutura, o que pode melhorar a eficiência na obtenção de respostas para o problema de verificação.

No Capítulo 6 são apresentadas as considerações finais e os trabalhos futuros.

[1] `https://github.com/tcvieira/tableau`.

Capítulo 2

Lógicas Temporais de Conhecimento e Crença

Diferentes aspectos dos sistemas que desejamos representar podem ser descritos através de diferentes lógicas modais e suas combinações. Uma lógica modal é, em geral, uma extensão da lógica clássica onde operadores que representam os *modos de verdade* são adicionados. Neste capítulo apresentaremos três importantes lógicas modais, que têm sido utilizadas na descrição de sistemas computacionais complexos (FHMV95; DFGFvdH07). Na Seção 2.1, serão descritas as lógicas epistêmicas de conhecimento e de crença, que permitem falar de aspectos idealizados da percepção de agentes sobre o mundo. Na Seção 2.2, apresentaremos a lógica temporal linear, que permite representar fatos que ocorrem em um certo momento ou em períodos do tempo. Por último, a Seção 2.3 apresenta as duas combinações entre as lógicas epistêmicas e temporal apresentadas.

2.1 Lógicas de Conhecimento e Crença

A lógica modal epistêmica permite expressar noções de conhecimento de maneira natural. Esta lógica consiste em um arcabouço formal baseado em um conjunto de axiomas com o qual podemos caracterizar propriedades epistêmicas. Dentro desse sistema temos fórmulas que podem expressar fatos como "1+1=2", "Alice sabe que 1+1=2"e "Alice sabe que sabe que 1+1=2". Esta lógica é reconhecida por caracterizar uma noção idealizada de conhecimento, em que o agente tem pleno conhecimento sobre o seu conhecimento (introspecção).

Outro aspecto importante é que podemos combinar vários sistemas modais que representam conhecimento, um para cada agente envolvido, e assim raciocinar sobre o conhecimento de outros agentes. Com isso podemos expressar fatos como, por exemplo, "Alice sabe que Bob sabe o seu próprio nome" e "Bob sabe que Alice sabe o seu próprio nome".

Nos sistemas aqui apresentados, serão introduzidos operadores modais na forma "K_i", onde i é o nome de um agente. Cada operador "K_i" é uma abreviação mnemônica para a expressão "*i knows*", o que nos remete à noção de conhecimento. Por exemplo, $\mathrm{K}_1\,\varphi$ significa que "o agente 1 sabe φ", onde φ pode ser qualquer fato como, por exemplo, "João ganhou na loteria". Este sistema é bastante estudado e utilizado como uma caracterização idealizada de conhecimento para descrever sistemas multiagente, distribuídos e programas baseados em conhecimento (FHMV95).

Além de conhecimento, iremos também apresentar sistemas multimodais que lidam com crença. A diferença básica entre crença e conhecimento é que este último exige que o fato sabido seja real, enquanto a crença em algo não o exige. Não serão adotados operadores distintos para conhecimento e crença, uma vez que não trataremos da combinação entre estas lógicas, mas entre cada uma delas com a lógica temporal que será apresentada na Seção 2.2. A sintaxe de ambas as lógicas epistêmicas é portanto idêntica, como apresentado a seguir na Seção 2.1.1. A semânticas destas lógicas modais serão apresentadas na Seção 2.1.2.

2.1.1 Sintaxe

Nesta seção apresentaremos a sintaxe das linguagens $S5_{(n)}$ e $KD45_{(n)}$, os sistemas lógicos que utilizaremos para representação de conhecimento e crença, respectivamente. No que se segue, letras gregas minúsculas representarão fórmulas quaisquer dentro do nosso sistema lógico. Em geral, letras cursivas maiúsculas serão utilizadas para representar conjuntos e letras gregas maiúsculas serão utilizadas para representar conjuntos de fórmulas.

As fórmulas são construídas a partir de um conjunto de *símbolos proposicionais* $\mathcal{P} = \{p, q, p', q', p_1, q_1, \ldots\}$ e um conjunto finito de agentes $\mathcal{A} = \{1, \ldots, n\}$. Além dos operadores proposicionais são introduzidos os novos operadores unários: $\mathrm{K}_1, \ldots, \mathrm{K}_n$. A sintaxe dos sistemas modais de conhecimento e crença é definida por um conjunto de fórmulas bem formadas conforme abaixo:

Definição 1 O conjunto de fórmulas bem formadas de $S5_{(n)}$ (resp. $KD45_{(n)}$), denotado por $FBF_{S5_{(n)}}$ (resp. $FBF_{KD45_{(n)}}$) é definido recursivamente:

- O conjunto $\mathcal{P} = \{p, q, p', q', p_1, q_1, \ldots\} \in FBF_{S5_{(n)}}$ (resp. $FBF_{KD45_{(n)}}$);
- **true**, **false** e **start** estão em $FBF_{S5_{(n)}}$ (resp. $FBF_{KD45_{(n)}}$);
- Se φ e $\psi \in FBF_{S5_{(n)}}$ (resp. $FBF_{KD45_{(n)}}$), então $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $(\varphi \Leftrightarrow \psi)$ e $\mathrm{K}_i\,\varphi$ $(\forall i \in \mathcal{A})$ também estão em $FBF_{S5_{(n)}}$ (resp. $FBF_{KD45_{(n)}}$).

Parênteses são omitidos se a leitura da fórmula não se tornar ambígua e a precedência dos operadores clássicos juntamente com os operadores modais é a usual, partindo da maior para a menor: $\neg, \mathrm{K}_i, \wedge, \vee, \Rightarrow$ e $\Leftrightarrow$.

Definição 2 Um *literal* é um símbolo proposicional ou sua negação.

Definição 3 Um *literal modal* é $\mathrm{K}_i\,l$ ou $\neg\mathrm{K}_i\,l$, onde l é um literal e $i \in \mathcal{A}$.

2.1.2 Semântica

A caracterização semântica de lógicas modais é, em geral, dada por estruturas de Kripke, que têm como base um conjunto $\mathcal{S}$ de *mundos* (ou *estados*), que podem ser entendidos como interpretações clássicas, e um conjunto de relações binárias $\mathcal{K}_i$, $i \in \mathcal{A}$, entre estes mundos, chamadas de *relações de acessibilidade*. A satisfação de uma fórmula depende tanto do mundo em que está sendo interpretada quanto da relação deste mundo com os demais.

Uma relação de acessibilidade entre mundos $\mathcal{K}_n \subseteq \mathcal{S} \times \mathcal{S}$, $i \in \mathcal{A}$, determina quais mundos são *visíveis* ou acessíveis a partir de outros mundos. Sejam $s, s' \in \mathcal{S}$. Se $s\mathcal{K}_i s'$, então dizemos que s' é acessível pelo agente $i \in \mathcal{A}$ a partir de s.

Definição 4 Uma ***estrutura de Kripke*** M para $\mathcal{A} = \{1, \ldots, n\}$ é uma tupla $M = \langle \mathcal{S}, s_0, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n \rangle$, onde $\mathcal{S}$ é um conjunto de mundos possíveis (ou estados), com $s_0 \in \mathcal{S}$; $\pi(s) : \mathcal{P} \to \{true, false\}$, $s \in \mathcal{S}$, é uma interpretação que associa a cada um dos estados em $\mathcal{S}$ uma valoração para seus símbolos proposicionais; e $\mathcal{K}_i$ é uma relação binária sobre $\mathcal{S}$.

A relação binária $\mathcal{K}_i$ corresponde a uma relação de acessibilidade do agente i. Logo, um par (s, t) está em $\mathcal{K}_i$ se o agente i considera possível o mundo t, dada a informação que ele tem no mundo s. Para obter a noção de conhecimento no sistema $S5_{(n)}$, a relação $\mathcal{K}_i$ é uma relação de equivalência sobre $\mathcal{S}$, ou seja, transitiva, simétrica e reflexiva (ou, equivalentemente, reflexiva, transitiva e euclidiana). Sendo assim, mundos que pertencem a mesma classe de equivalência detêm o mesmo conhecimento sobre os fatos. Para obter a noção de crença no sistema $KD45_{(n)}$, a relação é serial, transitiva e euclidiana. As diferentes restrições sobre a relação de acessibilidade permitem a diferenciação de conhecimento e crença: o agente pode acreditar em algo sem que isto seja, de fato, verdadeiro.

A semântica é definida pela relação $\models$, onde $(M, s) \models \varphi$ significa que φ é satisfeita no mundo s da estrutura de Kripke M.

Definição 5 A satisfação das fórmulas é definida conforme abaixo:

- $(M, s) \models$ **true**
- $(M, s) \not\models$ **false**
- $(M, s) \models$ **start** se, e somente se $s = s_0$ (onde s_0 é o mundo inicial)
- $(M, s) \models p$ se, e somente se, $\pi(s)(p) = true$, onde $p \in \mathcal{P}$
- $(M, s) \models \neg\varphi$ se, e somente se, $(M, s) \not\models \varphi$
- $(M, s) \models (\varphi \wedge \psi)$ se, e somente se, $(M, s) \models \varphi$ e $(M, s) \models \psi$
- $(M, s) \models (\varphi \vee \psi)$ se, e somente se, $(M, s) \models \varphi$ ou $(M, s) \models \psi$
- $(M, s) \models (\varphi \Rightarrow \psi)$ se, e somente se, $(M, s) \models \neg\varphi$ ou $(M, s) \models \psi$

- $(M, s) \models (\varphi \Leftrightarrow \psi)$ se, e somente se, $(M, s) \models (\varphi \Rightarrow \psi)$ e $(M, s) \models (\psi \Rightarrow \varphi)$
- $(M, s) \models \mathrm{K}_i\, \varphi$ se, e somente se, para todo t, tal que $(s, t) \in \mathcal{K}_i$, $(M, t) \models \varphi$.

Definição 6 Uma fórmula φ é *satisfatível* se existe um modelo $M = \langle \mathcal{S}, s_0, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n \rangle$ tal que $(M, s_0) \models \varphi$.

Definição 7 Uma fórmula φ é *válida* se para todo modelo $M = \langle \mathcal{S}, s_0, \pi, \mathcal{K}_1, \ldots, \mathcal{K}_n \rangle$ temos que $(M, s_0) \models \varphi$.

2.1.3 Axiomatização

A lógica modal epistêmica apresentada por $S5_{(n)}$ é originalmente nomeada pelos seus axiomas $KTD45_{(n)}$, da mesma forma que ocorre para a lógica $KD45_{(n)}$. A axiomatização para cada uma dessas lógicas inclui, portanto, os axiomas abaixo (FHMV95):

K_i:	$\vdash \mathrm{K}_i\, (\varphi \Rightarrow \psi) \Rightarrow (\mathrm{K}_i\, \varphi \Rightarrow \mathrm{K}_i\, \psi)$
T_i:	$\vdash \mathrm{K}_i\, \varphi \Rightarrow \varphi$
D_i:	$\vdash \mathrm{K}_i\, \varphi \Rightarrow \neg \mathrm{K}_i \neg\, \varphi$
4_i:	$\vdash \mathrm{K}_i\, \varphi \Rightarrow \mathrm{K}_i\, \mathrm{K}_i\, \varphi$
5_i:	$\vdash \neg \mathrm{K}_i \neg\, \varphi \Rightarrow \mathrm{K}_i\, \neg \mathrm{K}_i \neg\, \varphi$

juntamente com as regras de inferência: *modus ponens* (de $\vdash \varphi$ e $\vdash (\varphi \Rightarrow \psi)$ pode-se inferir $\vdash \psi$) e a *regra da necessidade modal* (de $\vdash \varphi$ pode-se inferir $\vdash \mathrm{K}_i\, \varphi$).

Observando os axiomas para $S5_{(n)}$ podemos perceber o porquê essa lógica é conhecida como de *conhecimento idealizado*. Um agente sabe todas as consequências das coisas que sabe (axioma K_i); um agente não pode ser ignorante dos fatos (axioma D_i); um agente não pode saber fatos que não sejam verdadeiros (axioma T_i); um agente sabe o que ele sabe (i.e. introspecção positiva) (axioma 4_i); e um agente sabe o que ele não sabe (i.e. introspecção negativa) (axioma 5_i). Como os aspectos de raciocínio perfeito, dado pelo axioma K_i, e de introspecção, dados pelos axiomas 4_i e 5_i, também caracterizam $KD45_{(n)}$, esta é considerada uma lógica de *crença idealizada*.

2.2 Lógica Temporal

A lógica temporal permite representar fatos como, por exemplo, "no próximo instante de tempo estará chovendo lá fora". Além disso, também podemos representar fatos que sejam verdadeiros em todos os próximos momentos, durante um intervalo ou em algum instante no futuro.

Lógica temporal tem sido usada para especificar e verificar sistemas concorrentes, distribuídos e protocolos ((MP92; DLKH96; DFGFvdH07; Bou11; DWFZ12; GL13)). Exis-

tem vários tipos de lógicas temporais que variam quanto à linguagem (e.g. quantificada ou não) e propriedades do modelo (linear, ramificada, discreta, densa, etc).

A lógica que iremos trabalhar é a temporal proposicional linear ($PLTL$): os modelos são discretos e compostos por linhas de tempo com passado finito e futuro infinito (GPSS80). Seu caráter linear deve-se ao fato de para cada instante do tempo existe somente um futuro possível.

Temos alguns novos operadores que caraterizam os modos introduzidos pela lógica, que são: "$\Diamond$" (em algum momento no futuro), "$\Box$" (em todos os momentos no futuro), "$\bigcirc$" (no próximo momento), "$\mathcal{U}$" (até que) e "$\mathcal{W}$" (a menos que).

2.2.1 Sintaxe

Definição 8 Uma fórmula no sistema $PLTL$ é construída a partir dos seguintes conectivos e símbolos proposicionais:

- um conjunto enumerável $\mathcal{P} = \{p, q, p', q', p_1, q_1, \ldots\}$ de símbolos proposicionais;
- constantes: **true**, **false**, **start**;
- conectivos proposicionais: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ e $\Leftrightarrow$;
- conectivos temporais: $\Diamond$, $\Box$, $\bigcirc$, $\mathcal{U}$ e $\mathcal{W}$;

Definição 9 O *conjunto de fórmulas bem-formadas* do sistema $PLTL$, denotado por FBF_{PLTL}, é definido recursivamente como:

- qualquer elemento de $\mathcal{P}$ está em FBF_{PLTL};
- **true**, **false** e **start** estão em FBF_{PLTL};
- Se φ e ψ estão em FBF_{PLTL}, então $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $(\varphi \Leftrightarrow \psi)$, $\Diamond\varphi$, $\Box\varphi$, $\bigcirc\varphi$, $(\varphi\,\mathcal{U}\,\psi)$, $(\varphi\,\mathcal{W}\,\psi)$ também estão em FBF_{PLTL}.

Nós poderíamos usar somente $\neg$, $\wedge$, $\bigcirc$ e $\mathcal{U}$ como operadores primitivos, ou seja, com os outros operadores definidos a partir destes pelas seguintes regras de equivalência:

$$\begin{array}{lcl}
\varphi \vee \psi & \equiv & \neg(\neg\varphi \wedge \neg\psi) \\
\varphi \Rightarrow \psi & \equiv & \neg(\varphi \wedge \neg\psi) \\
\varphi \Leftrightarrow \psi & \equiv & \neg(\varphi \wedge \neg\psi) \wedge \neg(\psi \wedge \neg\varphi) \\
\Diamond\varphi & \equiv & \mathbf{true}\,\mathcal{U}\,\varphi \\
\Box\varphi & \equiv & \varphi\,\mathcal{U}\,\mathbf{false} \\
\varphi\,\mathcal{W}\,\psi & \equiv & (\varphi\,\mathcal{U}\,\psi) \vee \Box\varphi
\end{array}$$

Parênteses são omitidos se a leitura da fórmula não se tornar ambígua e a precedência dos operadores clássicos juntamente com os operadores modais é a usual, partindo da maior para a menor: $\neg$, $\Box, \Diamond, \bigcirc, \mathcal{U}, \mathcal{W}, \wedge, \vee, \Rightarrow$ e $\Leftrightarrow$.

2.2.2 Semântica

A interpretação das fórmulas em $PLTL$ é baseada em uma estrutura de Kripke $M = \langle \mathbb{N}, 0, <, \pi \rangle$ discreta e linear, isto é, um modelo pode ser descrito como uma sequência de estados sobre o conjunto dos números naturais, $\mathbb{N}$, onde 0 é o estado inicial. A função de avaliação $\pi : (\mathbb{N} \times \mathcal{P}) \longrightarrow \{true, false\}$ associa a cada estado e símbolo proposicional um valor de verdade. Novamente, a interpretação de uma fórmula φ depende do modelo e do estado em que é avaliada. Denotamos por $(M, i) \models \varphi$ quando fórmula φ é satisfeita em um estado com índice $i \in \mathbb{N}$ do modelo M; caso contrário, $(M, i) \not\models \varphi$.

Definição 10 Sejam $\varphi, \psi \in FBF_{PLTL}$, M um modelo e $i \in \mathbb{N}$:

- $(M, i) \models$ **true**;
- $(M, i) \not\models$ **false**;
- $(M, i) \models$ **start** se, e somente se $i = 0$;
- $(M, i) \models p$ se, e somente se, $\pi(i)(p) = true$, onde $p \in \mathcal{P}$;
- $(M, i) \models \neg\varphi$ se, e somente se, $(M, i) \not\models \varphi$;
- $(M, i) \models (\varphi \wedge \psi)$ se, e somente se, $(M, i) \models \varphi$ e $(M, i) \models \psi$;
- $(M, i) \models (\varphi \vee \psi)$ se, e somente se, $(M, i) \models \varphi$ ou $(M, i) \models \psi$;
- $(M, i) \models (\varphi \Rightarrow \psi)$ se, e somente se, $(M, i) \models \neg\varphi$ ou $(M, i) \models \psi$;
- $(M, i) \models (\varphi \Leftrightarrow \psi)$ se, e somente se, $(M, i) \models (\varphi \Rightarrow \psi)$ e $(M, i) \models (\psi \Rightarrow \varphi)$;
- $(M, i) \models \bigcirc\varphi$ se, e somente se, $(M, i + 1) \models \varphi$;
- $(M, i) \models \Diamond\varphi$ se, e somente se, $\exists k, k \in \mathbb{N}, k \geq i, (M, k) \models \varphi$;
- $(M, i) \models \Box\varphi$ se, e somente se, $\forall k, k \in \mathbb{N}$, se $k \geq i$, então $(M, k) \models \varphi$;
- $(M, i) \models \varphi \mathcal{U} \psi$ se, e somente se, $\exists k, k \in \mathbb{N}, k \geq i, (M, k) \models \psi$ e $\forall j, j \in \mathbb{N}$, se $i \leq j < k$, então $(M, j) \models \varphi$;
- $(M, i) \models \varphi \mathcal{W} \psi$ se, e somente se, $(M, i) \models \varphi \mathcal{U} \psi$ ou $(M, i) \models \Box\varphi$.

As fórmulas são interpretadas no estado inicial, 0. Uma fórmula φ é dita *satisfatível* se $(M, 0) \models \varphi$, para algum modelo M; e caso isso seja verdade para todo modelo M, então φ é uma fórmula *válida*, denotado por $\models \varphi$.

2.2.3 Axiomatização

A axiomatização de $PLTL$ é dada pelos seguintes axiomas:

1. Todas as tautologias da lógica proposicional
2. $\Box(\varphi \Rightarrow \psi) \Rightarrow (\Box\varphi \Rightarrow \Box\psi)$
3. $\bigcirc\neg\varphi \Leftrightarrow \neg\bigcirc\varphi$
4. $\bigcirc(\varphi \Rightarrow \psi) \Rightarrow (\bigcirc\varphi \Rightarrow \bigcirc\psi)$
5. $\Box\varphi \Rightarrow \varphi \wedge \bigcirc\Box\varphi$
6. $\Box(\varphi \Rightarrow \bigcirc\varphi) \Rightarrow (\varphi \Rightarrow \Box\varphi)$
7. $(\varphi \,\mathcal{U}\, \psi) \Rightarrow \Diamond\psi$
8. $(\varphi \,\mathcal{U}\, \psi) \Leftrightarrow (\psi \vee (\varphi \wedge \bigcirc(\varphi \,\mathcal{U}\, \psi)))$.

juntamente com as regras de inferência: *modus ponens* (de $\vdash \varphi$ e $\vdash (\varphi \Rightarrow \psi)$ pode-se inferir $\vdash \psi$) e *generalização* (de φ pode-se inferir $\Box\varphi$).

2.3 Lógicas Temporais de Conhecimento e Crença

Nesta seção serão introduzidas as lógicas temporais de conhecimento, $KL_{(n)}$, e crença, $BL_{(n)}$, que consistem basicamente da fusão das duas lógicas epistêmicas já apresentadas com a lógica temporal linear. Ou seja, $KL_{(n)}$ é dada pela fusão de $S5_{(n)}$ e $PLTL$, enquanto $BL_{(n)}$ corresponde à fusão de $KD45_{(n)}$ e $PLTL$. A fusão corresponde à união dos sistemas axiomáticos das respectivas lógicas. Suas aplicações vão desde especificação de sistemas distribuídos e multiagentes (FHMV95; GL13) a protocolos de segurança (DFGFvdH07).

2.3.1 Sintaxe

Nesta seção apresentaremos as regras de formação das fórmulas no sistema $KL_{(n)}$ e $BL_{(n)}$. Como apresentado anteriormente, temos que o conjunto de agentes é definido como $\mathcal{A} = \{1, \ldots, n\}$.

Definição 11 Uma fórmula nos sistemas $KL_{(n)}$ e $BL_{(n)}$ é construída a partir dos seguintes conectivos e símbolos proposicionais:

- um conjunto enumerável $\mathcal{P} = \{p, q, p', q', p_1, q_1, \ldots\}$ de símbolos proposicionais;
- constantes: **true**, **false** e **start**;
- conectivos proposicionais: $\neg$, $\wedge$, $\vee$, $\Rightarrow$ e $\Leftrightarrow$;
- conectivos temporais: $\Diamond$, $\Box$, $\bigcirc$, $\mathcal{U}$ e $\mathcal{W}$;
- um conjunto de operadores modais unários $\mathrm{K}_1, \ldots, \mathrm{K}_n$, onde $n \in \mathcal{A}$.

Definição 12 O *conjunto de fórmulas bem-formadas* do sistema $KL_{(n)}$ e $BL_{(n)}$, denotado por FBF, é definido recursivamente como:

- qualquer elemento do conjunto de símbolos proposicionais $\mathcal{P}$ está em FBF;
- **true**, **false** e **start** estão em FBF;
- Se φ e ψ estão em FBF, então $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \Rightarrow \psi)$, $(\varphi \Leftrightarrow \psi)$, $\diamondsuit\varphi$, $\Box\varphi$, $\bigcirc\varphi$, $(\varphi\,\mathcal{U}\,\psi)$, $(\varphi\,\mathcal{W}\,\psi)$ e $\mathrm{K}_i\,\varphi$ $(i \in \mathcal{A})$ também estão em FBF.

2.3.2 Semântica

Para a caracterização de conhecimento e crença, as relações de acessibilidade são denotadas por $\mathcal{K}_i$, com $i \in \mathcal{A}$, ou seja, indexadas pelo índice de cada agente. Como usual, estas são relações de *equivalência*, ou seja, transitivas, simétricas e reflexivas para $KL_{(n)}$. No caso de $BL_{(n)}$, as relações são euclidianas, seriais e transitivas. Para a caracterização do tempo, a relação de acessibilidade é uma ordem total sobre os mundos. As seguintes definições são necessárias para a definição da caracterização semântica das combinações de lógicas.

Definição 13 Uma *linha do tempo* t é uma sequência discreta, infinitamente longa e linear de estados/mundos, os quais são indexados por números naturais. Definimos $TLinhas$ como o conjunto de todas as linhas do tempo.

Definição 14 Um *ponto* q é um par $q = (t, u)$, onde $t \in TLinhas$ é uma linha do tempo e $u \in \mathbb{N}$ é um índice temporal para t. Definimos $Pontos$ como o conjunto de todos os pontos.

Definição 15 Uma *valoração* π é uma função $\pi : Pontos \times \mathcal{P} \longrightarrow \{true, false\}$

Dadas estas definições, podemos agora caracterizar formalmente a noção de modelo para as lógicas $KL_{(n)}$ e $BL_{(n)}$:

Definição 16 Um *modelo* para $KL_{(n)}$ (resp. $BL_{(n)}$) é uma estrutura $M = \langle TL, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi\rangle$ onde:

- $TL \subseteq TLinhas$ é um conjunto de linhas do tempo com uma linha distinta, t_0;

- $\mathcal{K}_i$, para todo $i \in A$, é a relação de acessibilidade sobre os pontos do modelo, tal que $\mathcal{K}_i \subseteq \mathit{Pontos} \times \mathit{Pontos}$, onde cada $\mathcal{K}_i$ é uma relação de equivalência (resp. serial, transitiva e euclidiana);
- π é uma valoração (conforme Definição 15).

A satisfação de uma fórmula em um ponto (t, u) de um modelo M é definida da seguinte forma:

Definição 17 Uma fórmula é satisfeita em um modelo $M = \langle TL, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi \rangle$ em um determinado ponto (t, u) se e somente se:

- $\langle M, (t, u) \rangle \models \mathbf{true}$;
- $\langle M, (t, u) \rangle \not\models \mathbf{false}$;
- $\langle M, (t, u) \rangle \models \mathbf{start}$, se, e somente se, $t = t_0$ e $u = 0$;
- $\langle M, (t, u) \rangle \models p$ se, e somente se, $\pi(t, u)(p) = \mathit{true}$, onde $p \in \mathcal{P}$;
- $\langle M, (t, u) \rangle \models \neg\varphi$ se, e somente se, $\langle M, (t, u) \rangle \not\models \varphi$;
- $\langle M, (t, u) \rangle \models (\varphi \wedge \psi)$ se, e somente se, $\langle M, (t, u) \rangle \models \varphi$ e $\langle M, (t, u) \rangle \models \psi$;
- $\langle M, (t, u) \rangle \models (\varphi \vee \psi)$ se, e somente se, $\langle M, (t, u) \rangle \models \varphi$ ou $\langle M, (t, u) \rangle \models \psi$;
- $\langle M, (t, u) \rangle \models (\varphi \Rightarrow \psi)$ se, e somente se, $\langle M, (t, u) \rangle \models \neg\varphi$ ou $\langle M, (t, u) \rangle \models \psi$;
- $\langle M, (t, u) \rangle \models (\varphi \Leftrightarrow \psi)$ se, e somente se, $\langle M, (t, u) \rangle \models (\varphi \Rightarrow \psi)$ e $\langle M, (t, u) \rangle \models (\psi \Rightarrow \varphi)$;
- $\langle M, (t, u) \rangle \models \bigcirc\varphi$ se, e somente se, $\langle M, (t, u + 1) \rangle \models \varphi$;
- $\langle M, (t, u) \rangle \models \diamondsuit\varphi$ se, e somente se, $\exists k$, $k \in \mathbb{N}$, $k \geq u$, $\langle M, (t, k) \rangle \models \varphi$;
- $\langle M, (t, u) \rangle \models \Box\varphi$ se, e somente se, $\forall k$, $k \in \mathbb{N}$, se $k \geq u$, então $\langle M, (t, k) \rangle \models \varphi$;
- $\langle M, (t, u) \rangle \models \varphi \mathcal{U} \psi$ se, e somente se, $\exists k$, $k \in \mathbb{N}$, $k \geq u$, $\langle M, (t, k) \rangle \models \psi$ e $\forall j$, $j \in \mathbb{N}$, se $u \leq j < k$, então $\langle M, (t, j) \rangle \models \varphi$;
- $\langle M, (t, u) \rangle \models \varphi \mathcal{W} \psi$ se, e somente se, $\langle M, (t, u) \rangle \models \varphi \mathcal{U} \psi$ ou $\langle M, (t, u) \rangle \models \Box\varphi$;
- $\langle M, (t, u) \rangle \models \mathrm{K}_i \varphi$ se, e somente se, $\forall (t', u')$, tal que $(t, u) \mathcal{K}_i (t', u')$, temos que $\langle M, (t', u') \rangle \models \varphi$.

A semântica dos operadores também pode ser apresentada de maneira gráfica como mostrado nas Figuras 2.1 e 2.2. Na Figura 2.1, a primeira linha representa a satisfatibilidade de uma fórmula da forma $\diamondsuit\varphi$ no instante inicial. Note que a fórmula φ será satisfeita em algum momento posterior do tempo, mais precisamente, no quarto instante de tempo. Na segunda linha, a figura mostra que uma fórmula da forma $\bigcirc\varphi$ é satisfeita no instante inicial, se φ é satisfeita no próximo momento. Para fórmulas da forma $\Box\varphi$

temos que, em todos os instantes do tempo, φ é satisfeita. Por último, $\varphi \mathcal{U} \psi$ mostra φ sendo satisfeita até que ψ também seja.

A Figura 2.2 define uma classe de equivalência para o agente i que engloba duas linhas do tempo. Na figura, foram suprimidas setas correspondentes à relação $\mathcal{K}_i$ e os mundos pertencentes a tal relação pertencem à área delimitada pela linha pontilhada. Como φ é satisfeita em todos os mundos pertencentes à classe de equivalência, o agente i não é capaz de distinguir, com base em seu conhecimento de φ, entre estes mundos.

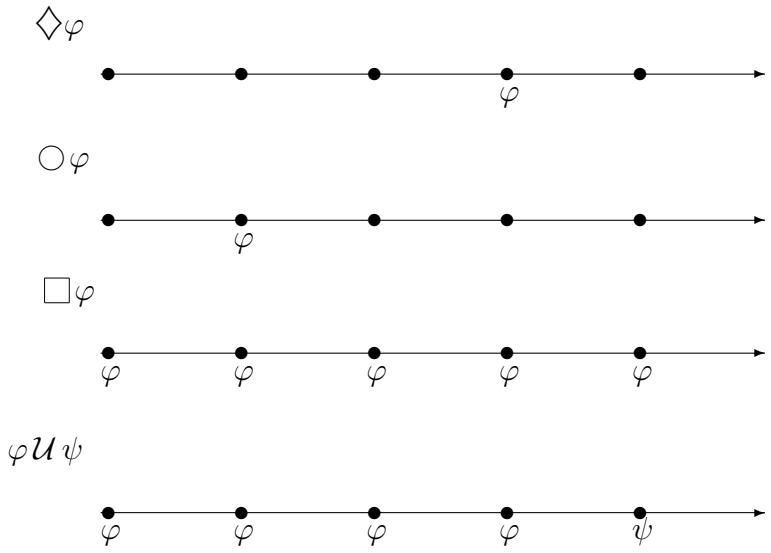

Figura 2.1: Operadores temporais

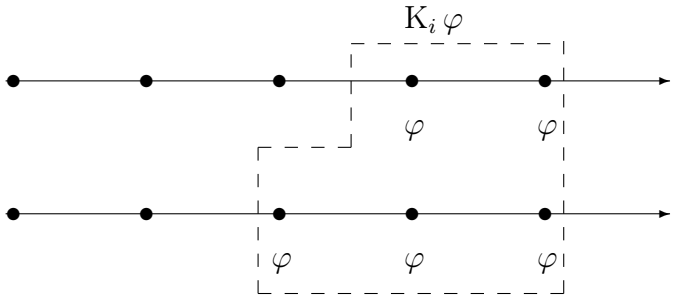

Figura 2.2: Operador de conhecimento

Seja $M = \langle TL, \mathcal{K}_1, \ldots, \mathcal{K}_n, \pi \rangle$ um modelo. Nós dizemos que uma fórmula φ é *satisfatível*, se existe M tal que $\langle M, (t_0, 0) \rangle \models \varphi$. Denotamos por $M \models \varphi$ se M satisfaz φ. Se não existe um modelo que satisfaça uma fórmula φ, dizemos que φ é *insatisfatível*. Uma fórmula φ é *válida*, denotado por $\models \varphi$, se $\langle M, (t_0, 0) \rangle \models \varphi$, para todo modelo M. Vale ressaltar que validade e satisfatibilidade são definidas em relação ao primeiro momento no tempo, isto é, na linha do tempo t_0 no instante 0. Observamos que o problema de satisfatibilidade de uma fórmula em $KL_{(n)}$ e $BL_{(n)}$ é PSPACE-completo (FHMV95), ou seja, está na mesma classe de complexidade dos problemas de satisfatibilidade para as linguagens componentes, $S5_{(n)}$ $(n \geq 2)$, $KD45_{(n)}$ e $PLTL$, que também são PSPACE-completos.

Definição 18 Seja $\Delta = \{\varphi_1, \ldots, \varphi_m\} \subseteq FBF$, $m \in \mathbb{N}$, um conjunto de fórmulas. Dizemos que Δ é *satisfatível* (ou *consistente*) se, e somente se, $\varphi_1 \wedge \ldots \wedge \varphi_m$ é satisfatível. O fato de um modelo M satisfazer Δ é denotado por $M \models \Delta$.

Um conjunto de fórmulas que não é consistente é dito *inconsistente* ou *insatisfatível*.

Definição 19 Seja $\Delta = \{\varphi_1, \ldots, \varphi_m\} \subseteq FBF$, $m \in \mathbb{N}$, um conjunto de fórmulas e $\varphi \in FBF$ uma fórmula. Nós dizemos que φ é *consequência lógica* de Δ se, e somente se, para todo modelo M temos que

$$\text{se } M \models \Delta \text{ então } M \models \varphi.$$

A partir das Definições 18 e 19 podemos mostrar que se uma fórmula φ é consequência lógica de um conjunto de fórmulas $\Delta = \{\varphi_1, \ldots, \varphi_m\}$, então $\varphi_1 \wedge \ldots \wedge \varphi_m \wedge \neg\varphi$ é insatisfatível. É nesta propriedade que se baseiam métodos de prova baseados em contradição e, em particular, o método de prova baseado em tableaux para as lógicas temporais epistêmicas, apresentado no próximo capítulo.

Capítulo 3

Tableau para Lógicas Temporais de Conhecimento e Crença

3.1 Introdução

Neste capítulo será introduzido o método de prova baseado em tableaux para as lógicas $KL_{(n)}$ e $BL_{(n)}$ proposto em (WDF98). Dada uma fórmula satisfatível φ, este método propõe um algoritmo para a construção de uma estrutura da qual um modelo para φ pode ser extraído. Para mostrar que uma dada fórmula φ é válida, o algoritmo é aplicado a $\neg\varphi$. Caso um modelo não seja encontrado, $\neg\varphi$ é insatisfatível e φ é válida.

O método busca sistematicamente por uma estrutura que permita a construção de um modelo para a fórmula dada. Como as lógicas $KL_{(n)}$ e $BL_{(n)}$ possuem mais de uma dimensão, uma temporal e outra epistêmica, a busca é realizada em ambas. A construção desta estrutura, chamada tableau, é feita intercalando a geração de sucessores temporais e epistêmicos para cada um dos estados do modelo. Uma vez que todos os sucessores tenham sido gerados, o procedimento remove estados que não podem fazer parte de um modelo, seja porque o estado é inconsistente (i.e. contém uma fórmula e sua negação) ou porque fórmulas da forma $\diamondsuit\varphi$ ou $\neg \mathrm{K}_i\,\varphi$ não podem ser satisfeitas. Se a estrutura resultante não for vazia e a fórmula para qual testamos a satisfatibilidade pertence a um dos estados iniciais desta estrutura, então esta fórmula é satisfeita na estrutura e um modelo para a fórmula pode ser construído.

Vamos apresentar algumas definições, começando pelos tipos de fórmulas:

Definição 20

1. Se φ é da forma $\mathrm{K}_i\,\psi$ ou $\neg\mathrm{K}_i\,\psi$, então φ é uma fórmula epistêmica;
2. Se φ é um símbolo proposicional ou a negação de um símbolo proposicional, então φ é um literal;
3. Se φ é uma fórmula epistêmica ou um literal, então φ é um átomo;
4. Se φ é da forma $\bigcirc\psi$, então φ é uma fórmula futura.

Seja Δ um conjunto de fórmulas de $KL_{(n)}$ ou $BL_{(n)}$. Define-se $next(\Delta)$ como:

$$next(\Delta) = \{\varphi \mid \bigcirc\varphi \in \Delta\}$$

O método é aplicado a fórmulas na Forma Normal Negada (BEL$^+$01), ou seja, em fórmulas onde negações aplicam-se somente a símbolos proposicionais. As regras para mover as negações para dentro de fórmulas modais, remover operadores e a regra de eliminação da dupla-negação são apresentadas na Tabela 3.1.

Fórmula	Fórmula com a negação deslocada
$\neg\neg\varphi$	φ
$\neg(\varphi \vee \psi)$	$(\neg\varphi \wedge \neg\psi)$
$\neg(\varphi \wedge \psi)$	$(\neg\varphi \vee \neg\psi)$
$\neg(\varphi \Rightarrow \psi)$	$(\varphi \wedge \neg\psi)$
$\neg \Box\varphi$	$\Diamond\neg\varphi$
$\neg\Diamond\varphi$	$\Box\neg\varphi$
$\neg\bigcirc\varphi$	$\bigcirc\neg\varphi$
$\neg(\varphi \mathcal{W} \psi)$	$\neg\psi \mathcal{U} (\neg\varphi \wedge \neg\psi)$
$\neg(\varphi \mathcal{U} \psi)$	$\neg\psi \mathcal{W} (\neg\varphi \wedge \neg\psi)$

Tabela 3.1: Equivalências para negações

Como todo método de decisão baseado em tableaux (Smu95), temos dois tipos de regras para serem aplicadas às fórmulas: α e β. Regras α são aplicadas a fórmulas que são equivalentes a fórmulas cujo operador principal são conjunções (fórmulas conjuntivas ou fórmulas do tipo α). As regras β, por sua vez, são aplicadas a fórmulas que são equivalentes a fórmulas cujo operador principal são disjunções (fórmulas disjuntivas ou fórmulas do tipo β). As regras devem ser entendidas da seguinte forma. Se Δ é um conjunto de fórmulas e φ é uma fórmula do tipo α, então ambas as componentes α_1 e α_2 devem ser adicionadas a Δ. Se Δ é um conjunto de fórmulas e φ é uma fórmula do tipo β, então pelo menos uma de suas componentes β_1 ou β_2 deve ser adicionadas a Δ[1]. Na Tabela 3.2 estão definidas as regras α e β que são usadas em ambos os métodos, tanto para $KL_{(n)}$ quanto para $BL_{(n)}$. No método para $KL_{(n)}$ temos uma regra α a mais, como mostra a Tabela 3.3. A aplicação desta regra α específica para $KL_{(n)}$ corresponde à reflexividade da relação de acessibilidade desta lógica. Cada regra é aplicada uma única vez a cada fórmula, para evitar problemas com terminação do algoritmo.

α	α_1	α_2
$\varphi_1 \wedge \varphi_2$	φ_1	φ_2
$\Box\varphi$	φ	$\bigcirc \Box\varphi$

β	β_1	β_2
$\varphi_1 \vee \varphi_2$	φ_1	φ_2
$\Diamond\varphi$	φ	$\neg\varphi \wedge \bigcirc\Diamond\varphi$
$\varphi \mathcal{U} \psi$	ψ	$\neg\psi \wedge \varphi \wedge \bigcirc(\varphi \mathcal{U} \psi)$
$\varphi \mathcal{W} \psi$	ψ	$\neg\psi \wedge \varphi \wedge \bigcirc(\varphi \mathcal{W} \psi)$

Tabela 3.2: Regras α e β para $KL_{(n)}$ e $BL_{(n)}$

[1]Como veremos mais adiante, a aplicação da regra β leva, de fato, à duplicação do conjunto Δ, de forma a obter $\Delta' = \Delta \cup \{\beta_1\}$ e $\Delta'' = \Delta \cup \{\beta_2\}$, sendo Δ descartado.

α	α_1	α_2
$\mathrm{K}_i\,\varphi$	φ	$\mathrm{K}_i\,\varphi$

Tabela 3.3: Regra α para $KL_{(n)}$

Apresentaremos primeiramente as definições relacionadas à construção de um *tableau proposicional completo* (PC-tableaux), o qual envolve basicamente dois grandes passos que são aplicados conjunta e exaustivamente: (1) a construção do tableau proposicional, o que corresponde à aplicação exaustiva das regras α e β; e (2) a inclusão, nos conjuntos gerados, de determinadas (sub)fórmulas relacionadas aos operadores epistêmicos.

Definição 21 Um tableau proposicional é um conjunto de fórmulas em que não é mais possível aplicar regras α e β.

Apresentamos a seguir a definição do conjunto de subfórmulas de uma fórmula.

Definição 22 Seja φ um fórmula de $KL_{(n)}$ ou $BL_{(n)}$, dizemos que $sub(\varphi)$ é o conjunto de todas as subfórmulas de φ:

$$sub(\varphi) = \begin{cases} \{\varphi\} & \text{se } \varphi \in \mathcal{P} \text{ ou } \varphi = \textbf{true} \text{ ou } \varphi = \textbf{false} \\ \{\neg\psi\} \cup sub(\psi) & \text{se } \varphi = \neg\psi \\ \{\psi \wedge \chi\} \cup sub(\psi) \cup sub(\chi) & \text{se } \varphi = \psi \wedge \chi \\ \{\psi \vee \chi\} \cup sub(\psi) \cup sub(\chi) & \text{se } \varphi = \psi \vee \chi \\ \{\mathrm{K}_i\,\psi\} \cup sub(\psi) & \text{se } \varphi = \mathrm{K}_i\,\psi \\ \{\bigcirc\psi\} \cup sub(\psi) & \text{se } \varphi = \bigcirc\psi \\ \{\,\Box\psi\} \cup sub(\psi) & \text{se } \varphi = \,\Box\psi \\ \{\psi\,\mathcal{U}\,\chi\} \cup sub(\psi) \cup sub(\chi) & \text{se } \varphi = \psi\,\mathcal{U}\,\chi \\ \{\psi\,\mathcal{W}\,\chi\} \cup sub(\psi) \cup sub(\chi) & \text{se } \varphi = \psi\,\mathcal{W}\,\chi \end{cases}$$

Definição 23 Um conjunto de fórmulas Δ é dito *subfórmula completo* se, e somente se, para toda fórmula $\mathrm{K}_i\,\varphi \in \Delta$ e para todo $\mathrm{K}_i\,\psi \in sub(\varphi)$, temos que $\mathrm{K}_i\,\psi \in \Delta$ ou $\neg\mathrm{K}_i\,\psi \in \Delta$.

Definição 24 Um tableau proposicional subfórmula completo (PC-tableau) é um conjunto de fórmulas Δ que é um tableau proposicional e subfórmula completo.

Uma forma de eliminar estados durante a construção do tableau é verificando se eles são próprios (ou consistentes), conforme Definição 25.

Definição 25 Seja Δ um conjunto de fórmulas, dizemos que Δ é *próprio*, denotado por $proper(\Delta)$ se, e somente se:
1. **false** $\notin \Delta$
2. Se $\varphi \in \Delta$, então $\neg\varphi \notin \Delta$

Um conjunto Δ de fórmulas que não é próprio é chamado de *impróprio*, denotado por $\neg proper(\Delta)$. Obviamente, conjuntos impróprios são insatisfatíveis (WDF98).

3.2 Construção do conjunto de PC-tableaux

O algoritmo abaixo, dados pelos Passos (1) a (3), apresenta o procedimento para a construção dos PC-tableaux a partir de um conjunto de fórmulas Δ. Primeiramente, aplicam-se os Passos (1) e (2) à $\mathcal{F} = \{\Delta\}$ até que não seja mais possível aplicá-los. Em seguida, aplica-se o Passo (3) para a remoção dos conjuntos de fórmulas que não sejam próprios. O resultado é um conjunto de PC-tableaux. Esta construção corresponde a um dos passos do algoritmo dado à Seção 3.3, onde a construção do tableau para $KL_{(n)}$ e $BL_{(n)}$ é detalhada.

Algoritmo 1 *Construção dos tableaux proposicionais completos e próprios a partir de um conjunto $\mathcal{F}$ de conjuntos de fórmulas.*

1. Criando um tableau proposicional
   Para qualquer $\Delta' \in \mathcal{F}$, tal que $proper(\Delta')$, escolhemos uma fórmula $\varphi \in \Delta'$ a qual não tenham sido aplicadas as regras α ou β. Se φ é uma fórmula α com componentes α_1 e α_2, substituímos Δ' por $\Delta' \cup \{\alpha_1 \cup \alpha_2\}$. Se φ é uma fórmula β com componentes β_1 e β_2 então $\mathcal{F}$ será dado por:

$$\mathcal{F} = \mathcal{F} - \Delta' \cup \{\Delta' \cup \{\beta_1\}\} \cup \{\Delta' \cup \{\beta_2\}\}$$

2. Criando um tableau subfórmula completo
   Para qualquer $\Delta' \in \mathcal{F}$, tal que $proper(\Delta')$, se $K_i\,\varphi \in \Delta'$ e $K_i\,\psi \in sub(\varphi)$ tal que, $K_i\,\psi \notin \Delta'$ e $\neg K_i\,\psi \notin \Delta'$ então $\mathcal{F}$ será:

$$\mathcal{F} = \mathcal{F} - \Delta' \cup \{\Delta' \cup \{K_i\,\psi\}\} \cup \{\Delta' \cup \{\neg K_i\,\psi\}\}$$

3. Deletando conjuntos impróprios
   Deletar qualquer $\Delta' \in \mathcal{F}$, tal que $\neg proper(\Delta')$.

Uma das estratégias sugeridas em (WDF98) para a construção dos PC-tableaux é aplicar as regras α primeiro, em seguida as regras β e em seguida inserir as subfórmulas e suas negações onde for necessário. Esta estratégia leva, na maioria dos casos, a um conjunto menor de fórmulas. O número de conjuntos obtidos ao final da aplicação do algoritmo é, no pior caso, na ordem do número de subconjuntos *próprios* de $sub(\varphi)$.

Vale ressaltar que se o conjunto de fórmulas do PC-tableaux para $\{\varphi\}$ é vazio, então φ é insatisfatível. A prova pode ser vista em (WDF98).

3.3 Tableau para $KL_{(n)}$ e $BL_{(n)}$

O tableau para $KL_{(n)}$ e $BL_{(n)}$ gera uma estrutura próxima a de um modelo de Kripke. É a partir desta estrutura, em forma de grafo, que o modelo pode ser extraído. No que se segue, $\wp(\Delta)$ denota o conjunto-potência de um conjunto Δ. Seja $Estados = \{s, s', \ldots\}$ um conjunto de estados.

Definição 26 Uma estrutura, H, é uma tupla $H = (S, \eta, R_1, \ldots, R_n, L)$, onde:

- $S \subseteq Estados$ é um conjunto de estados;
- $\eta \subseteq S \times S$ é a relação binária temporal sobre S;
- $R_i \subseteq S \times S$ representa a relação de acessibilidade sobre S para o agente $i \in A$;
- $L : S \rightarrow \wp(FBF)$ rotula cada estado com um conjunto de fórmulas bem-formadas de $KL_{(n)}$ e $BL_{(n)}$.

Definição 27 Seja $\eta \subseteq S \times S$ a relação binária temporal sobre S. Definimos η^* como sendo o fecho reflexivo e transitivo da relação η.

Definição 28 Se $\varphi \in FBF$ é da forma $\chi \mathcal{U} \psi$ ou $\diamondsuit \psi$, então dizemos que φ possui a *eventualidade* ψ. Seja $H = (S, \eta, R_1, \ldots, R_n, L)$ uma estrutura, $s \in S$ um estado e $\varphi \in FBF$. Dizemos que φ é *resolvível* em H a partir de s, denotado por $resolvable(\varphi, s, H)$, se, e somente se, se φ possui uma eventualidade ψ, então existe $s' \in S$ tal que $(s, s') \in \eta^*$ e $\psi \in L(s')$.

3.3.1 Algoritmo

A construção da estrutura $(S, \eta, R_1, \ldots, R_n, L)$ para uma dada fórmula $\varphi_0 \in FBF$ consiste de duas fases. A primeira é a construção sistemática de estados, rotulados por conjuntos de fórmulas que correspondem a PC-Tableaux próprios, e de arestas que correspondem às relações temporais e epistêmicas. A segunda é de contração, pela eliminação de estados que possuem eventualidades ou fórmulas epistêmicas que não podem ser satisfeitas na estrutura.

Algoritmo 2 *Construção de* $(S, \eta, R_1, \ldots, R_n, L)$ *para uma fórmula* $\varphi_0 \in FBF$.

1. Inicialização
Defina os seguintes valores iniciais:

$$S = \eta = R_1 = \ldots = R_n = L = \emptyset$$

Construa $\mathcal{F}$, o conjunto dos PC-tableaux para $\{\varphi_0\}$. Para cada $\Delta_i \in \mathcal{F}$, crie um novo estado s_i e faça $L(s_i) = \Delta_i$ e $S = S \cup \{s_i\}$. Para cada $\Delta_i \in \mathcal{F}$, repita os Passos (2)-(3), até que nenhum deles possa mais ser aplicado, então aplique o Passo (4).

2. Criando os sucessores R_i
   Para qualquer estado s rotulado pelas fórmulas $L(s)$, onde $L(s)$ é próprio e um PC-tableau, para cada fórmula da forma $\neg K_i \psi \in L(s)$ crie o conjunto de fórmulas:

$$\Delta = \{\neg\psi\} \cup \{\chi \mid K_i \chi \in L(s)\} \cup \{K_i \chi \mid K_i \chi \in L(s)\} \cup \{\neg K_i \chi \mid \neg K_i \chi \in L(s)\}$$

   Se s não possui fórmula da forma $\neg K_i \psi$, mas existe $K_i \psi \in L(s)$, então construa o seguinte conjunto de fórmulas:

$$\Delta = \{\chi \mid K_i \chi \in L(s)\} \cup \{K_i \chi \mid K_i \chi \in L(s)\}$$

   Para cada novo Δ gerado, construa o conjunto $\mathcal{F}$ de PC-tableaux para Δ. Para cada membro $\Delta' \in \mathcal{F}$, se $\exists s'' \in S$ tal que $\Delta' = L(s'')$ então inclua (s, s'') em R_i. Caso contrário adicione um novo estado s' a S, rotule $L(s') = \Delta'$ e inclua (s, s') em R_i.

3. Criando sucessores η
   Para qualquer estado s rotulado pelas fórmulas $L(s)$, onde $L(s)$ é próprio e um PC-tableau, se $\bigcirc\psi \in L(s)$ crie um conjunto $\Delta = next(L(s))$. Para cada novo Δ gerado, construa seu PC-tableau $\mathcal{F}$. Para cada membro $\Delta' \in \mathcal{F}$, se $\exists s'' \in S$ tal que $\Delta' = L(s'')$ então inclua (s, s'') a η. Caso contrário adicione um novo estado s' em S, rotulando $L(s') = \Delta'$ e inclua (s, s') a η.

4. Contração
   Continue removendo qualquer estado s, onde:

   (a) $\exists\psi \in L(s)$ tal que $\neg resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$; ou
   (b) $\exists\psi \in L(s)$ tal que ψ é da forma $\bigcirc\psi$ e $\nexists s' \in S$ tal que $(s, s') \in \eta$; ou
   (c) $\exists\psi \in L(s)$ tal que ψ é da forma $\neg K_i \chi$ e $\nexists s' \in S$ tal que $(s, s') \in R_i$ e $\neg\chi \in L(s')$; ou
   (d) $\exists\psi \in L(s)$ tal que ψ é da forma $K_i \chi$ e $\nexists s' \in S$ tal que $(s, s') \in R_i$ e $\chi \in L(s')$

   Até que não seja mais possível fazer remoções.

É importante ressaltar que os passos de expansão e contração não podem ser intercalados um com o outro.

O algoritmo do tableau é dito *bem-sucedido* para uma fórmula φ_0 se, e somente se, a estrutura obtida contém um estado s em que $\varphi_0 \in L(s)$. Se o tableau para φ_0 for bem-sucedido, então φ_0 é satisfatível. Caso contrário, φ_0 é insatisfatível.

A prova de correção para este algoritmo, bem como o procedimento para extração do modelo a partir da estrutura, que corresponde à prova de sua completude, podem ser encontrados em (WDF98). Na prova de correção, mostra-se que se uma fórmula é satisfatível, então a estrutura para ela criada é um tableau, conforme definição a seguir.

Definição 29 Seja $\varphi_0 \in FBF$, e $H = (S, \eta, R_1, \ldots, R_n, L)$ um estrutura, então H é um *tableau* para φ_0 se, e somente se:

1. $\exists s \in S$ tal que $\varphi_0 \in L(s)$;

e, para todo $s \in S$ temos que:

2. $L(s)$ é um conjunto *próprio*;
3. Se $\psi \in L(s)$ e ψ é um fórmula do tipo α com componentes α_1 e α_2, então $\alpha_1 \in L(s)$ e $\alpha_2 \in L(s)$;
4. Se $\psi \in L(s)$ e ψ é um fórmula do tipo β com componentes β_1 e β_2, então $\beta_1 \in L(s)$ ou $\beta_2 \in L(s)$;
5. Se $\mathrm{K}_i\,\psi \in L(s)$ ou $\neg\mathrm{K}_i\,\psi \in L(s)$ e $\mathrm{K}_i\,\varphi \in sub(\psi)$, então $\mathrm{K}_i\,\varphi \in L(s)$ ou $\neg\mathrm{K}_i\,\varphi \in L(s)$;
6. Se $\mathrm{K}_i\,\psi \in L(s)$ então $\forall s' \in S$, se $(s, s') \in R_i$, então $\psi \in L(s')$;
7. Se $\neg\mathrm{K}_i\,\psi \in L(s)$ então $\exists s' \in S$ tal que $(s, s') \in R_i$ e $\neg\psi \in L(s')$;
8. Se em $KL_{(n)}$
   (a) Se $\mathrm{K}_i\,\psi \in L(s)$ então $\psi \in L(s)$;
   (b) Para todo s', se $(s, s') \in R_i$, então $\mathrm{K}_i\,\psi \in L(s')$ se, e somente se, $\mathrm{K}_i\,\psi \in L(s')$;
9. Se em $BL_{(n)}$:
   (a) Se $(s, s'), (s, s'') \in R_i$ e $\mathrm{K}_i\,\psi \in L(s')$, então $\{\mathrm{K}_i\,\psi, \psi\} \in L(s'')$;
   (b) Se $\mathrm{K}_i\,\psi \in L(s)$, então $\exists s' \in S$ tal que $(s, s') \in R_i$;
   (c) Se $\mathrm{K}_i\,\psi \in L(s)$ e $(s, s') \in R_i$, então $\mathrm{K}_i\,\psi \in L(s')$.
10. Se $\bigcirc\psi \in L(s)$, então $\exists s' \in S$ tal que $(s, s') \in \eta$;
11. Se $\bigcirc\psi \in L(s)$, então $\forall s' \in S$, se $(s, s') \in \eta$, então $\psi \in L(s')$;
12. Se $\psi \in L(s)$, então $resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$.

A prova de completude, dada em (WDF98), garante que a partir de um tableau para uma fórmula satisfatível, um modelo pode ser gerado. Iremos falar um pouco mais sobre a prova de correção quando formos apresentar nossa proposta de verificação.

3.4 Exemplo

Nesta seção será apresentada a aplicação do método baseado em tableau para uma fórmula em $KL_{(2)}$. Mostremos, em detalhe, a construção da estrutura H para a fórmula:

$$(\mathrm{K}_1\ \Box p \wedge \mathrm{K}_2\ \Box p) \Rightarrow\ \Box p \tag{3.1}$$

cuja Forma Normal Negada é:

$$(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p \tag{3.2}$$

Seja $\Delta = \{(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p\}$ o conjunto inicial de fórmulas. Começaremos a construção do conjunto de PC-Tableaux a partir de $\mathcal{F} = \{\Delta\}$. Aplicando a regra β a Δ, obtemos $\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p$ e $\Box p$, resultando em dois conjuntos:

$$\Delta_0 = \{(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p, \neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p\} \tag{3.3}$$

$$\Delta_1 = \{(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p, \ \Box p\} \tag{3.4}$$

Aplicando a regra α a $\Box p$ em Δ_1, obtemos:

$$\Delta_1 = \{(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p, \ \Box p, \bigcirc \Box p, p\} \tag{3.5}$$

Aplicando a regra β em Δ_0 sobre a fórmula $\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p$, obtemos os conjuntos:

$$\Delta_2 = \{(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p, \neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p, \neg \mathrm{K}_1 \ \Box p\} \tag{3.6}$$

$$\Delta_3 = \{(\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p) \vee \ \Box p, \neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p, \neg \mathrm{K}_2 \ \Box p\} \tag{3.7}$$

e Δ_0 é descartado. Como não existem fórmulas da forma $\mathrm{K}_i \, \psi$ nestes conjuntos e todos os conjuntos obtidos são próprios, o conjunto de PC-tableaux para $\{\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p\}$ resulta em $\mathcal{F} = \{\Delta_1, \Delta_2, \Delta_3\}$. Os seguintes estados são criados: s_0, s_1 e s_2, tais que $L(s_0) = \Delta_1$, $L(s_1) = \Delta_2$ e $L(s_2) = \Delta_3$. Iniciamos, portanto, a criação dos sucessores epistêmicos e temporais para cada um destes estados.

Começando por s_0, a fórmula $\bigcirc \Box p$ em $L(s_0) = \Delta_1$ requer a criação de um sucessor temporal. O PC-tableau para $next(L(s_1)) = \{ \Box p\}$, contém apenas o seguinte conjunto:

$$\Delta_4 = \{ \Box p, \bigcirc \Box p, p\} \tag{3.8}$$

Criamos o estado s_3 com $L(s_3) = \Delta_4$ e o par (s_0, s_3) é adicionado a relação η da estrutura. Observe que o sucessor temporal de s_3, pela aplicação do algoritmo de construção do PC-tableaux a este conjunto, é exatamente igual a Δ_4. Portanto, ao invés de ser criado outro estado, simplesmente adiciona-se o par (s_3, s_3) à relação η. Os estados s_1 e s_2 requerem sucessores epistêmicos por possuírem fórmulas da forma $\neg \mathrm{K}_i \, \psi$. O conjunto de PC-tableaux para o estado s_1 contém dois conjuntos:

$$\Delta_5 = \{\neg \mathrm{K}_1 \ \Box p, \diamondsuit \neg p, \neg p\} \tag{3.9}$$

$$\Delta_6 = \{\neg \mathrm{K}_1 \ \Box p, \diamondsuit \neg p, p \wedge \bigcirc \diamondsuit \neg p, p, \bigcirc \diamondsuit \neg p\} \tag{3.10}$$

Criamos os estados s_4 e s_5 com $L(s_4) = \Delta_5$ e $L(s_5) = \Delta_6$. Os pares (s_0, s_4) e (s_0, s_5) são, então, adicionados à relação R_1. Ao aplicar a regra para os sucessores epistêmicos sobre s_4 e s_5 à fórmula $\neg \mathrm{K}_1 \ \Box p$, obtemos conjuntos de fórmulas que já rotulam estados na estrutura; assim, os pares (s_4, s_5), (s_4, s_4), (s_5, s_4) e (s_5, s_5) são adicionados à relação R_1. Observe que o estado s_5 também possui uma fórmula, $\bigcirc \diamondsuit \neg p$, que exige a criação de um

sucessor temporal. Aplicando o algoritmo da construção do PC-tableaux a $next(L(s_5)) = \{\Diamond p\}$, obtemos os seguintes conjuntos de fórmulas:

$$\Delta_7 = \{\Diamond\neg p, \neg p\} \tag{3.11}$$

$$\Delta_8 = \{\Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \tag{3.12}$$

Criamos os estados s_6 e s_7 com $L(s_6) = \Delta_7$ e $L(s_7) = \Delta_8$. Adicionamos (s_5, s_6) e (s_5, s_7) à relação η. O estado s_7 contém uma fórmula futura. Mas como $next(L(s_7)) = \{\Diamond p\}$, o conjunto de PC-tableaux resultante será o mesmo obtido no passo anterior. Ou seja, só adicionamos à relação η os pares (s_7, s_6) e (s_7, s_7). Falta ainda construir os sucessores epistêmicos de s_2. Aplicando o algoritmo do PC-Tableau a $\{\neg \mathrm{K}_2\ \Box p, \Diamond\neg p\}$, resulta em dois conjuntos:

$$\Delta_9 = \{\neg \mathrm{K}_2\ \Box p, \Diamond\neg p, \neg p\} \tag{3.13}$$

$$\Delta_{10} = \{\neg \mathrm{K}_2\ \Box p, \Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \tag{3.14}$$

Criamos os estados s_8 e s_9 com $L(s_8) = \Delta_9$ e $L(s_9) = \Delta_{10}$. Os pares (s_2, s_8) e (s_2, s_9) são adicionados à relação R_2. Ao aplicar a regra para os sucessores epistêmicos sobre s_8 e s_9, obtemos conjuntos que rotulam estados já existentes na estrutura. Portanto, basta adicionar os pares (s_8, s_9), (s_8, s_8), (s_9, s_8) e (s_9, s_9) à relação R_2. Por fim, os sucessores temporais de s_9 são s_6 e s_7, devido a aplicação da regra para a fórmula $\bigcirc\Diamond\neg p$. Logo, adicionamos à relação η os pares (s_9, s_6) e (s_9, s_7). Ao final da construção da estrutura, obtemos os seguintes conjuntos de fórmulas (onde os conjuntos correspondentes aos estados iniciais são aqueles que possuem a fórmula original):

$$\begin{aligned}
L(s_0) &= \{(\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p) \vee\ \Box p, \neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p, \neg \mathrm{K}_1\ \Box p\} \\
L(s_1) &= \{(\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p) \vee\ \Box p,\ \Box p, \bigcirc\ \Box p, p\} \\
L(s_2) &= \{(\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p) \vee\ \Box p, \neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p, \neg \mathrm{K}_2\ \Box p\} \\
L(s_3) &= \{\ \Box p, \bigcirc\ \Box p, p\} \\
L(s_4) &= \{\neg \mathrm{K}_1\ \Box p, \Diamond\neg p, \neg p\} \\
L(s_5) &= \{\neg \mathrm{K}_1\ \Box p, \Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \\
L(s_6) &= \{\Diamond\neg p, \neg p\} \\
L(s_7) &= \{\Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\} \\
L(s_8) &= \{\neg \mathrm{K}_2\ \Box p, \Diamond\neg p, \neg p\} \\
L(s_9) &= \{\neg \mathrm{K}_2\ \Box p, \Diamond\neg p, p \wedge \bigcirc\Diamond\neg p, p, \bigcirc\Diamond\neg p\}
\end{aligned}$$

e as seguintes relações:

$$R_1 = \{(s_0, s_4), (s_0, s_5), (s_4, s_5), (s_4, s_4), (s_5, s_4), (s_5, s_5)\}$$
$$R_2 = \{(s_2, s_8), (s_2, s_9), (s_8, s_9), (s_8, s_8), (s_9, s_8), (s_9, s_9)\}$$
$$\eta = \{(s_1, s_3), (s_3, s_3), (s_5, s_6), (s_5, s_7), (s_7, s_6), (s_7, s_7), (s_9, s_6), (s_9, s_7)\}$$

A estrutura resultante pode ser vista na Figura 3.1.

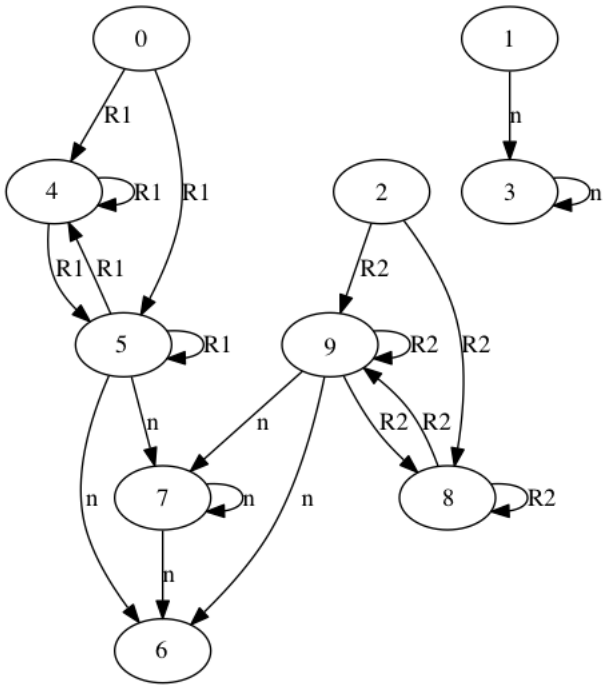

Figura 3.1: Tableau para a Fórmula 3.1

A partir do tableau da Figura 3.1 e utilizando a prova de correção de (WDF98) podemos extrair os modelos que satisfazem a Fórmula 3.1. Observe que os estados iniciais devem corresponder àqueles que contêm à fórmula para qual a estrutura foi construída. Neste caso, possíveis estados iniciais seriam s_0, s_1 e s_2. A Figura 3.2 ilustra um desses modelos em que temos duas linhas temporais: t_0 iniciando a partir do estado s_0 e t_1, iniciando a partir do estado s_5. A linha vermelha no modelo da Figura 3.2 engloba os estados que pertencem à classe de equivalência R_1. Como $\neg \mathrm{K}_i \, \Box p$ é satisfeita em $(t_0, 0)$, a disjunção dada pela Fórmula 3.1 também o é. É possível observar que o conhecimento do agente 1 em s_0 e s_5 é o mesmo, dado que nestes estados temos que $\neg \mathrm{K}_i \, \Box p$. Este modelo possui duas linhas temporais, porém podemos ter mais ou menos linhas dependendo dos estados que selecionamos para extrair o modelo. Por exemplo, se tivéssemos escolhido o estado s_1 como estado inicial, é fácil de ver, a partir da estrutura obtida, que somente uma linha temporal seria suficiente para mostrar a satisfatibilidade da Fórmula 3.1.

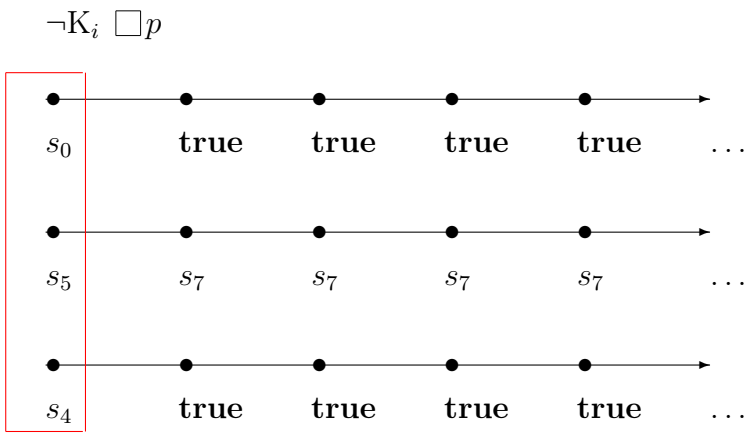

Figura 3.2: Um dos modelos extraídos do tableau da Figura 3.1

Capítulo 4

Implementação

Neste capítulo será apresentada a implementação do tableaux do Capítulo 3. A apresentação será divida em quatro partes nas quais falaremos sobre a linguagem e bibliotecas utilizadas, a arquitetura dos componentes criados para a implementação, como obter/utilizar o programa e alguns testes realizados.

O código fonte da implementação e sua documentação estão disponibilizados em `https://github.com/tcvieira/tableau`.

4.1 Linguagem

A linguagem *Python* foi criada em 1989 por Guido van Rossum (Pyt14b) voltada inicialmente para a criação de *scripts* de administração de sistemas. O nome *Python* é uma homenagem à série britânica de televisão *Monty Python's Flying Circus*.

Python foi desenvolvida no fim do ano de 1989 e sua primeira versão foi disponibilizada em 1991, ou seja, é um linguagem recente. Durante esses anos de existência, a linguagem passou a ser utilizada em diversas áreas, desde *scripts* para administração de sistemas à bioinformática.

Atualmente, o *Python* possui duas versões principais, a 2.7.x e a 3.x. A versão 3 não possui compatibilidade com as versões anteriores da linguagem, porém muitas das melhorias desta nova versão foram implementadas nas versões mais novas da família 2.x. O uso do *Python* é gratuito e seu código fonte é distribuído livremente e licenciado pela *Python license* (compatível com a *GPL*).

O projeto desta linguagem foi guiado por quatro princípios:

1. **Qualidade de Software**: A sintaxe da linguagem facilita o entendimento do código e por conseguinte a qualidade dele. A linguagem foi projetada para ser ortogonal, explícita e minimalista, enquanto suas bibliotecas modulares se encarregam de prover ferramentas mais complexas;

2. **Produtividade**: É uma linguagem ágil, flexível e de alto nível, resultado de ser tipada dinamicamente (*duck typing*) e bastante expressiva, podendo ser utilizada sob diversos paradigmas (orientado a objetos, funcional, procedural);

3. **Portabilidade**: Linguagem independente de plataforma, assim como a maioria de suas bibliotecas;

4. **Integração**: Projeto para possibilidade a integração com outras ferramentas e linguagens.

A linguagem *Python* (Pyt14b) tem ganhado muitos adeptos no meio científico por ser fácil de aprender, portável, interoperável com outras linguagens e possuir diversas bibliotecas voltadas especificamente para a área científica como NumPy (Num14), BioPython (Bio14) e Sage (Sag14).

Na implementação do tableau foram utilizados somente os tipos básicos da linguagem como *strings, listas e dicionários* e a versão 2.7.8. O paradigma utilizado foi o procedural, buscando modularizar o conjunto de funções para facilitar a extensão do programa.

4.2 Dependências

A versão do *Python* utilizada na implementação é a 2.7.8. Além de alguns módulos padrões do *Python*, também foram utilizadas os seguintes módulos:

- **doctopt-0.6.2**: Módulo para criação de CLI (*command line interface*)(Doc14);
- **lrparsing-1.0.7**: Módulo para análise de gramáticas LR (Stu14);
- **networkx-1.9.1**: Módulo com diversas funcionalidades para trabalhar com grafos utilizando tipos básicos do *Python* (Net14);
- **pygraphviz-1.2**: Módulo para geração das imagens dos grafos (PyG14);
- **Sphinx-1.2.3**: Módulo para geração da documentação (Bra14).

Cada um desses módulos possui algumas outras dependências que podem ser vistas em suas documentações.

4.3 Arquitetura

A implementação foi dividida em cinco arquivos de código-fonte, dos quais dois arquivos (*formula_utils.py e utils.py*) fornecem funções auxiliares para o programa como um todo. A Figura 4.1 ilustra os arquivos mencionados.

- **TableauVerifier.py**: Responsável por implementar a CLI do programa e invocar as funções dos outros arquivos;
- **grammar.py**: Responsável por implementar a gramática das lógicas utilizadas;
- **tableau.py**: Responsável por implementar o PC-Tableaux;
- **graph.py**: Responsável por implementar a estrutura de grafo do Tableau e também prover as funções para exportação da estrutura;
- **formula_utils.py e utils.py**: Responsável por prover funções que facilitam a implementação do Tableaux.

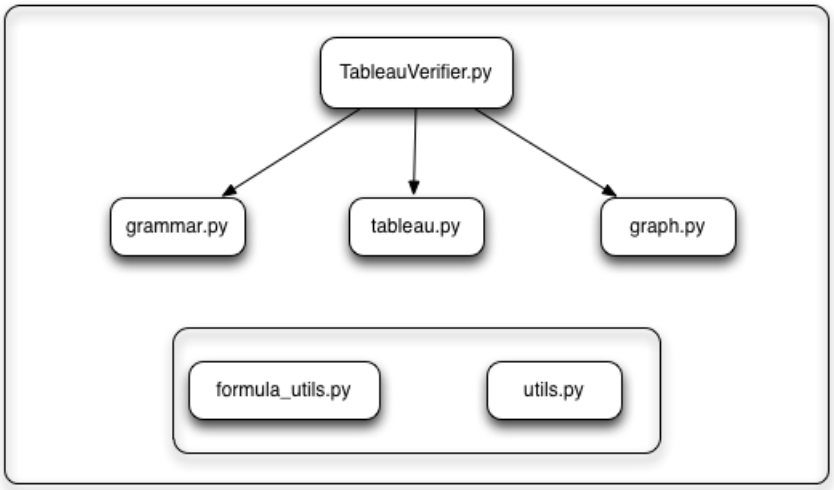

Figura 4.1: Arquitetura da implementação

A implementação possui aproximadamente 1200 linhas de código-fonte. Foi usado o módulo de teste de unidade *unittest* para assegurar o correto funcionamento de algumas funções, principalmente aquelas implementadas em *formula_utils.py*.

A extensão da implementação para lidar com outros tipos de lógicas e regras para o respectivos tableaux é facilitada pela arquitetura adotada. Inicialmente, a implementação abordava apenas as regras para $S5 + PLTL$. Para incluir as regras para $S4 + PLTL$ foi somente necessário realizar algumas modificações para incluir as novas regras em **tableau.py** e **graph.py**.

4.4 Utilização

A implementação disponibiliza três comandos: **parse**, **pctableau** e **tableau**. O comando **parse** realiza a análise sintática, a partir de um arquivo, de uma ou mais cadeias de caracteres que representam as fórmulas. O comando **pctableau** aplica as regras do PC-Tableaux para as fórmulas de um arquivo e retorna os conjuntos obtidos com suas respectivas fórmulas, caso sejam consistentes. O comando **tableau** aplica as regras do algoritmo do tableau, retornando a estrutura e o conjunto de fórmulas para cada um dos estados. Vale observar que, na implementação, bem como na descrição feita neste capítulo, optou-se por identificar estados com seus rótulos.

Abaixo, temos um exemplo da tela de manual do programa.

```
Usage:
  tableauverifier.py parse --file=<formula-file> [--nnf]
  tableauverifier.py pctableau --file=<formula-file> <>[--per-line] [--verbose]
  [--belief]
  tableauverifier.py tableau --file=<formula-file> --prop=<property>
  [--per-line] [--verbose] [--belief]
  tableauverifier.py -h | --help
  tableauverifier.py -v | --version
```

```
  Tableau Verifier for S5+PTL and KD45+PLTL multimodal logics

Options:
    -h --help  Show this screen.
    -v --version  Show version of the program
    --nnf  Return formula in Negation Normal Form
    --per-line  A separated tableau for each formula in the file
    --verbose  Show formulae for each tableaux branch
    --belief  Change to the logic of Belief instead of Knowledge

Arguments:
   <formula-file>  formula file containing formulae to be used
   <property>  property formula to be verified
```

Atualmente, a implementação permite a verificação da satisfatibilidade de fórmulas e de conjuntos de fórmulas, ou seja, como proposto em (WDF98). As modificações para a verificação de propriedades, que serão apresentadas no próximo capítulo, ainda não foram implementadas e constitui trabalho futuro. Todos os exemplos deste trabalho, bem como as figuras dos tableaux, foram geradas pela implementação aqui descrita. Todos os exemplos apresentados em (WDF98) foram testados e foram obtidos os mesmos resultados.

Os conectivos utilizados na ferramenta são definidos pelas expressões regulares abaixo:

- Parenteses = "(" e ")";
- Constantes = "true" e "false";
- Símbolos proposicionais = "[p-s][0-9]*";
- $\wedge$ = "^", o acento circunflexo;
- $\vee$ = "v";
- $\Rightarrow$ = "->";
- $\neg$ = " ~ ", o acento til;
- $\bigcirc$ = "N";
- $\Box$ = "G";
- $\Diamond$ = "F";
- $\mathcal{U}$ = "U";
- $\mathcal{W}$ = "W";
- K_i = "'k[0-9]*"'.

A gramática implementada no programa é apresentada na Figura 4.2.
Alguns exemplos de fórmulas são:

⟨*fórmula*⟩ ::= ⟨*átomo*⟩
  | ‘(’ ⟨*fórmula unária*⟩ ‘)’
  | ‘(’ ⟨*fórmula binária*⟩ ‘)’

⟨*fórmula binária*⟩ ::= ⟨*fórmula*⟩ ⟨*conjunção*⟩ ⟨*fórmula*⟩
  | ⟨*fórmula*⟩ ⟨*disjunção*⟩ ⟨*fórmula*⟩
  | ⟨*fórmula*⟩ ⟨*implicação*⟩ ⟨*fórmula*⟩
  | ⟨*fórmula*⟩ ⟨*até que*⟩ ⟨*fórmula*⟩
  | ⟨*fórmula*⟩ ⟨*ao menos que*⟩ ⟨*fórmula*⟩

⟨*fórmula unária*⟩ ::= ⟨*negação*⟩ ⟨*fórmula*⟩
  | ⟨*conhecimento*⟩ ⟨*fórmula*⟩
  | ⟨*eventualmente*⟩ ⟨*fórmula*⟩
  | ⟨*sempre*⟩ ⟨*fórmula*⟩
  | ⟨*próximo*⟩ ⟨*fórmula*⟩

⟨*átomo*⟩ ::= ⟨*símbolo proposicional*⟩
  | ⟨*constante true*⟩
  | ⟨*constante false*⟩

Figura 4.2: Gramática da implementação

(((k1(Gp0))^(k2(Gp0)))^(k1(~(k2q0)))),
((k1(k2p0))->(k2p0))
(~(G(p999->true)))

Em `https://github.com/tcvieira/tableau`, o usuário pode encontrar um arquivo texto contendo vários exemplos de entrada para que possa testar. Ao executar o programa, é dado como saída algumas informações relacionadas à quantidade de estados, arestas e a listagem das fórmulas para cada estado (seus rótulos). Além disso, o arquivo que contém a imagem do grafo é salvo na pasta “\img” do diretório atual.

Abaixo temos um exemplo de saída ao executar o comando “python tableauverifier.py tableau –file formulae.txt –per-line”, onde o arquivo “formulae.txt” possui a fórmula “((Gp0)^(p2vp1))”.

```
1  ==================================================
2  Name: Tableau Graph
3  Type: MultiDiGraph
4  Number of nodes: 3
5  Number of edges: 3
6  Average in degree:   1.0000
7  Average out degree:   1.0000
8  nodes:
9  0:  [['binary', ['G', ['p0']], '^', [['binary', ['p1'], 'v', ['p2']]]]],
10 ['G', ['p0']], [['binary', ['p1'], 'v', ['p2']]], ['p0'], ['N', ['G', ['p0']]],
11  ['p2'],
12 1:  [['binary', ['G', ['p0']], '^', [['binary', ['p1'], 'v', ['p2']]]]],
```

```
13 ['G', ['p0']], [['binary', ['p1'], 'v', ['p2']]], ['p0'], ['N', ['G', ['p0']]],
14 ['p1'],
15 2:  ['G', ['p0']], ['p0'], ['N', ['G', ['p0']]],
16 building graph... graph built took 0.025064 seconds
17 overall time... overall time (including prints) took 0.039956 seconds
```

A linha 1 mostra o nome padrão para oa tableaux gerados. A linha 2 mostra o tipo de estrutura usada pelo módulo *NetworkX* para representar o tableaux, neste caso temos um "MultiDiGraph" que é um grafo dirigido com múltiplas arestas. Nas linhas de 4 a 7 temos informações sobre a estrutura como a quantidade de nós, quantidade de arestas e grau médio de entrada/saída de arestas para o nós, respectivamente. Nas linhas 9, 12 e 15 temos respectivamente o conjunto de fórmulas para os nós *0*, *1* e *2*. As fórmulas são representadas pela sua estrutura em lista aninhadas, tal como é utilizada pelo programa. As fórmulas binárias possuem o rótulo *binary* para facilitar a sua identificação. As duas últimas linhas, 16 e 17, mostram respectivamente o tempo gasto para a construção do tableau e o tempo gasto para executar o programa como um todo.

Maiores informações sobre a implementação podem ser vistas na documentação disponibilizada em `https://github.com/tcvieira/tableau`, a qual foi criada automaticamente pela ferramenta *Sphinx* (Bra14), a partir da documentação do código.

Capítulo 5

Algoritmo para Verificação de Propriedades

5.1 Introdução

Neste capítulo será discutido como utilizar o método de prova baseado em tableaux apresentado no Capítulo 3 para resolver o problema de verificação de uma propriedade a partir de uma especificação, que é definido da seguinte maneira:

Definição 30 Sejam φ uma fórmula, representando uma propriedade, e Δ um conjunto de fórmulas, representando uma especificação. A verificação de φ, dado Δ, consiste em estabelecer se a relação $\Delta \models \varphi$ é satisfeita, ou seja, se φ é consequência lógica de Δ.

Para resolver o problema de verificação, queremos portanto determinar se todo modelo que satisfaz Δ também satisfaz φ. Observamos, primeiramente, que este problema é distinto do problema de verificação de modelos, que é definido como a determinação se uma dada fórmula é satisfeita em um modelo específico e dado. O problema que desejamos tratar consiste em dizer se φ é satisfeita em *todos* os modelos que satisfazem a especificação, Δ. Segundo, o método baseado em tableaux para $KL_{(n)}$ e $BL_{(n)}$ não retorna um modelo específico, mas a sua prova de completude garante que dada uma fórmula consistente φ, um modelo para φ pode ser construído a partir do tableau para essa fórmula. Embora as provas de correção e completude do cálculo sejam fracas (i.e. as provas garantem que $\models \varphi$ se, e somente se, $\vdash \varphi$), correção forte deriva facilmente a partir de resultados clássicos: considerando a definição de consequência lógica local, como dada pela Definição 19, e satisfatibilidade definida em relação ao mundo inicial, temos que $\Delta = \{\varphi_1, \ldots, \varphi_m\} \models \varphi$ se, e somente se, $\not\models \varphi_1 \wedge \ldots \wedge \varphi_m \wedge \neg\varphi$; ou seja, $\varphi_1 \wedge \ldots \wedge \varphi_m \wedge \neg\varphi$ é insatisfatível (Fit06). Pelos resultados em (WDF98), obteremos um tableau bem-sucedido para $\varphi_1 \wedge \ldots \wedge \varphi_m \wedge \neg\varphi$ se, e somente se, existe um modelo M tal que $M \models \varphi_1 \wedge \ldots \wedge \varphi_m \wedge \neg\varphi$. Logo, se o procedimento de construção do tableau para $\varphi_1 \wedge \ldots \wedge \varphi_m \wedge \neg\varphi$ retornar uma estrutura, podemos mostrar que existe um modelo que satisfaz Δ, mas não satisfaz φ, provendo a resposta para o problema que desejamos resolver.

Foram consideradas três formas distintas de se utilizar o método baseado em tableaux para prover resposta para o problema de verificação de uma propriedade por uma especificação:

1. **Tradicional**: Incluir a negação da propriedade ao conjunto de fórmulas iniciais (especificação) e gerar um tableau para este conjunto;

2. **Novo**: Realizar uma busca na estrutura gerada para a especificação, verificando se a propriedade é satisfeita usando as definições semânticas para cada tipo de fórmula; ou

3. **Otimizado**: Realizar uma busca na estrutura gerada pela especificação, a partir dos estados iniciais e incluindo a negação da propriedade nesta estrutura, realizando expansões e contrações minimamente necessárias.

O método tradicional resolve o problema, porém a construção do tableau para ambos a especificação e cada propriedade a ser testada pode ser pouco eficiente, já que toda vez que quisermos verificar uma propriedade, por mais simples que seja, precisamos também construir um novo tableau também para a especificação (que, em geral, será um conjunto maior de fórmulas e, em geral, não é modificado com muita frequência). É desejável criar um modo de aproveitar o tableau já construído para a especificação e realizar tão somente testes e/ou construções que permitam a verificação da propriedade em questão.

O método apresentado como novo é baseado justamente no aproveitamento do tableau já construído. A estrutura do tableau gerada pelo algoritmo para a especificação pode ser exportada e guardada para futuras verificações, por exemplo. Seja φ uma propriedade e H o tableau para Δ, o algoritmo consiste em realizar uma busca a partir dos estados iniciais a fim de verificar se φ é satisfeita a partir destes. Dependendo da estrutura de φ, suas subfórmulas também são verificadas no estado atual ou a partir dos estados alcançados pelas relações epistêmico-temporais. Se durante a busca todas as subfórmulas de φ forem satisfeitas, então a propriedade é satisfeita neste tableau. Caso contrário, a propriedade não é satisfeita no tableau e a busca é encerrada. Note que neste caso, φ não é negada e o percorrimento procura simplesmente verificar se a fórmula seria satisfeita em modelos que poderiam ser construídos a partir desta estrutura. O método novo é interessante para ser usado na verificação de modelos, em que buscamos verificar se existe um estado em que determinada propriedade é satisfeita.

Embora não requeira novas construções, o método novo não provê uma forma fácil de apresentação de um modelo no caso da propriedade ser satisfeita pela especificação. Ou seja, para a apresentação deste modelo seria necessário novo percorrimento da estrutura e a simulação das expansões e/ou construções necessárias para a correta definição de todos os elementos da estrutura. Por exemplo, suponha que $\Delta = \{p\}$ seja a especificação em $KL_{(n)}$. O tableau resultante é $\{\{s\}, \emptyset, \emptyset\}$, onde $L(s) = \{p\}$. Lembrando que as arestas reflexivas não são construídas pelo algoritmo do tableau, se quisermos verificar a propriedade $\neg \mathrm{K}_1 \neg p$, temos que simular a construção da aresta (s, s) em R_1 para dizermos que a propriedade é de fato satisfeita. Este é um exemplo simples, mas é fácil de ver que fórmulas mais complexas exigiriam esforço praticamente igual ao do método tradicional para a apresentação do modelo.

5.1.1 Método Otimizado

O método otimizado, que é a proposta desta dissertação, é uma tentativa de obter melhores resultados do que o método novo com relação ao problema em que é preciso obter um modelo, mas não queremos construir toda uma nova estrutura, como no método tradicional. Caso φ não seja *imediatamente satisfeita* (conforme definição abaixo) a partir do estado inicial, ao invés de tentar construir um novo tableau usando o método tradicional, este método busca inserir sistematicamente novas fórmulas/conjuntos de fórmulas na estrutura de modo que esta satisfaça a nova fórmula. Esta nova construção, diferentemente do método tradicional, não aplica o conjunto de regras de construção dos PC-Tableaux e o algoritmo do tableau para todo o conjunto de fórmulas (especificação + propriedade), mas somente para a propriedade que é verificada. E, diferentemente do método novo, não se restringe em somente dizer se a propriedade é ou não satisfeita pelo tableau, fornecendo um modelo no caso da propriedade ser satisfeita.

Enquanto o algoritmo do tableau em (WDF98) cria uma estrutura que representa uma abstração para os possíveis modelos para o conjunto inicial de fórmulas, realizando construções e contrações a partir de todos os estados iniciais, o método otimizado termina quando consegue gerar uma (sub)estrutura consistente a partir de um único estado que satisfaça o conjunto inicial de fórmulas e a propriedade. Ou seja, o método otimizado, somente no pior caso, irá verificar se a propriedade é satisfeita em todos os estados iniciais.

Este método segue praticamente o mesmo conjunto de regras para a construção dos PC-Tableaux e para a construção da estrutura, porém leva em consideração que parte da estrutura já está construída. O método é dividido em duas etapas aplicadas recursivamente na estrutura da fórmula da propriedade: inclusão de novas fórmulas nos estados e criação de novos sucessores. Como no algoritmo tradicional, os tipos de fórmulas que levarão à criação de novos estados são $\neg \mathrm{K}_i \varphi$ e $\bigcirc \varphi$. Para a apresentação do algoritmo, precisaremos da seguinte definição.

Definição 31 Dados uma fórmula φ e um estado $s \in S$, dizemos que φ é *imediatamente satisfeita* em s se, e somente se:

1. $\varphi \in L(s)$;
2. se φ é da forma $\neg \mathrm{K}_i \psi$, então $\exists s'$ tal que $(s, s') \in R_i$ e $\neg\psi \in L(s')$;
3. se φ é da forma $\bigcirc \psi$, então $\forall s'$ tal que $(s, s') \in \eta$ é o caso que $\psi \in L(s')$.

Dada uma estrutura $H = (S, \eta, R_1, \ldots, R_n, L)$ para a conjunção das fórmulas em Δ e uma propriedade φ, verificamos se existe $s \in S$ tal que $\neg\varphi$ (em sua Forma Normal Negada) é imediatamente satisfeita em s. Se este for o caso, não há mais nada a fazer e a própria estrutura H é um tableau de onde podemos extrair um modelo que satisfaz Δ e $\neg\varphi$, ou seja, a propriedade não é consequência da especificação. Caso contrário, é preciso caminhar recursivamente sobre a estrutura H incluindo fórmulas e realizando expansão onde necessário. Mais especificamente, se s é um estado inicial do tableau H (um estado que satisfaz todas as fórmulas em Δ), aplicamos a construção dada pelo algoritmo abaixo a s e a $\{\neg\varphi\}$.

Algoritmo 3 *Expansão de um Tableau H por um conjunto de fórmulas Ω a partir de um estado s.*

1. Adicionando Ω a um estado s

   Seja $\Gamma = L(s)$. Aplique o Algoritmo 1 a $\Omega \setminus \Gamma$, obtendo um conjunto de PC-tableaux $\mathcal{F} = \{\Gamma_1, \ldots, \Gamma_m\}$. Se $\mathcal{F} = \emptyset$, faça $L(s) = L(s) \cup \{\mathbf{false}\}$. Senão, para cada $\Gamma_j \in \mathcal{F}$, faça $\mathcal{F}' = \{\Gamma'_j \mid \Gamma_j \cup \Gamma\}$. Faça $S = S \setminus \{s\}$ e para cada $\Gamma'_j \in \mathcal{F}'$ tal que proper(Γ'_j) crie um estado s_j, faça $S = S \cup \{s_j\}$ e modifique adequadamente as relações η e R_i, ou seja:

   - *para todo $(s, s') \in \eta$, faça $\eta = (\eta \setminus \{(s, s')\}) \cup \{(s'_j, s')\}$;*
   - *para todo $(s', s) \in \eta$, faça $\eta = (\eta \setminus \{(s', s)\}) \cup \{(s', s'_j)\}$;*
   - *para todo $(s, s') \in R_i$, para todo $i \in \mathcal{A}$, faça $R_i = (R_i \setminus \{(s, s')\}) \cup \{(s'_j, s')\}$;*
   - *para todo $(s', s) \in R_i$, para todo $i \in \mathcal{A}$, faça $R_i = (R_i \setminus \{(s', s)\}) \cup \{(s', s'_j)\}$;*

2. Criando os sucessores R_i

   (a) *Escolha um estado s'_j criado Passo (1) ou para todos os estados s'_j criados nos Passos(2) e (3), para cada fórmula da forma $\neg K_i \psi \in L(s'_j)$, mas $\neg K_i \psi \notin L(s)$ (onde s é o estado original do tableau a partir do qual s'_j foi construído), se não existe $s'' \in S$ tal que $(s'_j, s'') \in R_i$ e $\neg K_i \psi$ seja imediatamente satisfeita, crie*

   $$\Gamma = \{\neg\psi\} \cup \{\chi \mid K_i \chi \in L(s'_j)\} \cup \{K_i \chi \mid K_i \chi \in L(s'_j)\} \cup \{\neg K_i \chi \mid \neg K_i \chi \in L(s'_j)\}$$

   e aplique o Algoritmo 1 a $\{\Gamma\}$, obtendo um conjunto de PC-tableaux $\mathcal{F}$. Para cada estado $s''' \in \mathcal{F}$, inclua $(s'_j, s''') \in R_i$.

   (b) *Para toda fórmula da forma $K_i \psi \in L(s'_j)$, mas $K_i \psi \notin L(s)$, para todos os estados s'' tal que $(s'_j, s'') \in R_i$, aplique o Passo (1) a s'' e $\{K_i \psi, \psi \mid K_i \psi \in L(s'_j)\}$. Se não existir $s'' \in S$ tal que $(s'_j, s) \in R_i$, crie*

   $$\Delta = \{\chi \mid K_i \chi \in L(s'_j)\} \cup \{K_i \chi \mid K_i \chi \in L(s'_j)\}$$

   e aplique o Algoritmo 1 a $\{\Gamma\}$, obtendo um conjunto de PC-tableaux $\mathcal{F}$. Para cada estado $s''' \in \mathcal{F}$, inclua $(s'_j, s''') \in R_i$.

3. Criando sucessores η

   Escolha um estado s'_j criado no Passo (1) ou para todos os estados s'_j criados nos Passos(2) e (3), inclua cada uma das fórmulas em next$(L(s'_j) \setminus L(s))$ em todo s'' tal que $(s'_j, s'') \in \eta$, usando o procedimento descrito no Passo (1). Se não existir s'' tal que $(s'_j, s'') \in \eta$, aplique o Algoritmo 1 a next$(L(s'_j))$, obtendo um conjunto de PC-tableaux $\mathcal{F}$. Para cada estado $s''' \in \mathcal{F}$, inclua $(s'_j, s''') \in \eta$.

4. Contração
   Enquanto possível, remova os estados s tais que:

   (a) $\exists \psi \in L(s)$ *tal que* $\neg resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$*; ou*

   (b) $\exists \psi \in L(s)$ *tal que* ψ *é da forma* $\bigcirc \psi$ *e* $\nexists s' \in S$ *tal que* $(s, s') \in \eta$*; ou*

   (c) $\exists \psi \in L(s)$ *tal que* ψ *é da forma* $\neg K_i \chi$ *e* $\nexists s' \in S$ *tal que* $(s, s') \in R_i$ *e* $\neg \chi \in L(s')$*; ou*

   (d) $\exists \psi \in L(s)$ *tal que* ψ *é da forma* $K_i \chi$ *e* $\nexists s' \in S$ *tal que* $(s, s') \in R_i$ *e* $\chi \in L(s')$ *(Somente quando estiver trabalhando com* $BL_{(n)}$*).*

Assim como no algoritmo original, os passos de expansão (1, 2 e 3) devem ser repetidos até que não seja mais possível aplicá-los. Da mesma forma, expansão e contração não podem ser intercalados.

O algoritmo do tableau proposto é dito *bem-sucedido* para uma propriedade φ se, e somente se, a estrutura obtida contém pelo menos um estado inicial s tal que $\Delta \cup \{\varphi\} \in L(s)$. É importante observar que este algoritmo é não determinístico: na escolha do estado inicial e dos ramos que serão expandidos. A implementação determinística requer que, caso os estados escolhidos levem a uma estrutura que não seja bem-sucedida, seja feito *backtracking* para uma nova escolha. No pior caso, o algoritmo modificado irá realizar os mesmos passos do algoritmo tradicional.

5.1.2 Exemplos

Nesta seção vamos mostrar alguns exemplos em $KL_{(n)}$ para a verificação de propriedades utilizando o método apresentado. No que se segue, iremos identificar conjuntos de fórmulas a partir da função que rotula estados.

Exemplo 1

Dado o tableau para $\mathrm{K}_1 \ \Box p \wedge \mathrm{K}_2 \ \Box p$ vamos mostrar que a propriedade $\Box p$ também pode ser satisfeita sem precisar construir um novo tableau. Vale lembrar que no Exemplo 1 do Capítulo 3 construímos um tableau para $(\mathrm{K}_1 \ \Box p \wedge \mathrm{K}_2 \ \Box p) \Rightarrow \Box p$ e que, portanto, esta fórmula é satisfatível.

Neste exemplo iremos mostrar como verificar a fórmula $\Box p$ a partir da estrutura para $\mathrm{K}_1 \ \Box p \wedge \mathrm{K}_2 \ \Box p$, cuja estrutura resultante da aplicação do algoritmo do tableaux, H_1, possui os seguintes estados:

$$\begin{array}{lcl} s_0 & = & \{\mathrm{K}_1 \ \Box p \wedge \mathrm{K}_2 \ \Box p, \mathrm{K}_1 \ \Box p, \mathrm{K}_2 \ \Box p, \ \Box p, \bigcirc \Box p, p\} \\ s_1 & = & \{\mathrm{K}_1 \ \Box p, \ \Box p, \bigcirc \Box p, p\} \\ s_2 & = & \{\mathrm{K}_1 \ \Box p, \ \Box p, \bigcirc \Box p, p\} \\ s_3 & = & \{\ \Box p, \bigcirc \Box p, p\} \end{array}$$

e as seguintes relações:

$$\begin{aligned} \eta &= \{(s_0, s_3), (s_3, s_3)\} \\ R_1 &= \{(s_0, s_1), (s_1, s_1)\} \\ R_2 &= \{(s_0, s_2), (s_2, s_2)\} \end{aligned}$$

A verificação da satisfatibilidade da propriedade $\Box p$ é trivial, já que ela é *imediatamente satisfeita* no único estado inicial, s_0.

Exemplo 2

Usando a estrutura H_1 apresentada no exemplo anterior, vamos verificar se $\mathrm{K}_1\, q$ é consequência lógica de $\mathrm{K}_1\, \Box p \wedge \mathrm{K}_2\, \Box p$. Para isso, negamos a propriedade, obtendo $\neg \mathrm{K}_1\, q$, cujo conjunto de PC-Tableaux $\mathcal{F}$ é $\{\{\neg \mathrm{K}_1\, q\}\}$. Tomando a união de $\{\neg \mathrm{K}_1\, q\}$ com s_0, obtemos $\Gamma_0 = \{\mathrm{K}_1\, \Box p \wedge \mathrm{K}_2\, \Box p, \mathrm{K}_1\, \Box p, \mathrm{K}_2\, \Box p,\ \Box p, \bigcirc \Box p, p, \neg \mathrm{K}_1\, q\}$. Criamos o estado s'_0 e a estrutura é então modificada, onde passamos a ter $S = \{s'_0, s_1, s_2, s_3\}$, $\eta = \{(s'_0, s_3), (s_3, s_3)\}$, $R_1 = \{(s'_0, s_1), (s_1, s_1)\}$ e $R_2 = \{(s'_0, s_2), (s_2, s_2)\}$.

Observe que nenhum dos sucessores epistêmicos de s'_0, dados pela relação R_1 satisfaz imediatamente $\neg \mathrm{K}_1\, q$. Portanto, aplicamos o Passo (2), criando o conjunto:

$$\Gamma_1 = \{\neg q, \neg \mathrm{K}_1\, q, \mathrm{K}_1\, \Box p,\ \Box p\}$$

e obtendo o seguinte conjunto de PC-Tableaux:

$$\mathcal{F}' \;=\; \{\Gamma_2 = \{\neg q, \neg \mathrm{K}_1\, q, \mathrm{K}_1\, \Box p,\ \Box p, \bigcirc \Box p, p\}\}$$

Criamos o estado s_4, fazemos $L(s_4) = \Gamma_2$ e atualizamos a estrutura, obtendo $S = \{s_0, s_1, s_2, s_3, s_4\}$ e $R_1 = \{(s'_0, s_1), (s_1, s_1), (s'_0, s_4)\}$, observando que η e R_2 não são modificadas. Aplicando novamente o Passo 2 a este novo estado, observa-se que o resultado requer apenas que atualizemos a relação R_1 incluindo (s_4, s_4), ou seja, $R_1 = \{(s'_0, s_1), (s_1, s_1), (s'_0, s_4), (s_4, s_4)\}$. Resta agora criar o sucessor temporal de s_4. Isto é feito pela aplicação do algoritmo do PC-Tableaux a $next(L(s_4)) = next(\Gamma_2) = \{\,\Box p\}$, que resulta em

$$\Gamma_3 = \{\,\Box p, \bigcirc \Box p, p\} \tag{5.1}$$

Temos que $L(s_3) = \Gamma_3$, logo atualizamos a estrutura, obtendo $S = \{s_0, s_1, s_2, s_3, s_4\}$ e $\eta = \{(s'_0, s_3), (s_3, s_3), (s_4, s_3)\}$. As relações R_1 e R_2 não são modificadas. Isto termina a construção do tableau, já que não há mais contrações a serem feitas. Como existe um estado inicial, no caso s'_0 que satisfaz a negação da propriedade, a mesma não é consequência lógica de $\mathrm{K}_1\, \Box p \wedge \mathrm{K}_2\, \Box p$

Exemplo 3

Usando a estrutura H_1 para $\mathrm{K}_1\, \Box p \wedge \mathrm{K}_2\, \Box p$ construída no Exemplo 1, vamos verificar se a fórmula $\mathrm{K}_1 \neg \mathrm{K}_2\, q$ é consequência lógica de $\mathrm{K}_1\, \Box p \wedge \mathrm{K}_2\, \Box p$.
Inicialmente, negamos a fórmula da propriedade e obtemos $\neg \mathrm{K}_1 \neg \mathrm{K}_2\, q$. O conjunto PC-Tableaux $\mathcal{F}$ é $\{\{\neg \mathrm{K}_1 \neg \mathrm{K}_2\, q\}\}$. Tomando a união de $\{\neg \mathrm{K}_1 \neg \mathrm{K}_2\, q\}$ com s_0, obtemos $\Gamma_0 = \{\mathrm{K}_1\, \Box p \wedge \mathrm{K}_2\, \Box p, \mathrm{K}_1\, \Box p, \mathrm{K}_2\, \Box p,\ \Box p, \bigcirc \Box p, p, \neg \mathrm{K}_1 \neg \mathrm{K}_2\, q\}$.
Criamos o estado s'_0 e a estrutura é então modificada, onde passamos a ter $S = \{s'_0, s_1, s_2, s_3\}$, $\eta = \{(s'_0, s_3), (s_3, s_3)\}$, $R_1 = \{(s'_0, s_1), (s_1, s_1)\}$ e $R_2 = \{(s'_0, s_2), (s_2, s_2)\}$.

Como nenhum dos sucessores epistêmicos de s'_0, dados pela relação R_1 satisfaz imediatamente $\neg \mathrm{K}_1 \neg \mathrm{K}_2\, q$. Portanto, aplicamos o Passo (2), criando o conjunto:

$$\Gamma_1 = \{\neg\neg \mathrm{K}_2\, q, \neg \mathrm{K}_1 \neg \mathrm{K}_2\, q, \mathrm{K}_1\ \Box p,\ \Box p\}$$

e obtendo o seguinte conjunto de PC-Tableaux:

$$\mathcal{F}' \ = \ \{\Gamma_2 = \{\mathrm{K}_2\, q, \neg \mathrm{K}_1 \neg \mathrm{K}_2\, q, \mathrm{K}_1\ \Box p,\ \Box p, \bigcirc \Box p, p, q\}\}$$

Criamos o estado s_4, fazemos $L(s_4) = \Gamma_2$ e atualizamos a estrutura, obtendo $S = \{s_0, s_1, s_2, s_3, s_4\}$ e $R_1 = \{(s'_0, s_1), (s_1, s_1), (s'_0, s_4)\}$, observando que η e R_2 não são modificadas. Aplicando novamente o Passo 2 a este novo estado e sobre as fórmulas do agente 1, observa-se que o resultado requer apenas que atualizemos a relação R_1 incluindo (s_4, s_4), ou seja, $R_1 = \{(s'_0, s_1), (s_1, s_1), (s'_0, s_4), (s_4, s_4)\}$. Resta agora criar o sucessor temporal de s_4. Isto é feito pela aplicação do algoritmo do PC-Tableaux a $next(L(s_4)) = next(\Gamma_2) = \{\ \Box p\}$, que resulta em

$$\Gamma_3 = \{\ \Box p, \bigcirc \Box p, p\} \tag{5.2}$$

Temos que $L(s_3) = \Gamma_3$, logo atualizamos a estrutura, obtendo $S = \{s_0, s_1, s_2, s_3, s_4\}$ e $\eta = \{(s'_0, s_3), (s_3, s_3), (s_4, s_3)\}$. As relações R_1 e R_2 não são modificadas.
Aplicando novamente o Passo 2 ao estado s_4 sobre as fórmulas do agente 2, criamos o conjunto:

$$\Gamma_4 = \{\mathrm{K}_2\, q, q\}$$

e obtendo o seguinte conjunto de PC-Tableaux:

$$\mathcal{F}' \ = \ \{\Gamma_5 = \{\mathrm{K}_2\, q, q\}\}$$

Criamos, então, o estado s_6 com $L(s_6) = \Gamma_5$ e atualizamos a estrutura, obtendo $S = \{s_0, s_1, s_2, s_3, s_4, s_5, s_6\}$ e $R_2 = \{(s_0, s_2), (s_2, s_2), (s_4, s_6)\}$. As relações R_1 e η não são modificadas. Aplicando novamente o Passo 2 a este novo estado e sobre as fórmulas do agente 2, observa-se que o resultado requer apenas que atualizemos a relação R_2 incluindo (s_6, s_6), ou seja, $R_1 = \{(s'_0, s_1), (s_1, s_1), (s'_0, s_4), (s_4, s_4), (s_6, s_6)\}$.
Isto termina a construção do tableau, já que não há mais contrações a serem feitas. Como existe um estado inicial, no caso s'_0 que satisfaz a negação da propriedade, a mesma não é consequência lógica de $\mathrm{K}_1\ \Box p \wedge \mathrm{K}_2\ \Box p$

Exemplo 4

Dada a estrutura H_2 do tableau para

$$\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p \tag{5.3}$$

onde a estrutura H_2 é composta pelos seguintes estados e relações:

$$
\begin{aligned}
s_0 &= \{\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p, \neg \mathrm{K}_1\ \Box p\} \\
s_1 &= \{\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p, \neg \mathrm{K}_2\ \Box p\} \\
s_2 &= \{\neg \mathrm{K}_1\ \Box p, \Diamond \neg p, \neg p\} \\
s_3 &= \{\neg \mathrm{K}_1\ \Box p, \Diamond \neg p, p \wedge \bigcirc \Diamond \neg p, p, \bigcirc \Diamond \neg p\} \\
s_4 &= \{\Diamond \neg p, \neg p\} \\
s_5 &= \{\Diamond \neg p, p \wedge \bigcirc \Diamond \neg p, p, \bigcirc \Diamond \neg p\} \\
s_6 &= \{\neg \mathrm{K}_2\ \Box p, \Diamond \neg p, \neg p\} \\
s_7 &= \{\neg \mathrm{K}_2\ \Box p, \Diamond \neg p, p \wedge \bigcirc \Diamond \neg p, p, \bigcirc \Diamond \neg p\} \\
R_1 &= \{(s_0, s_3), (s_0, s_1), (s_2, s_2), (s_2, s_3), (s_3, s_2), (s_3, s_3)\} \\
R_2 &= \{(s_1, s_7), (s_1, s_6), (s_6, s_6), (s_6, s_7), (s_7, s_6), (s_7, s_7)\} \\
\eta &= \{(s_3, s_5), (s_3, s_4), (s_7, s_5), (s_7, s_4), (s_5, s_5), (s_5, s_4)\}
\end{aligned}
$$

A partir de H_2 vamos verificar se a propriedade $\Box \neg p$ é consequência lógica de $\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p$. Inicialmente, negamos a fórmula da propriedade e obtemos $\Diamond p$. O conjunto PC-Tableaux $\mathcal{F}$ é $\{\{\Diamond p, p\}, \{\Diamond p, \neg p \wedge \bigcirc \Diamond p, \neg p, \bigcirc \Diamond p\}\}$. Tomando a união de $\mathcal{F}$ com s_0, que escolhemos como o estado inicial, obtemos:

$$\Gamma_0 = \{\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p, \neg \mathrm{K}_1\ \Box p, \Diamond p, p\}$$

e

$$\Gamma_1 = \{\neg \mathrm{K}_1\ \Box p \vee \neg \mathrm{K}_2\ \Box p, \neg \mathrm{K}_1\ \Box p, \Diamond p, \neg p \wedge \bigcirc \Diamond p, \neg p, \bigcirc \Diamond p\}.$$

Criamos os estados s'_0 ($L(s'_0) = \Gamma_0$) e s'_1 ($L(s'_1) = \Gamma_1$) e a estrutura é então modificada e passamos a ter:

$$
\begin{aligned}
S &= \{s'_0, s'_1, s_1, s_2, s_3\} \\
\eta &= \{(s_3, s_5), (s_3, s_4), (s_7, s_5), (s_7, s_4), (s_5, s_5), (s_5, s_4)\}, \\
R_1 &= \{(s'_0, s_3), (s'_1, s_3), (s'_0, s_1), (s'_1, s_1), (s_2, s_2), (s_2, s_3), (s_3, s_2), (s_3, s_3)\} \\
R_2 &= \{(s_1, s_7), (s_1, s_6), (s_6, s_6), (s_6, s_7), (s_7, s_6), (s_7, s_7)\}.
\end{aligned}
$$

Não há mais expansões para serem feitas sobre o estado s'_0 por não possuir novas fórmulas epistêmicas ou temporais. Resta agora criar o sucessor temporal de s'_1 já que temos a nova fórmula $\bigcirc \Diamond p$. Isto é feito pela aplicação do algoritmo do PC-Tableaux a $next(L(s'_1)) = next(\Gamma_2) = \{\Diamond p\}$, que resulta em

$$\Gamma_3 = \{\Diamond p, p\} \tag{5.4}$$

$$\Gamma_4 = \{\Diamond p, \neg p \wedge \bigcirc \Diamond p, \neg p, \bigcirc \Diamond p\} \tag{5.5}$$

Criamos os estados s'_2 ($L(s'_2) = \Gamma_3$) e s'_3 ($L(s'_3) = \Gamma_4$) e a estrutura é então modificada, ou seja, obtemos:

$$
\begin{aligned}
S &= \{s'_0, s'_1, s'_2, s'_3, s_1, s_2, s_3\} \\
\eta &= \{(s_3, s_5), (s_3, s_4), (s_7, s_5), (s_7, s_4), (s_5, s_5), (s_5, s_4), (s'_1, s'_2), (s'_1, s'_3)\} \\
R_1 &= \{(s'_0, s_3), (s'_1, s_3), (s'_0, s_1), (s'_1, s_1), (s_2, s_2), (s_2, s_3), (s_3, s_2), (s_3, s_3)\} \\
R_2 &= \{(s_1, s_7), (s_1, s_6), (s_6, s_6), (s_6, s_7), (s_7, s_6), (s_7, s_7)\}.
\end{aligned}
$$

Não há mais expansões para serem feitas sobre o estado s'_2 por não possuir novas fórmulas epistêmicas ou temporais tipo *next*. Resta agora criar o sucessor temporal de s'_3 já que temos a nova fórmula $\bigcirc\diamondsuit p$. Isto é feito pela aplicação do algoritmo do PC-Tableaux a $next(L(s'_3)) = next(\Gamma_5) = \{\diamondsuit p\}$, que resulta em dois conjuntos de fórmulas iguais a Γ_3 e Γ_4, que geraram os estados s'_2 e s'_3. Portanto, a relação temporal é somente atualizada, resultando em:

$$\eta = \{(s_3, s_5), (s_3, s_4), (s_7, s_5), (s_7, s_4), (s_5, s_5), (s_5, s_4), (s'_1, s'_2), (s'_1, s'_3), (s'_3, s'_3), (s'_3, s'_2)\}.$$

Não há mais passos de expansão para serem feitos. No passo de contração iremos eliminar inicialmente o estado s'_2 por não ser *resolvível* sobre a fórmula $\diamondsuit p$. Com isto, temos que s'_3 também será eliminado por não ser *resolvível* sobre a fórmula $\diamondsuit p$. Da mesma forma, após estas eliminiações, temos que o estado s'_1 também será removido por não ser *resolvível* sobre a fórmula $\diamondsuit p$. O estado s'_0 é removido da mesma forma qe o estado s'_2, por não ser *resolvível* sobre a fórmula $\diamondsuit p$. Neste ponto, os estados iniciais gerados com a escolha do estado inicial s_0 foram eliminados. Dessa forma, o algoritmo escolhe outro estado inicial que neste caso é o estado s_1. Temos que o mesmo irá acontecer ao escolhermos este estado e incluírmos a fórmula negada da propriedade, resultando em um tableau vazio. Logo, pela definição de tableau *bem-sucedido* temos que a propriedade é consequência lógica de $\neg \mathrm{K}_1 \ \Box p \vee \neg \mathrm{K}_2 \ \Box p$.

5.2 Correção

A prova de correção do método otimizado é baseada na prova de correção do algoritmo de construção do tableau dada em (WDF98). Usaremos estes resultados para mostrar que uma determinada propriedade é consequência lógica de uma especificação se o algoritmo otimizado produz uma estrutura bem-sucedida a partir da estrutura da especificação para a propriedade. Especificamente, utilizaremos os seguintes resultados de (WDF98):

Teorema 1 (Theorem 2, (WDF98)) *Se $\varphi \in FBF$ e $H = (S, \eta, R_1, \ldots, R_n, L)$ é um tableau para φ, então φ é satisfatível em $KL_{(n)}$.*

A prova do Teorema 1 baseia-se nos seguintes passos: define-se o conjunto de estados de um modelo a partir do conjunto de estados de uma estrutura e fazendo com que a função de avaliação retorne verdadeiro se, e somente se, o símbolo proposicional pertence ao estado; mostra-se como se construir um conjunto de linhas do tempo a partir da relação η de modo que todas as eventualidades sejam satisfeitas; mostra-se como construir as relações de acessibilidade de modo que estas sejam relações de equivalência; por fim, mostra-se que se φ pertence a um estado inicial de H, então φ é satisfeita no modelo assim construído. Detalhes de como modificar a prova para lidar com $BL_{(n)}$ também são dados.

Dado este resultado, aqui, como em (WDF98, Lemma 4), para finalizar a prova de correção do método de prova é preciso mostrar que a estrutura retornada pelo procedimento de fato retorna um tableau, ou seja, é preciso mostrar que a construção atende a todas as condições dadas na Definição 29.

Teorema 2 *Se* T_Δ *é um tableau* $KL_{(n)}$ *(resp.* $BL_{(n)}$*) bem-sucedido, obtido pelo Algoritmo 2 para um conjunto de fórmulas* Δ*,* φ *uma fórmula e* $\mathrm{T}_{O(\Delta,\varphi)}$*, a estrutura obtida pelo Algoritmo 3 a partir de* T_Δ *e* φ*, é bem-sucedida, então também é um tableau* $KL_{(n)}$*.*

Prova do Teorema 2

Para mostrar que $\mathrm{T}_{O(\Delta,\varphi)}$ é um tableau, devemos mostrar que a construção dada pelo Algoritmo 3 é correta, ou seja, satisfaz todas as condições dadas pela Definição 29. A prova deve ser feita para os estados construídos pelo algoritmo.

Primeiramente, lembramos que todo estado s'_j em $\mathrm{T}_{O(\Delta,\varphi)}$ é construído ou pelo Passo (1) do Algoritmo 3, incluindo fórmulas em estados já existentes na estrutura, ou pelo Algoritmo 1 por chamadas dos Passos (2) e (3), incluindo estados ainda não existentes na estrutura. A construção destes últimos segue exatamente todos os passos de construção dados pelo Algoritmo 2, já que a construção dos PC-tableaux, que correspondem aos estados sucessores, é feito para *todas* as fórmulas em $L(s'_j)$ (tanto no Passo (2) para fórmulas epistêmicas, quanto no passo (3) para fórmulas temporais). Segue, portanto, de (WDF98, Lemma 4), que as Condições (1) a (12) da Definição 29 são atendidas.

Resta mostrar que os demais estados, aqueles modificados pelo Passo (1) também atendem estas condições. Para facilitar a leitura da prova estas condições serão reapresentadas aqui. Seja s'_j o estado obtido de $s \in \mathrm{T}_\Delta$ tal que $L(s'_j) = L(s) \cup \Gamma_j$. Pela correção do Algoritmo 1, é imediato que todas as fórmulas em $L(s)$ atendem às Condições (1) a (12). Sem perda de generalidade, iremos considerar que $L(s)$ e Γ_j são disjuntos e mostraremos o que falta, ou seja, que as fórmulas em Γ_j atendem as Condições (1) a (12).

1. $\exists s'_j \in S$ tal que $\Delta \subseteq L(s'_j)$ e $\varphi \in L(s'_j)$: imediato da definição de estrutura bem-sucedida.

2. $L(s'_j)$ é um conjunto *próprio*: todo conjunto que não seja próprio é descartado pelo Algoritmo 1 e pelo Passo (1) do Algoritmo 3.

3. Se $\psi \in L(s'_j)$ e ψ é um fórmula do tipo α com componentes α_1 e α_2, então $\alpha_1 \in L(s'_j)$ e $\alpha_2 \in L(s'_j)$: imediato, pela correção do Algoritmo 1.

4. Se $\psi \in L(s'_j)$ e ψ é um fórmula do tipo β com componentes β_1 e β_2, então $\beta_1 \in L(s'_j)$ ou $\beta_2 \in L(s'_j)$: imediato, pela correção do Algoritmo 1.

5. Se $\mathrm{K}_i\,\psi \in L(s'_j)$ ou $\neg\mathrm{K}_i\,\psi \in L(s'_j)$ e $\mathrm{K}_i\,\varphi' \in sub(\psi)$, então $\mathrm{K}_i\,\varphi' \in L(s'_j)$ ou $\neg\mathrm{K}_i\,\varphi' \in L(s'_j)$: o Passo (2), juntamente com a correção do Algoritmo 1, garantem que s'_j é um PC-tableau e, portanto, subfórmula completo.

6. Se $\mathrm{K}_i\,\psi \in L(s'_j)$ então $\forall s' \in S$, se $(s'_j, s') \in R_i$, então $\psi \in L(s')$: o Passo (2), juntamente com a correção do Algoritmo 1, garantem que todos os sucessores epistêmicos de s'_j contém ψ para todas as subfórmulas epistêmicas de s'_j.

7. Se $\neg\mathrm{K}_i\,\psi \in L(s'_j)$ então $\exists s' \in S$ tal que $(s'_j, s') \in R_i$ e $\neg\psi \in L(s')$: primeiramente, deve existir $(s'_j, s') \in R_i$, pois caso contrário,$\neg\mathrm{K}_i\,\psi$ não seria satisfeita e o Passo (4c) do Algoritmo 3 teria removido s'_j. Segundo, se s' é sucessor de s'_j, então deve ter sido criado pelo Passo (1) a partir de s'_j, incluindo todas as subfórmulas epistêmicas em $L(s'_j)$ e este passo garante que todos os conjuntos gerados são subfórmula completos (pela correção do Algoritmo 1). Logo, $\neg\psi \in L(s')$.

8. Se em $KL_{(n)}$

   (a) Se $\mathrm{K}_i\,\psi \in L(s)$ então $\psi \in L(s)$: imediato, pela correção do Algoritmo 1, já que todas as regras α são aplicadas.

   (b) Para todo s', se $(s'_j, s') \in R_i$, então $\mathrm{K}_i\,\psi \in L(s'_j)$ se, e somente se, $\mathrm{K}_i\,\psi \in L(s')$: se $(s'_j, s') \in R_i$, então s' foi criado a partir de s'_j e, pelo Passo (2) e pela correção do Algoritmo 1, todas as fórmulas epistêmicas de s'_j estão em s e vice-versa.

9. Se em $BL_{(n)}$:

   (a) Se $(s'_j, s'), (s'_j, s'') \in R_i$ e $\mathrm{K}_i\,\psi \in L(s')$, então $\{\mathrm{K}_i\,\psi, \psi\} \in L(s'')$: Se $(s'_j, s'), (s'_j, s'') \in R_i$ e $\mathrm{K}_i\,\psi \in \Gamma_j$, então ambos s' e s'' foram criados a partir de Γ_j pela aplicação do Passo (2) do Algoritmo 3. Mas esta aplicação requer que $\mathrm{K}_i\,\psi$ seja uma fórmula em Γ_j, pois pela correção do Algoritmo 1 para a construção dos PC-Tableaux, todas as subfórmulas epistêmicas desta forma fazem parte de Γ_j e o Passo (2) para a construção dos sucessores epistêmicos é feita a partir destas subfórmulas. Além disso, pelo Passo (2), todos os sucessores de Γ_j possuem as mesmas fórmulas epistêmicas de Γ_j. Portanto, $\{\mathrm{K}_i\,\psi, \psi\} \subseteq L(s'')$, o que prova que a condição é satisfeita.

   (b) Se $\mathrm{K}_i\,\psi \in L(s'_j)$, então $\exists s' \in S$ tal que $(s'_j, s') \in R_i$: imediato, do Passo (4d) do Algoritmo 3.

   (c) Se $\mathrm{K}_i\,\psi \in L(s'_j)$ e $(s'_j, s') \in R_i$, então $\mathrm{K}_i\,\psi \in L(s')$: Se $\mathrm{K}_i\,\psi \in \Gamma_j$ e $(s'_j, s') \in R_i$, então s' contém as mesmas fórmulas epistêmicas de s'_j, pelo Passo (2) do Algoritmo 3 e pela correção do Algoritmo 1. Logo, $\mathrm{K}_i\,\psi \in L(s')$.

10. Se $\bigcirc\psi \in L(s)$, então $\exists s' \in S$ tal que $(s, s') \in \eta$: imediato, pelo Passo (4b) do Algoritmo 3.

11. Se $\bigcirc\psi \in L(s)$, então $\forall s' \in S$, se $(s, s') \in \eta$, então $\psi \in L(s')$: se $\bigcirc\psi \in \Gamma_j$, então, pela aplicação do Passo (3) e pela correção do Algoritmo 1, todos os sucessores temporais de Γ_j contém ψ.

12. Se $\psi \in L(s)$, então $resolvable(\psi, s, (S, \eta, R_1, \ldots, R_n, L))$: imediato, pelo Passo (4a) do Algoritmo 3.

□

A partir dos Teoremas 1 e 2, podemos estabelecer o seguinte resultado.

Teorema 3 *Se* T_Δ *é um tableau* $KL_{(n)}$ *(resp.* $BL_{(n)}$*) bem-sucedido, obtido pelo Algoritmo 2 para um conjunto de fórmulas* Δ*,* φ *uma fórmula e* $\mathrm{T}_{O(\Delta,\varphi)}$*, a estrutura obtida pelo Algoritmo 3 a partir de* T_Δ *e* φ*, é bem-sucedida, então* $\Delta \cup \{\varphi\}$ *é satisfatível.*

Ou seja, se obtivermos uma estrutura bem-sucedida, então é possível aplicar as construções apresentadas em (WDF98) e apresentar um modelo para $\Delta \cup \{\varphi\}$. Esperamos também obter a prova de completude do método apresentado em um momento futuro.

Capítulo 6

Conclusão

A verificação formal possui um papel muito importante na Ciência da Computação, em que diferentes metodologias de verificação contribuem para melhorar a segurança e confiabilidade dos sistemas e ferramentas computacionais que permeiam cada vez mais o dia a dia das pessoas. É de grande relevância, portanto prover soluções eficientes para o problema de verificar se uma especificação ou propriedades de um sistema, protocolo ou processo é correto.

Este trabalho tem como contribuição principal a implementação do método de prova baseado em tableaux para as lógicas multimodais $KL_{(n)}$ e $BL_{(n)}$ proposto em (WDF98). Foi proposto ainda um algoritmo que busca minimizar o trabalho de verificação usando este método. Este algoritmo descreve um modo de verificar propriedades sobre um tableau obtido pelo método (WDF98), sem que seja preciso construir um novo tableau para o conjunto inicial de fórmulas e a propriedade.

Como trabalhos futuros, pretendemos demonstrar a completude do método aqui proposto, bem como realizar sua implementação e experimentação sobre *benchmarks* robusto, como por exemplo algum dos conjunto de fórmulas/problemas encontrados em TPTP (*Thousands of Problems for Theorem Provers*) (Sut09). Pretende-se também comparar a performance da nossa implementação em relação a ferramentas computacionais semelhantes (KNN$^+$07). Futuras extensões para outros tipos de lógicas que possuem métodos de prova baseado em tableaux serão consideradas. Além disso, com relação à implementação, diversas otimizações podem ser realizadas como por exemplo: uso de *threads*, com *Jython* (Pyt14a), em algumas partes da construção do tableaux e extensão das operações mais lentas da implementação utilizando o *binding* do *Python* para a linguagem C.

Referências

[BEL+01] M. Baaz, U. Egly, A. Leitsch, J. Goubault-Larrecq, and D. Plaisted. Chapter 5 - normal form transformations. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*, Handbook of Automated Reasoning, pages 273 – 333. North-Holland, Amsterdam, 2001. 16

[Bio14] Biopython developers. Biopython is a set of freely available tools for biological computation written in python by an international team of developers. `http://biopython.org/`, Novembro 2014. 27

[Bou11] I. C. Boureanu. *Model Checking Security Protocols: A Multiagent System approach.* PhD thesis, Imperial College London, 2011. 1, 2, 7

[Bra14] G. Brandl. Sphinx is a tool that makes it easy to create intelligent and beautiful documentation. `http://sphinx-doc.org/`, Novembro 2014. 27, 31

[CFdCGH97] M. Castilho, L. Fariñas del Cerro, O. Gasquet, and A. Herzig. Modal tableaux with propagation rules and structural rules. *Fundamenta Informaticae*, 32(3/4), 1997. 2

[Che80] B. F. Chellas. *Modal Logic: an introduction.* Cambridge University Press, 1980. 1

[DFGFvdH07] C. Dixon, M.-C. Fernández-Gago, M. Fisher, and W. van der Hoek. Temporal logics of knowledge and their applications in security. *Eletronic Notes in Theoretical Computer Science*, 182:27–42, 2007. 1, 2, 4, 7, 10

[DGHP99] M. D'Agostino, D. Gabbay, R. Hähnle, and J. Posegga, editors. *Handbook of Tableau Methods.* Kluwer, Dordrecht, 1999. Reviewed in: *Journal of Logic, Language and Information*, 10(4), pages 518–523, 2001, by Maarten de Rijke; *Mathematical Reviews* #2002b:03001 by Henry Africk. 2

[DLKH96] F. Dietrich, X. Logean, S. Koppenhoffer, and J. Hubaux. Testing temporal logic properties in distributed systems. In *IFIP International Workshop on Testing of Communicating Systems (IWTCS)*, pages 247–262. Kluwer Academic Publishers, 1996. 1, 7

[Doc14] Docopt developers. Docopt - command-line interface description language. `http://www.docopt.org/`, Novembro 2014. 27

[DWFZ12] C. Dixon, A. F. T. Winfield, M. Fisher, and C. Zeng. Towards temporal verification of swarm robotic systems. *Robot. Auton. Syst.*, 60(11):1429–1441, November 2012. 1, 7

[FdCFG+01] L. Fariñas del Cerro, D. Fauthoux, O. Gasquet, A. Herzig, D. Longin, and F. Massacci. Lotrec: the generic tableau prover for modal and description logics. In *International Joint Conference on Automated Reasoning*, LNCS, page 6. Springer Verlag, 18-23 juin 2001. 2

[FHMV95] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. MIT Press, 1995. 1, 2, 4, 7, 10, 14

[Fit06] M. Fitting. Modal proof theory. In P. Blackburn, J. F. A. K. v. Benthem, and F. Wolter, editors, *Handbook of Modal Logic, Volume 3 (Studies in Logic and Practical Reasoning)*. Elsevier Science Inc., New York, NY, USA, 2006. 32

[Gen69] G. Gentzen. Investigations into Logical Deduction. In M. E. Szabo, editor, *The Collected Papers of Gerhard Gentzen*, pages 68–213. North-Holland, Amsterdam, 1969. 2

[GL13] A. Griesmayer and A. Lomuscio. Model checking distributed systems against temporal-epistemic specifications. In D. Beyer and M. Boreale, editors, *Formal Techniques for Distributed Systems*, volume 7892 of *Lecture Notes in Computer Science*, pages 130–145. Springer Berlin Heidelberg, 2013. 1, 2, 7, 10

[GPSS80] D. Gabbay, A. Pnueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *POPL '80: Proceedings of the 7th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 163–173, New York, NY, USA, 1980. ACM. 1, 8

[HM92] J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artif. Intell.*, 54(3):319–379, 1992. 1

[KNN+07] M. Kacprzak, W. Nabialek, A. Niewiadomski, W. Penczek, A. Pólrola, M. Szreter, B. Wozna, and A. Zbrzezny. Verics 2006 - a model checker for real-time and multi-agent systems. In *In Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'07)*, page 345–356, 2007. 43

[Li08] Z. Li. *Efficient and Generic Reasoning for Modal Logics*. PhD thesis, University of Manchester, 2008. 2

[MP92] Z. Manna and A. Pnueli. *The temporal logic of reactive and concurrent systems*. Springer-Verlag New York, Inc., New York, NY, USA, 1992. 1, 7

[Net14] NetworkX developer team. Networkx - high-productivity software for complex networks. https://networkx.github.io/, Novembro 2014. 27

[Num14] Numpy developers. Numpy is the fundamental package for scientific computing with python. http://www.numpy.org/, Novembro 2014. 27

[Pau90] L. C. Paulson. Isabelle: the Next 700 Theorem Provers. In P. Odifreddi, editor, *Logic and Computer Science*. Academic Press, 1990. 2

[PyG14] PyGraphviz developer team. Pygraphviz - a python interface to graphviz graph layout and visualization package. http://pygraphviz.github.io/, Novembro 2014. 27

[Pyt14a] Python Software Foundation. The jython project. http://www.jython.org/, Novembro 2014. 43

[Pyt14b] Python Software Foundation. Python. https://www.python.org/, Novembro 2014. 3, 26, 27

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, January 1965. 2

[Sag14] Sage developers. Sage is a free open-source mathematics software system licensed under the gpl. http://www.sagemath.org/, Novembro 2014. 27

[Sch14] R. Schmidt. Computational tools useful for modal logics. http://www.cs.man.ac.uk/ schmidt/tools/, Novembro 2014. 3

[Smu95] R. Smullyan. *First-order logic*. Dover Publications, 1995. 2, 16

[Stu14] R. Stuart. Lrparsing - yet another parser for python. http://lrparsing.sourceforge.net/, Novembro 2014. 27

[Sut09] G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009. 43

[TAB03] TABLEAUX. *System description: The tableaux workbench*. Springer, 2003. 2

[TSK13] D. Tishkovsky, R. A. Schmidt, and M. Khodadadi. Mettel²: Towards a tableau prover generation platform. In P. Fontaine, R. A. Schmidt, and S. Schulz, editors, *PAAR-2012*, volume 21 of *EPiC Series*, pages 149–162. EasyChair, 2013. 2

[vL14] T. van Laarhoven. Moltap is a modal logic tableau prover. http://twan.home.fmf.nl/moltap/, Novembro 2014. 2

[vVvdVV10] G. van Valkenhoef, E. van der Vaart, and R. Verbrugge. Oops: An {S5n} prover for educational settings. *Electronic Notes in Theoretical Computer Science*, 262(0):249 – 261, 2010. Proceedings of the 6th Workshop on Methods for Modalities (M4M-6 2009). 2

[WDF98] M. Wooldridge, C. Dixon, and M. Fisher. A tableau-based proof method for temporal logics of knowledge and belief. *Journal of Applied Non-Classical Logics*, 8(3):225–258, 1998. vi, vii, 2, 3, 15, 18, 19, 20, 21, 24, 29, 32, 34, 40, 41, 42, 43